z/OS

DFSMSrmm Implementation and Customization Guide

SC26-7405-04

z/OS

DFSMSrmm Implementation and Customization Guide

SC26-7405-04

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 439.

Fifth Edition, September 2004

This edition applies to Version 1 Release 6 of z/OS® (5694-A01), Version 1 Release 6 of z/OS.e (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC26-7405-03.

IBM® welcomes your comments. A form for readers’ comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About This Document

This document is intended for storage administrators and system programmers who are responsible for implementing and customizing DFSMSrmm™.

Before using this document, you should read *z/OS Migration* for detailed migration information for DFSMSrmm as well as other DFSMS™ functional components. Additionally, you should have installed DFSMSrmm with SMP/E using the directions in *z/OS Program Directory*.

How to Use this Document

This document explains how to implement DFSMSrmm for users with no existing media management system.

Refer to the following chapters to implement DFSMSrmm:

Chapter 1, "Introducing DFSMSrmm," on page 1
Chapter 2, "Implementing DFSMSrmm," on page 23
| Chapter 3, "Setting Up DFSMSrmm Client and Server Systems," on page 57
Chapter 4, "Organizing the Removable Media Library," on page 63
Chapter 5, "Running DFSMSrmm with System-Managed Tape Libraries," on page 85
Chapter 6, "Running DFSMSrmm with BTLS," on page 113
Chapter 7, "Managing Storage Locations," on page 117
Chapter 8, "Using the Parmlib Member EDGRMMxx," on page 127
Chapter 9, "Authorizing DFSMSrmm Users and Ensuring Security," on page 171
Chapter 10, "Using DFSMSrmm Programming Interfaces," on page 199
Chapter 11, "Using DFSMSrmm Installation Exits," on page 221
Chapter 12, "Running DFSMSrmm with DFSMShsm," on page 253
Chapter 13, "Running DFSMSrmm with JES3," on page 271
Chapter 14, "Performing Inventory Management," on page 275
Chapter 15, "Maintaining the Control Data Set," on page 317
Chapter 16, "Initializing and Erasing Tape Volumes," on page 351
Appendix A, "DFSMSrmm Installation Verification Procedures," on page 403

To set up and run DFSMSrmm utilities, refer to:

Chapter 14, "Performing Inventory Management," on page 275
Chapter 15, "Maintaining the Control Data Set," on page 317
Chapter 16, "Initializing and Erasing Tape Volumes," on page 351

To customize DFSMSrmm, refer to:

Chapter 10, "Using DFSMSrmm Programming Interfaces," on page 199
Chapter 11, "Using DFSMSrmm Installation Exits," on page 221
Chapter 17, "Customizing DFSMSrmm," on page 373
Chapter 18, "Using the Problem Determination Aid Facility," on page 385
Chapter 19, "Setting Up DFSMSrmm Disposition Processing," on page 389
Chapter 20, "Running DFSMSrmm with the IBM Tivoli Workload Scheduler for z/OS," on page 395

How to Find Samples in this Document

Throughout this document, when a task has an available sample, we point you to the sample using a figure like the one that follows:

DFSMSrmm Sample Provided in SAMPLIB

CBRUXENT Programming Interface to EDGLCSUX

See Appendix C, "Using DFSMSrmm Samples," on page 427 for a list of the DFSMSrmm supplied samples.

Required product knowledge

You should be familiar with several products before using this document:

- RACF®, a component of the Security Server for z/OS™ for defining resources and creating access lists
- ISPF, for using and customizing the DFSMSrmm ISPF dialog
- DFSMShsm™, for customizing the interaction between DFSMShsm and DFSMSrmm
- OAM, for customizing the interface between DFSMSrmm and OAM
- DFSMSdss™, for backing up the DFSMSrmm control data set and journal

Referenced documents

The following documents have additional information about DFSMSrmm:

Document Title	Order Number
z/OS DFSMSrmm Application Programming Interface	SC26-7403
z/OS DFSMSrmm Diagnosis Guide	GY27-7619
z/OS DFSMSrmm Guide and Reference	SC26-7404
z/OS DFSMSrmm Reporting	SC26-7406

This document also refers to the following documents:

Document Title	Order Number
TCP/IP: Performance Tuning Guide	SC31-7188
z/OS Communications Server: IP Configuration Guide	SC31-8775
Basic Tape Library Support Version 1 Release 1 User's Guide and Reference	SC26-7016
Data Facility Removable Media Manager for MVS/DFP Version 3 Program Offering	SC26-7011
z/OS DFSORT Application Programming Guide	SC26-7523
IBM 3590 High Performance Tape Subsystem Introduction and Planning Guide	GA32-0330
IBM TotalStorage Enterprise Automated Tape Library (3494) Introduction and Planning Guide	GA32-0449
IBM TotalStorage Enterprise Automated (3494) Tape Library Operator Guide	GA32-0448
TotalStorage Automated Tape Library (3495) Introduction	GA32-0234
IBM TotalStorage Enterprise Automated Tape Library (3495) Operator's Guide	GA32-0235

Document Title	Order Number
MVS Batch Local Shared Resources	GC28-1469
z/OS DFSMS Installation Exits	SC26-7396
z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries	SC35-0427
z/OS DFSMS: Using Data Sets	SC26-7410
z/OS DFSMS: Using Magnetic Tapes	SC26-7412
z/OS DFSMSdfp Diagnosis Reference	GY27-7618
z/OS DFSMSdss Storage Administration Reference	SC35-0424
z/OS DFSMShsm Implementation and Customization Guide	SC35-0418
z/OS DFSMShsm Storage Administration Guide	SC35-0421
z/OS JES3 Initialization and Tuning Guide	SA22-7549
z/OS MVS JCL Reference	SA22-7597
z/OS MVS Planning: Global Resource Serialization	SA22-7600
z/OS MVS System Commands	SA22-7627
z/OS MVS System Management Facilities (SMF)	SA22-7630
z/OS MVS System Messages, Vol 1 (ABA-AOM)	SA22-7631
z/OS MVS System Messages, Vol 2 (ARC-ASA)	SA22-7632
z/OS MVS System Messages, Vol 3 (ASB-BPX)	SA22-7633
z/OS MVS System Messages, Vol 4 (CBD-DMO)	SA22-7634
z/OS MVS System Messages, Vol 5 (EDG-GFS)	SA22-7635
z/OS MVS System Messages, Vol 6 (GOS-IEA)	SA22-7636
z/OS MVS System Messages, Vol 7 (IEB-IEE)	SA22-7637
z/OS MVS System Messages, Vol 8 (IEF-IGD)	SA22-7638
z/OS MVS System Messages, Vol 9 (IGF-IWM)	SA22-7639
z/OS MVS System Messages, Vol 10 (IXC-IZP)	SA22-7640
z/OS MVS Using the Subsystem Interface	SA22-7642
OS/390 Program Directory	
z/OS Security Server RACF Security Administrator's Guide	SA22-7683
z/OS Security Server RACF System Programmer's Guide	SA22-7681
z/OS TSO/E Customization	SA22-7783
z/OS V1R6.0 ISPF User's Guide Vol I	SC34-4822
ServerPac: Installing Your Order	

Accessing z/OS DFSMS documents on the Internet

In addition to making softcopy documents available on CD-ROM, IBM provides access to unlicensed z/OS softcopy documents on the Internet. To view, search, and print z/OS documents, go to the z/OS Internet Library: http://www.ibm.com/eserver/zseries/zos/bkserv/

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM® messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, and VSE:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX® System Services running OMVS).
- Your Windows® workstation. You can install code to access IBM message explanations on the *z/OS Collection* (SK3T-4269), using LookAt from a Windows DOS command line.
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Windows workstation from a disk on your *z/OS Collection* (SK3T-4269), or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Accessing z/OS licensed documents on the Internet

z/OS licensed documentation is available on the Internet in PDF format at the IBM Resource Link™ Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed documents are available only to customers with a z/OS license. Access to these documents requires an IBM Resource Link user ID and password, and a key code. With your z/OS order you received a Memo to Licensees, (GI10-0671), that includes this key code. [1]

To obtain your IBM Resource Link user ID and password, log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed documents:

1. Sign in to Resource Link using your Resource Link user ID and password.
2. Select **User Profiles** located on the left-hand navigation bar.

Note: You cannot access the z/OS licensed documents unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

1. z/OS.e customers received a Memo to Licensees, (GI10-0684) that includes this key code.

Printed licensed documents are not available from IBM.

You can use the PDF format on either **z/OS Licensed Product Library CD-ROM** or IBM Resource Link to print licensed documents.

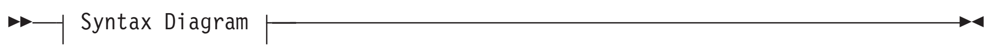
Notational conventions

This section explains the notational conventions used in this document.

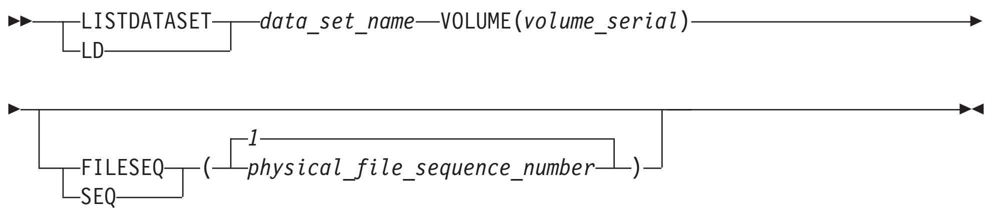
How to read syntax diagrams

Throughout this library, diagrams are used to illustrate the programming syntax. Keyword parameters are parameters that follow the positional parameters. Unless otherwise stated, keyword parameters can be coded in any order. The following list tells you how to interpret the syntax diagrams:

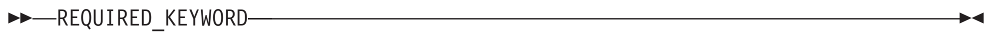
- Read the diagrams from left-to-right, top-to-bottom, following the main path line. Each diagram begins on the left with double arrowheads and ends on the right with two arrowheads facing each other.



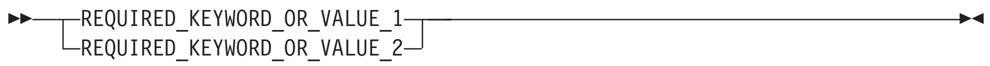
- If a diagram is longer than one line, each line to be continued ends with a single arrowhead and the next line begins with a single arrowhead.



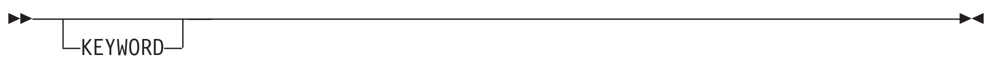
- Required keywords and values appear on the main path line. You must code required keywords and values.



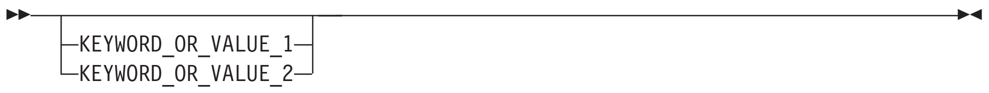
  If several mutually exclusive required keywords or values exist, they are stacked vertically in alphanumeric order.



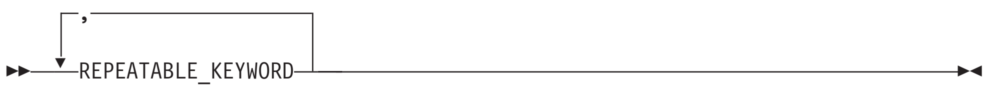
- Optional keywords and values appear below the main path line. You can choose not to code optional keywords and values.



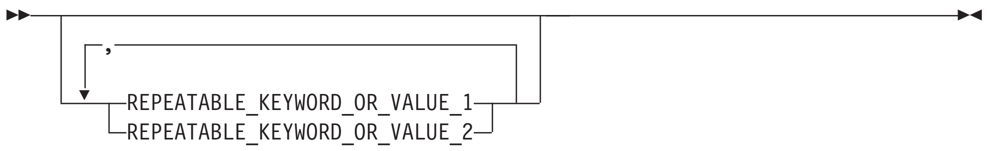
  If several mutually exclusive optional keywords or values exist, they are stacked vertically in alphanumeric order below the main path line.



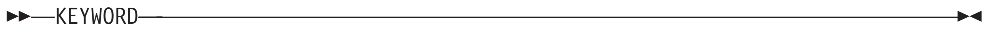
- An arrow returning to the left above a keyword or value on the main path line means that the keyword or value can be repeated. The comma means that each keyword or value must be separated from the next by a comma.



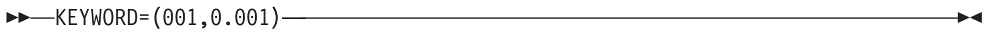
- An arrow returning to the left above a group of keywords or values means more than one can be selected, or a single one can be repeated.



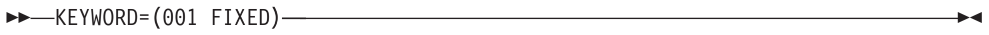
- A word in all uppercase is a keyword or value you must spell exactly as shown. In this example, you must code ***KEYWORD***.



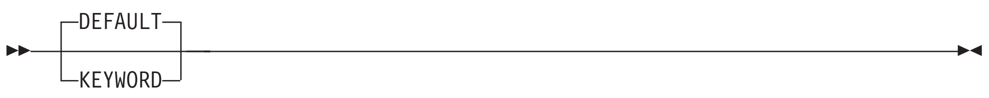
  If a keyword or value can be abbreviated, the abbreviation is discussed in the text associated with the syntax diagram.
- If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code ***KEYWORD=(001,0.001)***.



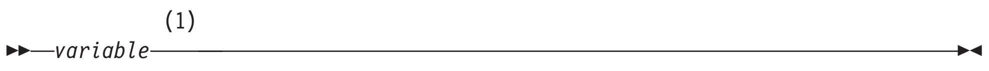
- If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code ***KEYWORD=(001 FIXED)***.



- Default keywords and values appear above the main path line. If you omit the keyword or value entirely, the default is used.



- A word in all lowercase italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.



Notes:

1 An example of a syntax note.

- References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.

```
►►──KEYWORD─────────────────────────────────────────────────────►◄
```

- Some diagrams contain *syntax fragments*, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

```
►►──┤ Reference to Syntax Fragment ├────────────────────────────►◄
```

Syntax Fragment:

```
├──1ST_KEYWORD,2ND_KEYWORD,3RD_KEYWORD──────────────────────────┤
```

The following is an example of a syntax diagram.

```
►►──┬─DELETEOWNER─┬──owner_ID──┬──────────────┬─────────────────►◄
    └─DO──────────┘            └─┤ newowner ├─┘
```

newowner

```
                           (1)
├──NEWOWNER(new_owner_ID)──────────────────────────────────────┤
```

Notes:

1 Must be specified if the owner owns one or more volumes.

The possible valid versions of the RMM DELETEOWNER command are:

```
RMM DELETEOWNER owner
RMM DO          owner
RMM DELETEOWNER owner NEWOWNER(new_owner)
RMM DO          owner NEWOWNER(new_owner)
```

How to abbreviate commands and operands

The TSO abbreviation convention applies for all DFSMSrmm commands and operands. The TSO abbreviation convention requires you to specify as much of the command name or operand as is necessary to distinguish it from the other command names or operands.

Some DFSMSrmm keyword operands allow unique abbreviations. All unique abbreviations are shown in the command syntax diagrams.

How to use continuation characters

The symbol **-** is used as the continuation character in this document. You can use either **-** or **+**.

- Do not ignore leading blanks on the continuation statement

+ Ignore leading blanks on the continuation statement

Delimiters

When you type a command, you must separate the command name from the first operand by one or more blanks. You must separate operands by one or more blanks or a comma. Do not use a semicolon as a delimiter because any character you enter after a semicolon is ignored.

Character sets

To code job control statements, use characters from the character sets in Table 1. Table 2 lists the special characters that have syntactical functions in job control statements.

Table 1. Character Sets

Character Set	**Contents**	
Alphanumeric	Alphabetic Numeric	Capital A through Z 0 through 9
National (See note)	"At" sign Dollar sign Pound sign	@ (Characters that can be $ represented by hexadecimal # values X'7C', X'5B', and X'7B')
Special	Comma Period Slash Apostrophe Left parenthesis Right parenthesis Asterisk Ampersand Plus sign Hyphen Equal sign Blank	, . / ' ( ) * & + - =
EBCDIC text	EBCDIC printable character set	Characters that can be represented by hexadecimal X'40' through X'FE'
Note: The system recognizes the following hexadecimal representations of the U.S. National characters; @ as X'7C'; $ as X'5B'; and # as X'7B'. In countries other than the U.S., the U.S. National characters represented on terminal keyboards might generate a different hexadecimal representation and cause an error. For example, in some countries the $ character may generate a X'4A'.		

Table 2. Special Characters Used in Syntax

Character	**Syntactical Function**
,	To separate parameters and subparameters
=	To separate a keyword from its value, for example, BURST=YES
(ƀ)	To enclose subparameter list or the member name of a PDS or PDSE
&	To identify a symbolic parameter, for example, &LIB
&&	To identify a temporary data set name, for example, &&TEMPDS, and, to identify an in-stream or sysout data set name, for example, &&PAYOUT
.	To separate parts of a qualified data set name, for example, A.B.C., or parts of certain parameters or subparameters, for example, nodename.userid

Table 2. Special Characters Used in Syntax (continued)

Character	Syntactical Function
*	To refer to an earlier statement, for example, OUTPUT=*.name, or, in certain statements, to indicate special functions: //label CNTL * //ddname DD * RESTART=* on the JOB statement
’	To enclose specified parameter values which contain special characters
(blank)	To delimit fields

Summary of Changes

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

You may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only or procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

Summary of Changes for SC26-7405-04 z/OS Version 1 Release 6

This document contains information previously presented in *z/OS Version 1 Release 5 DFSMSrmm Implementation and Customization* (SC26-7405-03).

The following sections summarize the changes to that information.

New Information

This edition includes the following new information:
- Added new information about setting up DFSMSrmm client/server for z/OS systems.
- Added additional considerations for running DFSMSrmm utilities when client/server support is used.
- Added new EDGRMM*xx* parmlib member OPTION command operands for client/server support and VLPOOL command operands for release processing.
- Added new samples to the list of DFSMSrmm-supplied samples.
- Added information about DFSMSrmm support for the IBM TotalStorage Enterprise Tape System 3592 WORM tapes.
  - Added information about the validation that DFSMSrmm performs for WORM tapes.
  - Added WORM pooling information.
  - Added information for the EDGRMM*xx* parmlib member VLPOOL MASTEROVERWRITE operand.

Changed Information

The following information was changed in this edition:
- Added client/server-related terms to the glossary.

Summary of Changes for SC26-7405-03 z/OS Version 1 Release 5

This document contains information previously presented in *z/OS Version 1 Release 3 DFSMSrmm Implementation and Customization* (SC26-7405-02).

The following sections summarize the changes to that information.

New Information

This edition includes the following new information:
- DFSMSrmm provides added protection for data sets and volumes by supporting the RACF name-hiding function.

- DFSMSrmm provides capabilities to set a default media name for your installation that is used when adding a volume, defining a pool, or using the DFSMSrmm utility EDGINERS to initialize or erase volumes.

Changed Information

The following information was changed in this edition:
- The DFSMSrmm utility EDGINERS EXEC parameters MEDIATYPE and RECORDINGFORMAT have been changed to support the IBM TotalStorage™ Enterprise Tape System 3592.

Summary of Changes for SC26-7405-02 z/OS Version 1 Release 3

This document contains information previously presented in *z/OS Version 1 Release 3 DFSMSrmm Implementation and Customization* (SC26-7405-01).

The following sections summarize the changes to that information.

New Information

This edition includes the following new information:
- You can back up the DFSMSrmm control data set and journal at any time. New JCL examples have been added that show how to request a back up of the DFSMSrmm control data set and how to clear the journal.
- You can control whether DFSMSrmm manages or ignores volumes based on whether they are defined to DFSMSrmm or not. You can use RACF profiles to separate authorization checking for volumes that are defined DFSMSrmm and that are not defined to DFSMSrmm.
- You can use volumes with duplicate volume serial numbers. Information and examples about how to define volumes with duplicate volume serial numbers have been added. You can also use the DFSMSrmm EDGINERS utility to initialize duplicate volumes.
- DFSMSrmm provides added protection for data sets and volumes by providing command authorization support using the DFSMSrmm EDGRMM*xx* parmlib OPTION COMMANDAUTH operand.

Changed Information

The following information was changed in this edition:
- Information about backing up and restoring the control data set and journal.
- Information about creating an extract data set has been updated as a result of Reader's Comment Form 26541.
- Information about labeling ANSI/ISO Version 3 and Version 4 tapes has been clarified.

Summary of Changes for SC26-7405-01 z/OS Version 1 Release 3

This document contains information previously presented in *z/OS Version 1 Release 1 DFSMSrmm Implementation and Customization* (SC26-7405-00).

The following sections summarize the changes to that information.

New Information

This edition includes the following new information:
- Additional DFSMSrmm resource symbolic names used with global resource serialization have been added.
- DFSMSrmm Report Generator set up information has been added.

- DFSMSrmm minimal bin support has been added.
- DFSMSrmm reporting has been enhanced to support production of an extended extract data set that contains a new type of record which is a combination of the data set record and the volume record.
- DFSMSrmm enhancements for supporting storage location management including allowing storage locations to defined as home locations.

Changed Information

The following information was changed in this edition:

- DFSMSrmm pre-ACS support has been enhanced so that you can control how policy information is assigned using the EDGRMM*xx* PARMLIB OPTION command PRE-ACS and SMSACS operands.
- The DFSMSrmm installation exit EDGUX200 has been modified with the addition of fields for description and owner information.

Moved Information

The following information was moved from this edition:

- Information about customizing DSSOPT has been moved to Chapter 15, “Maintaining the Control Data Set,” on page 317.
- DFSMSrmm system code and user code information has been moved to the *z/OS DFSMSrmm Diagnosis Guide*.

Chapter 1. Introducing DFSMSrmm

DFSMSrmm™ is a z/OS feature. In your enterprise, you probably store and manage removable media in several types of media libraries. For example, in addition to your traditional tape library, a room with tapes, shelves, and drives, you might have several automated, virtual, and manual tape libraries. You probably also have both on-site libraries and off-site storage locations, also known as vaults or stores.

With DFSMSrmm, you can manage your removable media as one enterprise-wide library across systems and sysplexes. DFSMSrmm manages your installation's tape volumes and the data sets on those volumes. DFSMSrmm also manages the shelves where volumes reside in all locations except in automated tape libraries.

DFSMSrmm manages all tape media, such as cartridge system tapes and 3420 reels, as well as other removable media you define to it. For example, DFSMSrmm can record the shelf location for optical disks and track their vital record status; it does not manage the objects on optical disks.

This chapter discusses basic tape management concepts and introduces terminology used throughout the DFSMSrmm publications and in the DFSMSrmm Interactive System Productivity Facility (ISPF) dialog. This chapter also discusses tape management tasks you can perform with DFSMSrmm.

What is a RMMplex?

An RMMplex is one or more MVS systems each running a DFSMSrmm subsystem sharing a control data set. An RMMplex can optionally include one or more DFSMSrmm subsystems as servers, one or more client subsystems, in addition to standard DFSMSrmm subsystems. The server subsystems and standard subsystems have direct access to and share the DFSMSrmm control data set. The client systems have no direct access to the DFSMSrmm control data set, but share the control data set via the server. All systems that share a control data set in this way are part of the same RMMplex.

What Libraries and Locations Can DFSMSrmm Manage?

You decide where to store your removable media based on how often you access the media and for what purpose you retain the media. For example, you might keep volumes that are frequently accessed in an automated tape library. You probably use at least one storage location to retain volumes for disaster recovery and audit purposes. You might also have locations to which you send volumes for further processing. These locations might be other data centers within your company or at your customers and vendors locations.

DFSMSrmm can manage:

- A removable media library, which incorporates all other libraries, such as:
  - System-managed tape libraries; for example, the automated IBM TotalStorage™ Enterprise Automated Tape Library (3494), IBM TotalStorage Virtual Tape Servers (VTS), and manual tape libraries
  - Non-system-managed tape libraries or traditional tape libraries
- Storage locations that are on-site and off-site
- Storage locations defined as home locations

What Is in a Removable Media Library?

A *removable media library* contains all the tape and optical volumes that are available for immediate use, including the shelves where they reside. A removable media library usually includes other libraries: *system-managed libraries* such as *automated* or *manual tape libraries*; and *non-system-managed libraries*, containing the volumes, shelves, and drives not in an automated or a manual tape library.

In the removable media library, you store your volumes in shelves, where each volume occupies a single *shelf location*. This shelf location is referred to as a *rack number* in the RMM TSO subcommands and in the DFSMSrmm ISPF dialog. A rack number matches the volume's external label. In DFSMSrmm volume serial numbers are used to identify volumes and to identify the volume label. DFSMSrmm allows you to define a volume using a different serial number than is recorded in the volume label. In this way you can define volumes with duplicate volume serial numbers. DFSMSrmm uses the external volume serial number to assign a rack number when adding a volume, unless you specify otherwise. The format of the volume serial and rack you define must be one to six alphanumeric, national, or special characters.

What Is in a System-Managed Tape Library?

A *system-managed tape library* consists of tape volumes and tape devices that are defined in the tape configuration database. The tape configuration database is an integrated catalog facility user catalog marked as a volume catalog (VOLCAT) containing tape volumes and tape library records. A system-managed tape library can be either automated or manual.

You can have several automated tape libraries or manual tape libraries. You use an installation-defined library name to define each automated tape library or manual tape library to the system. DFSMSrmm treats each system-managed tape library as a separate location or destination. See *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries* for additional information.

Automated Tape Libraries

An *automated tape library* is a device consisting of robotic components, cartridge storage areas (or shelves), tape subsystems, and controlling hardware and software, together with the set of tape volumes that reside in the library and can be mounted on the library tape drives. DFSMSrmm provides support for the automated IBM TotalStorage Enterprise Automated Tape Library (3494) and also the IBM TotalStorage Enterprise Automated Tape Library (3495). The IBM 3494 supports
| 3490E, 3590, and 3592 tape subsystems. The IBM 3495 supports 3490, 3490E, and 3590 tape subsystems. The IBM 3494 can also include the Virtual Tape Server subsystem. DFSMSrmm supports the IBM TotalStorage Peer-to-Peer Virtual Tape Server by ensuring that you cannot use the names of distributed VTS libraries with DFSMSrmm. You must only use the names of consolidated libraries with DFSMSrmm.

DFSMSrmm can automatically replenish the scratch volumes in an automated tape library when the supply of volumes becomes low. See "Replenishing Scratch Volumes in a System-Managed Library" on page 381 for information about how DFSMSrmm reclaims volumes that are eligible for returning to scratch.

DFSMSrmm has a cartridge entry installation exit that you can use to help partition volumes in a single system-managed tape library across multiple systems. Use the installation exit for support for sharing between Multiple Virtual Storage (MVS) and other systems and also for partitioning between MVS™ systems and SMS complexes. When using the cartridge entry installation exit, consider these factors.

- DFSMSrmm supports partitioning when you use the DFSMSrmm parmlib options to identify volumes that you want to partition. Volumes can be identified based on volume naming conventions and as individual volumes defined to DFSMSrmm on MVS.
- You can use the DFSMSrmm partitioning support to partition between multiple MVS systems only when each MVS partition has a separate DFSMSrmm control data set.
- When you want to share the DFSMSrmm control data set across multiple MVS partitions, use installation exits such as CBRUXENT and EDGUX200 to control the partitioning.

Manual Tape Libraries

A *manual tape library* is a set of tape drives and the set of system-managed volumes the operator can mount on those drives. The manual tape library provides more flexibility, enabling you to use various tape volumes in a given manual tape library. This support allows volumes to be associated with manual tape libraries so that only those volumes defined for a specific manual tape library can be mounted on drives in that library.

Unlike the automated tape library, the manual tape library does not use the library manager. With the manual tape library, a human operator responds to mount messages that are generated by the host and displayed on a console. This manual tape library implementation completely replaces the IBM 3495-M10 implementation. IBM no longer supports the 3495-M10.

See “Setting Up DFSMSrmm for the System-Managed Tape Library” on page 106 for implementation details for these scenarios.

What Is in a Non-System-Managed Tape Library?

A *non-system-managed tape library* is all the volumes, shelves, and drives not in an automated tape library or manual tape library. You might know this as the traditional tape library in a data center or as an automated environment that is not system-managed. DFSMSrmm provides complete tape management functions for the volumes and shelves in this traditional tape library.

All tape media and drives supported by z/OS are supported in this environment. Use DFSMSrmm to fully manage all types of tapes in a non-system-managed tape library, including 3420 reels, 3480, 3490, and 3590 cartridge system tapes.

You can also use DFSMSrmm to manage volumes in any automated tape library that has special software including an IBM Tape Library Data server that is managed using Basic Tape Library Support (BTLS).

What Is in a Storage Location?

Storage locations are not part of the removable media library because the volumes in storage locations are not generally available for immediate use. A storage location is comprised of shelf locations that you define to DFSMSrmm. A shelf location in a storage location is identified by a *bin number*. Storage locations are typically used to store removable media that are kept for disaster recovery or vital records. DFSMSrmm manages two types of storage locations: installation-defined storage locations and DFSMSrmm built-in storage locations.

You can define an unlimited number of installation-defined storage locations, using any eight-character name for each storage location. Within the installation-defined storage location, you can define the type or shape of the media in the location. You can also define the bin numbers that DFSMSrmm assigns to the shelf locations in the storage location. You can request DFSMSrmm shelf-management when you want DFSMSrmm to assign a specific shelf location to a volume in the location.

You can also use the DFSMSrmm built-in storage locations which are LOCAL, DISTANT, and REMOTE. Although the names of these locations imply their purpose, they do not mandate their actual location. All volumes can be in the same or separate physical location. For example, an installation could have the LOCAL storage location on-site, as a vault in the computer room, the DISTANT storage location could be a vault in an adjacent building, and the REMOTE storage location could be a secure facility across town or in another state. DFSMSrmm provides shelf-management for storage locations so that storage locations can be managed at the shelf location level.

Although DFSMSrmm automatically shelf-manages built-in storage locations, you must first define the bins you want to use to DFSMSrmm. For bin numbers in built-in storage locations, the numbers are fixed in range, starting at bin number 000001. For installation-defined storage locations, you can use any alphanumeric characters.

How Does DFSMSrmm Manage These Libraries and Locations?

DFSMSrmm records the complete inventory of the removable media library and storage locations in the DFSMSrmm control data set which is a VSAM key-sequenced data set. DFSMSrmm records all changes made to the inventory, such as adding or deleting volumes, and also keeps track of all movement between libraries and storage locations.

DFSMSrmm manages the movement of volumes among all library types and storage locations. This lets you control where a volume, and hence a data set, resides and how long it is retained.

DFSMSrmm helps you manage the movement of your volumes and retention of your data over their full life, from initial use to the time they are retired from service. Among the functions DFSMSrmm performs for you are:
- Automatically initializing and erasing volumes.
- Recording information about volumes and data sets as they are used.
- Expiring volumes based on controls you define.
- Identifying volumes with high error levels that require replacement.

To use all of the DFSMSrmm functions, you specify installation setup options and define retention and movement policies. DFSMSrmm provides you with utilities to implement the policies you define. The following sections describe options, policies, and utilities.

Setting Up Your Installation Options

The DFSMSrmm parmlib member EDGRMMxx includes many options for setting up DFSMSrmm, such as:

- Defining system options, such as the date format for reports and messages, default retention periods, and notification to volume owners when their volumes are ready to be released.
- Preventing a range of tapes from being used on specific systems.

- Specifying if a pool has tape profile processing as provided by RACF, a component of the Security Server for z/OS®.
- Tailoring mount messages with either the volume's shelf location or the pool identifier.
- Defining security classes for data sets and volumes.
- Defining the DFSMSrmm running mode to determine when DFSMSrmm records volume usage and performs tape validation as described in "How Does DFSMSrmm Validate Tape Mounts?" on page 18.
- Defining how DFSMSrmm bypass label processing is performed.
- Defining storage locations to DFSMSrmm.

Your system programmer or storage administrator defines your DFSMSrmm installation options during implementation. For information on the options that can be set in EDGRMMxx, see Chapter 8, "Using the Parmlib Member EDGRMMxx," on page 127.

Defining Retention and Movement Policies

The retention and movement policies you define to DFSMSrmm are known as *vital record specifications*. You use them to specify how long and where you want to keep data sets or volumes. You also use them to define how volumes are to be moved among the libraries DFSMSrmm supports and to define the storage locations for vital records and disaster recovery purposes.

Use vital record specifications to control retention requirements for production data in accordance with the service level agreement and known requirements.

Use the DFSMSrmm parmlib member EDGRMMxx to set expiration and retention defaults for your installation. See *z/OS DFSMSrmm Guide and Reference* for information about using the parmlib OPTION VRSEL operand to select vital record processing. When users are allowed to specify expiration and retention period to override vital record specifications, use the parmlib OPTION MAXRETPD operand to set a maximum retention period for your installation. See Chapter 8, "Using the Parmlib Member EDGRMMxx," on page 127 for information.

You can define policies, or vital record specifications, for data sets and volumes. You can also define name vital record specifications to provide retention information for data sets and volumes that must be moved through multiple locations before they expire.

Defining Home Location and Target Destinations

DFSMSrmm records the starting location for a volume when the volume is initially defined to DFSMSrmm or when volume information is changed. This starting location is known to DFSMSrmm as a *home* location. Home is the location where volumes start from and are returned to when the identified retention and movement actions have been completed. You can use system-managed libraries, storage locations, and SHELF as home locations. A non-system-managed library identified as SHELF can only be used as a home location and not as a target location in a vital record specification. SHELF can be used as a target location when a volume is moved by issuing a command.

You can give any system-managed library or storage location as a target destination for a volume move.

You can also use the DFSMSrmm-reserved location name CURRENT to avoid moving a volume from its current location.

Defining Retention Policies

Use data set names and data set name masks to define retention policies for data sets. Use job names and job name masks to define retention policies to further qualify the criteria for applying retention and movement policies. For data sets, you can request the following types of retention:

Retention by cycles
You can use retention by cycles for generation data groups (GDGs), pseudo-GDG data set names, or data sets with the same name. For non-GDG data sets, DFSMSrmm considers each occurrence of a data set to be a cycle.

Retention by cycles by days
You can use retention by cycles by days when you want DFSMSrmm to manage all copies of a data set created on a single calendar day as a single cycle.

Retention by number of elapsed days
You can retain a data set specifying a number of days since it was created.

Retention for a number of extra days
You can retain a data set for a number of days beyond the date when the data set is no longer retained by the previous vital record specification in a vital record specification chain.

Retention by days since last referenced
You can retain each copy of a data set produced for a set number of days since the data set was read or written.

Retention while data set is cataloged
You can retain any data set as long as it remains cataloged.

Retention to a specific date
You can set a deletion date for a vital record specification. When that date is reached, the vital record specification is deleted. All data sets and volumes that would match the vital record specification become eligible for release processing, or might match a less specific vital record specification that might specify different retention and movement information.

Retention by expiration date
You can retain the data set on a volume as long as the volume expiration date has not yet been reached.

Retention of open data sets
You can specify a separate policy to apply to all data sets that are currently open.

Retention of data sets closed by abend processing
You can specify a separate policy to apply to all data sets that were open at the time of an application or system abend.

You can also use a *vital record specification management value* instead of a data set name to define retention policies. A vital record specification management value assigns management and retention policies to tape data sets and is defined by your installation. You can define data set vital record specifications for vital record specification management values to provide support for special JCL-specified expiration dates. You can use the sample installation exit contained in SYS1.SAMPLIB, EDGUX100, to assign vital record specification management values.

You can use management class names when you are managing data sets. See “Managing Volumes with Special Dates” on page 78 for information on assigning management class names and vital record specification management values using EDGUX100.

Defining Vital Record Specification Chains

You can combine retention types and movements within a vital record specification to manage data sets and volumes by defining vital record specifications chains. A vital record specification chain can be one or more vital record specifications linked together using NEXTVRS and ANDVRS operands. The first vital record specification in the chain is a data set or volume vital record specification. You can link vital record specifications to the first vital record specification to add additional retention types and movement policies for a data set or volume. When combining retention types, you can include the WHILECATALOG operand and the UNTILEXPIRED operand with one other retention type in a single vital record specification. To combine more retention types together so that all must be true concurrently, you link multiple vital record specifications using the ANDVRS operand. To combine different retention types to be implemented consecutively or to move a data set or volume through multiple locations consecutively, you link multiple vital record specifications using the NEXTVRS operand. An exception is that when the EXTRADAYS operand is used as a retention type. EXTRADAYS must be the only retention type at that point in the vital record specification chain. Do not link to or from the vital record specification that specifies the EXTRADAYS operand using the ANDVRS operand.

See *z/OS DFSMSrmm Guide and Reference* for information about chaining vital record specifications.

Figure 1 is an example of creating a vital record specification chain. The data set name vital record specification moves a data set from the installation media library to the REMOTE storage location. The name vital record specification, MOVE2, moves the volume from the REMOTE storage location to the DISTANT storage location.

```
RMM ADDVRS DSNAME(STEGO.SAURUS) COUNT(1855) DAYS LOCATION(REMOTE) -
     STORENUMBER(30) NEXTVRS(MOVE2)
RMM ADDVRS NAME(MOVE2) LOCATION(DISTANT) STORENUMBER(1825)
```

Figure 1. Defining a Vital Record Specification Chain

You can define retention and movement policies for specific volumes and volumes that match a generic volume serial number. If you use a specific volume serial number, DFSMSrmm retains the volume that matches that volume serial number. If you use a generic volume serial number, DFSMSrmm retains a number of volumes matching the generic volume serial number based on the number you specify.

When multiple vital record specifications retain a volume, and each vital record specification contains a different destination, DFSMSrmm decides where to move the volume based on priority number. Lower priority numbers have higher priorities. Table 3 on page 8 shows the movement priority DFSMSrmm uses if you do not assign movement priority numbers. For example, if you define both REMOTE and DISTANT as the destination for the volume, DFSMSrmm selects the REMOTE storage location as the destination. The selection is based on the REMOTE priority number of 100 which has a higher priority selection value than the DISTANT priority number of 200.

Table 3. DFSMSrmm Movement Priority

Priority Number	Location Name or Location Type
100	REMOTE DFSMSrmm built-in storage location name
200	DISTANT DFSMSrmm built-in storage location name
300	LOCAL DFSMSrmm built-in storage location
2000	Installation defined storage locations
4800	AUTO automated tape libraries
4900	MANUAL manual tape libraries
5000	SHELF location name

Running DFSMSrmm Utilities

DFSMSrmm provides utilities to manage your inventory, create reports, maintain the DFSMSrmm control data set, and erase and initialize volumes.

Use the EDGHSKP utility to run inventory management activities.

- Vital record processing determines which data sets to retain and what volume moves are required based on the retention and movement policies you define to DFSMSrmm.

  DFSMSrmm supports trial run and production run vital record processing. Trial run vital record processing does not change data set and volume information in the DFSMSrmm control data set. Use trial run vital record processing to analyze the effect that your movement and retention policies will have. Based on your analysis, you can then determine if you need to change vital record specifications before performing production run vital record processing. This is helpful when you are defining your initial set of policies and when you need to make changes as you gain more experience with DFSMSrmm.
- Expiration processing identifies volumes ready to be released and returned to scratch.
- Storage location management processing assigns shelf locations to volumes being moved to storage locations.
- DFSMSrmm control data set functions back up the control data set and the journal, reset the journal data set when the control data set and journal are backed up, and create an extract data set.

Use the EDGAUD and EDGRPTD utilities and the EDGRRPTE exec to get information about your removable media library and storage locations. You can also get security trail information about volumes and data sets defined to DFSMSrmm and audit trail information about volumes, shelf assignments, and user activity.

Use the EDGUTIL utility to create, update, mend, and verify the control data set.

Use the EDGBKUP utility to back up and recover the control data set and journal.

Use DFSMSrmm backup utilities instead of other backup utilities, such as access method services EXPORT, because DFSMSrmm provides the necessary serialization and forward recovery functions using the journal data sets. DFSMSrmm backup utilities check whether the control data set is in use and tell the DFSMSrmm subsystem that backup or recovery is in process.

Use the EDGINERS utility to erase and initialize tape volumes either automatically or manually. You can use EDGINERS instead of the DFSMSdfp™ utility IEHINITT.

See "Replacing IEHINITT with EDGINERS" on page 352 for details on the differences between the utilities and for information on controlling the use of IEHINITT.

You can schedule these utilities to run at the time and frequency that are best for your installation. Use a scheduling system such as IBM Tivoli® Workload Scheduler for z/OS for this purpose.

What Resources Does DFSMSrmm Manage?

The following sections describe how DFSMSrmm helps you manage shelf locations, volumes, data sets, information about volume owners, and software products in all libraries and storage locations.

Shelf Locations

DFSMSrmm automatically:

- Assigns a rack number to a volume that matches the volume serial number you specify except when:
  - You already specified a rack number or pool ID when you first defined a volume to DFSMSrmm.
  - There is no rack number that matches the volume.
  - The volume is a logical volume.
- Assigns bin numbers for the shelf locations in shelf-managed storage locations
- Provides a specific rack number or pool information in the fetch/mount message to help the operator respond to mount requests

You can specify that DFSMSrmm make bins available for reuse when a volume move has started or make bins available only when a volume move has been confirmed. You can request that DFSMSrmm move volumes by specific location and request that DFSMSrmm move the volumes to bins in sequential order, beginning with the lowest volume serial number and the lowest bin number. You can also request that DFSMSrmm reassign the volumes to bins during storage location management processing.

DFSMSrmm also helps you delete rack or bin numbers for obsolete, empty shelf locations. You can create lists containing information about the shelf locations in your removable media library and in storage locations.

You can have more control over where volumes reside and how they are managed by defining your removable media into pools. A pool is a group of shelf locations defined by a common prefix or a group of volumes that require DFSMSrmm pool management functions. For example, based on your pool definitions, DFSMSrmm can ensure that RACF profiles exist for all volumes in a pool. You could also request that all volumes in a pool that are protected with an expiration date are processed automatically. See "Organizing the Library by Pools" on page 63 for information about defining your pools in parmlib member EDGRMMxx. If you do not define any pools, DFSMSrmm considers all volumes part of one default pool.

In DFSMSrmm, there are two categories of pools:

Rack pool
A *rack pool* is shelf space that can be assigned to hold any volumes. Although you can add scratch volumes to these pools, you cannot normally use these volumes to satisfy non-specific mount requests.

Each rack pool can contain private volumes and scratch volumes:

- Private volumes
  - Are not generally available for use because the data on them is not yet expired
  - Can have a master status or user status

    A master status indicates that the volume contains data that can only be overwritten based on criteria set by the EDGRMMxx MASTEROVERWRITE operand.

    A user status indicates that the volume contains data that can be overwritten even when a data set name does not match.
- Scratch volumes
  - Are available for use because they are either unused volumes or they contain expired data
  - Are used to satisfy scratch requests.

    With scratch volumes in rack pools, you can provide a user or group-based pool of volumes that is selected using the RMM GETVOLUME subcommand or with the EDGUX100 installation exit. Scratch volumes in a rack pool cannot be used where DFSMSrmm system-based pooling is in use.

Scratch pool

A *scratch pool* is shelf space assigned to hold volumes for use with the DFSMSrmm system-based scratch pooling or exit-based pooling. The volumes assigned to this shelf space can be used to satisfy scratch requests as long as the volumes are in scratch status. Once the volume has been written to, it becomes a volume with master status, until the volume is returned to scratch status.

Pools within system-managed libraries are based on the external volume serial number, which is also used as the rack number. Most volumes in a system-managed library either belong to a general scratch pool or they can be separated into multiple scratch pools, one for each media type. For example, volumes can be divided into one pool for cartridge system tape and one pool for enhanced capacity cartridge system tapes. If volumes from other data centers are entered into a system-managed library for processing, a rack pool can be defined to allow you to take advantage of DFSMSrmm pool management functions.

Pools in non-system-managed libraries are logical sections of shelf space. The shelf space is divided based on rack number, logical groups of volumes based on volume prefix, or logical groups of volumes with the same storage group name. You can control shelf location assignment by specifying a pool ID when you define a volume to DFSMSrmm. DFSMSrmm automatically assigns the volume to the next available rack number in that pool. If you do not specify a shelf location or pool ID for a volume, DFSMSrmm attempts to assign the rack number that matches the volume serial number.

Volumes

DFSMSrmm provides support for the following volumes.

- DFSMSrmm allows you to use all types of physical volumes, tracking their use and the data they contain, and providing vaulting and retention services according to policies assigned to the data sets on the volumes. DFSMSrmm also allows you to use duplicate volumes. These are volumes that are defined to

DFSMSrmm with a unique external volume serial number and a VOL1 label that might duplicate another volume but that does not match its own external volume serial number.

- Logical volumes reside in a VTS or on exported stacked volumes. Applications can access the data on these volumes only when they reside in a VTS. The data in a VTS is accessed in its tape volume cache or after the data has been copied to a physical volume through the use of special utilities. DFSMSrmm tracks the use of logical volumes and the data that resides on the volumes. Retention services are provided according to policies assigned to the data sets on the volumes. DFSMSrmm supports exporting of logical volumes on stacked volumes and provides vaulting services for these stacked volumes based on the policies assigned to the data sets on the contained logical volumes. Logical volumes can be removed from a VTS using export processing described in the "Using DFSMSrmm with the IBM TotalStorage Peer-to-Peer Virtual Tape Server (PtP VTS)" on page 93.
- Stacked volumes have a one-to-one association with physical tape media and are used in a VTS to store logical volumes. Stacked volumes are not used by MVS applications but by the VTS and its associated utilities. Stacked volumes can be removed from a VTS to allow transportation of logical volumes to a vault or to another VTS. DFSMSrmm can be used to track stacked volumes and automatically records the logical volumes contained on stacked volumes during export processing when stacked volume support is enabled. DFSMSrmm provides support for importing logical volumes on stacked volumes either into the VTS from which it was exported or into a different VTS.

DFSMSrmm automatically validates volumes to ensure that only valid scratch volumes are mounted for non-specific mount requests and that the right volume is mounted for a specific mount request. This eliminates unintentionally overwriting a valid master volume or a volume retained for disaster recovery or vital record management.

When a data set on a volume is opened and closed, DFSMSrmm automatically performs the following functions:

- Changes the volume status from scratch to master for non-specific mount requests at open time
- Sets an expiration date for the volume and ensures that the maximum expiration date is not exceeded at open time
- Records information about data sets on the volume (the data set name is recorded at open time; all other information is recorded at close time)
- Counts the number of times a volume is used since the volume was last in scratch status at open time
- Counts the number of temporary and permanent errors encountered at close time only
- Sets a security classification based on the data sets that reside on the volume at open time
- Prevents reading of data on a volume in scratch status when DFSMSrmm is running in protect mode at open time

DFSMSrmm automatically controls the movement of volumes within the removable media library and the built-in or installation-defined storage locations, based on criteria you specify in vital record specifications.

When volumes no longer reside in the removable media library or storage locations, you can define loan locations where these volumes can be found. DFSMSrmm does not, however, manage the movement of volumes residing in loan locations.

DFSMSrmm automatically releases volumes that have reached their expiration date. Any non-scratch volume defined to DFSMSrmm has an expiration date indicating when the volume is to be considered for release. The volume expiration date can be:

- An expiration date or retention period, given in the JCL by the user when writing a data set to the volume.
- The date specified by the user when a scratch volume is requested using the RMM GETVOLUME subcommand or when information about the volume is manually added or changed.
- A default retention period for data sets set by your installation with the RETPD option in parmlib member EDGRMMxx in SYS1.PARMLIB.
- The expiration date that is set using the DFSMSrmm EDGUX100 installation exit. If the exit sets a zero date, the installation default retention period is used.

Your installation can also set a maximum retention period in parmlib member EDGRMMxx. An expiration date or retention period set for a volume cannot exceed this value.

You can retain a volume beyond its expiration date by using a vital record specification to define a retention policy. You can also manually override an expiration date and release a volume before the expiration date is reached.

When a volume is eligible for release, DFSMSrmm can perform *release actions* for the volume based on information you provide. Examples of release actions include:

- Returning a volume to scratch status
- Initializing a volume
- Erasing a volume
- Notifying a volume's owner of the volume's release
- Returning a volume to its owner and deleting the volume record in the DFSMSrmm control data set

See *z/OS DFSMSrmm Guide and Reference* for information about setting release actions and how processing is performed.

DFSMSrmm tracks the number of I/O errors recorded for a volume. For permanent errors, DFSMSrmm automatically marks the volumes for replacement before returning them to scratch. This occurs during expiration processing.

You can get additional information to help you analyze volumes with temporary errors from these tools and services:

- DFSMSrmm records volume use and temporary I/O errors. You can obtain this information through the extract data provided by EDGHSKP.
- The Environmental Record Editing and Printing Program (EREP) uses LOGREC records to identify tapes with unacceptably high error conditions.
- Service Director™ identifies volumes with high error levels. See your IBM service representative for information about using this tool.
- System management facilities (SMF) records, produced by DFSMSdfp, track information on volume use, such as how much data is written to a volume. You can use this information to analyze volumes with errors.

You can then use this information to identify volumes you want replaced or managed by DFSMSrmm. Use the RMM CHANGEVOLUME RELEASEACTION(REPLACE) subcommand or the dialog change volume function when you want DFSMSrmm to manage the replacement.

DFSMSrmm provides support for using RACF standard tape volume security protection. You can use any combination of RACF TAPEVOL and TAPEDSN options.

DFSMSrmm allows you to use volumes with duplicate volume serial numbers as well as volumes undefined to DFSMSrmm. You can request DFSMSrmm to ignore a volume so it can be used, even if another volume with the same volume serial number is already defined in the DFSMSrmm control data set. If DFSMSrmm ignores a volume, it does not track volume use. If the volume is not defined to DFSMSrmm, DFSMSrmm cannot provide any management functions for the volume.

DFSMSrmm Tape Label Support

DFSMSrmm supports the following tape label types:

- IBM standard labels (SL)
- ISO/ANSI labels (AL)
- Both IBM standard and user header or trailer labels (SUL)
- Both ISO/ANSI and user header or trailer labels (AUL)
- No labels (NL)

You can explicitly use AL, SL, and NL label types when you define volumes to DFSMSrmm. Although a tape is defined to DFSMSrmm as SL or AL, a user can still process it as SUL or AUL. A user could ask for a non-specific volume by specifying LABEL=(,SUL) and create an SL volume or SUL volume without affecting DFSMSrmm. DFSMSrmm records the value specified in the JCL.

DFSMSrmm provides support for ISO/ANSI Version 3 and ISO/ANSI Version 4 labels. You can provide the current ISO/ANSI label version of a volume that you define to DFSMSrmm. Then you can use the DFSMSrmm EDGINERS utility to initialize the tape with either ISO/ANSI Version 3 or ISO/ANSI Version 4 labels.

DFSMSrmm supports no label (NL) for private volumes and for scratch tapes in a system-managed automated tape library and private volumes in a non-system-managed tape library. If you use the IFG019VM volume mount installation exit, DFSMSrmm supports no label (NL) for scratch tapes in a non-system-managed tape library. All other scratch tapes must be standard label tapes. DFSMSrmm allows the creation of no label tapes from standard label scratch volumes by changing the volume to master status and setting the initialize release action so that the tape is relabeled as a standard label tape when returned to scratch. DFSMSrmm also overrides the logical volume serial number generated by OPEN for NL output to scratch tapes, so that the correct volume serial number is used for cataloging data sets. Nonstandard labels (NSL) can be used but only if the volume is not defined to DFSMSrmm. “How Does DFSMSrmm Validate Tape Mounts?” on page 18 describes how DFSMSrmm validates volume usage and does not allow the processing of certain label types.

DFSMSrmm also supports the use of bypass label processing (BLP) for:

- Volumes that are either not defined to DFSMSrmm or volumes that DFSMSrmm should ignore.

- Non-scratch (user and master status) volumes that are used for input processing, and only those user status volumes that are used for output processing. You can select this system option in the EDGRMMxx parmlib member using the OPTION BLP(RMM) command.
- Non-scratch (user and master status) volumes that are used for input processing, or user, master and scratch tapes that are used for output processing. You can select this option in the EDGRMMxx parmlib member using the OPTION BLP(NORMM) command.

  When BLP is used with scratch tapes, DFSMSrmm changes the volume to master status. DFSMSrmm also sets the initialize volume release action to ensure that the volume has a valid standard label before it returns to scratch status. When a volume is written with BLP, DFSMSrmm can no longer perform the 44-character data set name check for the files on the volume.

  For BLP output to a non-specific volume, DFSMSrmm does not check the file sequence number. For example, LABEL=(2,BLP) can be used. For non-BLP requests, DFSMSrmm restricts the use to LABEL=(1,label_type).

  The data set information for files processed with BLP is only updated for output to first file on a volume. All other types of BLP requests change only the volume information such as the date last read and the date last written. Also for BLP output requests, the data set information is only updated in the DFSMSrmm control data set when the data set is closed.

Data Sets

DFSMSrmm automatically records information about a data set when the data set is opened. DFSMSrmm can automatically record information when you request data set recording and one of the following conditions is true:

- The data set is the first file on the volume.
- You already processed all files preceding it on the volume.
- You have previously defined the data sets on the volume to DFSMSrmm.

Data class, management class, storage class, and storage group are included in the information that DFSMSrmm records for system-managed data sets.

You can manually add information about a data set if the volume on which the data set resides is already defined to DFSMSrmm and the volume has not been already processed as a result of open processing and close processing. There are some limitations on the information that you can change if it was originally recorded by DFSMSrmm during open, close, and end-of-volume processing.

DFSMSrmm supports generic data set names as filter criteria for searching the control data set which makes it easier to create lists of resources.

Year 2000 Support

DFSMSrmm provides support for dates beyond the 20th century by ensuring that DFSMSrmm stores and displays all dates using a 4-digit year. DFSMSrmm also allows you to specify dates using the 4-digit year. DFSMSrmm provides the same support for dates as DFSMSdfp™.

Software Products

You can also define software products to DFSMSrmm and associate volumes with the products.

Owner Information

DFSMSrmm can help you keep track of volume owners. It provides functions to electronically notify owners when their volumes are being considered for release. DFSMSrmm automatically records owner IDs as volumes are used. If you want DFSMSrmm to use owner information to automatically notify owners, you must manually define the owner's electronic address to DFSMSrmm. Additionally, the notify action must have been requested for the volume.

DFSMSrmm ensures that there is an owner ID for each volume. If a job is started and does not have a known RACF user ID, DFSMSrmm uses the job name as the owner ID.

DFSMSrmm also provides for notification to product owners when new volumes are added for a product.

DFSMSrmm keeps track of volume ownership. To delete a record for a user who still owns volumes, DFSMSrmm optionally allows you to transfer ownership of those volumes before deleting the owner record.

How Does DFSMSrmm Help You Create Reports?

You can obtain information and create reports using the following methods:
- DFSMSrmm Report Generator
- DFSMSrmm ISPF dialog or RMM TSO commands
- EDGAUD and EDGRPTD report utilities
- DFSMSrmm EDGRRPTE exec shipped in SAMPLIB
- The DFSORT™ ICETOOL utility
- The DFSMSrmm application programming interface

Using DFSMSrmm Report Generator

The DFSMSrmm Report Generator is an ISPF application that you use to create reports to show the status of the resources that DFSMSrmm manages for you. These resources include volumes, data sets, racks, owners, and the retention and movement policies that are established for your installation. Refer to *z/OS DFSMSrmm Reporting* for detailed information.

Using DFSMSrmm ISPF Dialog and RMM TSO Subcommands

You can search on-line using either the DFSMSrmm ISPF dialog or RMM TSO subcommands to create lists of resources and display information recorded in the DFSMSrmm control data set. Here are some examples:
- Operators can create lists of scratch volumes to be pulled for use.
- Tape librarians and system programmers can create lists of software products and the volumes on which they reside.
- General users can create lists of volumes they own, such as the example in Figure 2 on page 16:

```
Volume Owner    Rack   Assigned   Expiration Location Dsets St Act   Dest.
                       date       date
 ------ -------- ------ ---------- ---------- -------- ----- -- ----- --------
 VOL600 AMYW01   RAC500 06/11/2000 11/11/2000 SHELF    0     UR SI
 VOL601 AMYW01   RAC501 06/11/2000 11/11/2000 SHELF    0     UR SI
 VOL603 AMYW01   RAC502 06/11/2000 11/11/2000 SHELF    0     UR SI
 EDG3011I 3 ENTRIES LISTED
```

Figure 2. Example of a List of Volumes Owned by a Single User

With DFSMSrmm, you can use the RMM TSO SEARCH subcommands with the CLIST operand to create a data set of executable subcommands. For example, you can create subcommands to confirm volume movement for volumes identified during a SEARCHVOLUME request.

Using the EDGAUD and EDGRPTD Report Utilities

You can create several types of reports using two DFSMSrmm report utilities. Use EDGRPTD to create movement and inventory reports and EDGAUD to create security and audit reports. EDGRPTD uses the DFSMSrmm extract data set as input. EDGAUD uses SMF records as input.

You can use the reports for the following purposes:

- Identifying volumes that should be moved between the removable media library and storage locations.
- Determining your volume inventory in the removable media library and storage locations.
- Identifying volumes that are in transit or that should be marked as moved.
- Identifying all accesses to volumes and changes to information recorded in the DFSMSrmm control data set.
- Separating volumes that are waiting to return to scratch from those that are private or have other release actions pending.
- Producing new scratch volume reports or scratch volume inventory reports.

Using the DFSMSrmm EDGRRPTE Exec

Use the DFSMSrmm-supplied EDGJRPT JCL to run the EDGRRPTE exec to produce reports using the extract data set as input. See *z/OS DFSMSrmm Reporting* for more information.

Using the DFSORT ICETOOL Utility

You can use DFSORT™ or a similar program to generate a formatted report using the information in the extract data set created by the EDGHSKP utility. For example, you could produce an extract data set listing all volumes to be used on VM with information about volume owners. Then use the DFSORT ICETOOL utility to sort the information by volume and produce a report, complete with title and header information.

Using the DFSMSrmm Application Programming Interface

You can use the DFSMSrmm application programming interface to obtain information about resources defined to DFSMSrmm. See the *z/OS DFSMSrmm Application Programming Interface* publication for information about how to use the DFSMSrmm application programming interface.

How Does DFSMSrmm Authorization and Security Work?

You can choose the authorization levels of users for all DFSMSrmm functions. DFSMSrmm uses MVS System Authorization Facility (SAF) for its authorization checking. You define DFSMSrmm resources to RACF for use during authorization checking. DFSMSrmm can create volume profiles, change them, and delete them on registration, expiration, or release of volumes. DFSMSrmm provides an access list you can use to set the access list in RACF. You can use the DFSMSrmm access list for authorization checking on non-RACF systems. Use the RMM LISTVOLUME subcommand or the DFSMSrmm ISPF dialog to display the DFSMSrmm access list. You can also view the access list in the volume records in the report extract data set.

DFSMSrmm provides automatic security classification through installation-specified criteria based on data set names. DFSMSrmm security includes these elements:

- An audit trail of access and change of status through SMF. This audit trail produces information about RACF user IDs, groups, and job names.
- Required operator confirmation prior to using certain volumes.
- Erasure of data when a volume is released prior to the volume returning to scratch status.

DFSMSrmm provides the following ways of optionally keeping an audit trail for volumes defined to it:

- Control data set information
- SMF audit records
- RACF audit information

For more information about DFSMSrmm authorization and security, see the Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171.

What Tape Usage Does DFSMSrmm Support?

DFSMSrmm supports the following tape usage:

- Using BLP to read any private volume.
- Using BLP to write to any volume.
- Using nonstandard label volumes that are not defined to DFSMSrmm.
- Using both AL and SL scratch tapes.
- Using no label (NL) processing for any private volume.
- Performing no label (NL) output to any system-managed scratch volume or standard label scratch volume.
- Using no label (NL) for private volumes and for scratch tapes in a system-managed automated tape library and private volumes in a non-system-managed tape library.
- Using no label (NL) also for scratch tapes in a non-system-managed tape library if you use the IFG019VM volume mount installation exit.
- Overwriting a volume that is in master status. For volumes in master status, DFSMSrmm allows the overwriting of data based on criteria set by the EDGRMMxx MASTEROVERWRITE operand.
- Overwriting a volume that is in user status no matter what the data set name is.
- Using volumes not defined to DFSMSrmm.
- Using duplicate volumes.

- Automatically labeling scratch tapes in an automated tape library.
- Reusing 36-track recorded scratch tapes on 18-track drives and 256-track recorded scratch tapes on 128-track drives.
- Using multifile and multivolume data sets.
- Recording and validating only the first file on the volume.
- Creating checkpoint data sets on scratch volumes in an automated system-managed tape library.
- Creating non-checkpoint data sets on scratch volumes in an automated system-managed library, where the scratch volumes were previously a checkpoint secure volume.
- Exploiting high-speed cartridge tape positioning when an application does not exploit it.

How Does DFSMSrmm Validate Tape Mounts?

DFSMSrmm performs validation for all data sets on a volume that have been recorded by DFSMSrmm when a data set is opened. DFSMSrmm validates tape volumes as follows:

- For specific tape requests, DFSMSrmm checks that the correct private volume is mounted.
- For non-specific tape requests, DFSMSrmm checks that a scratch volume is mounted and that the volume is from an acceptable rack, scratch pool, or storage group.
- For volumes where DFSMSrmm has recorded the first file creation at open time, DFSMSrmm checks that the last 17 characters of the data set name from the tape volume HDR1 label match the first file data set name known to DFSMSrmm.
- For specific requests to overwrite data on a master volume, DFSMSrmm allows the overwriting of data based on criteria set by the EDGRMM*xx* OPTION MASTEROVERWRITE operand. DFSMSrmm checks that the data set name used matches the data set name that DFSMSrmm has recorded. For generation data group data (GDG) sets, DFSMSrmm removes the GDG suffix before checking the data set name.
- At open time for the file being referenced, DFSMSrmm checks that the data set name used matches the one DFSMSrmm has recorded. If only the first file on a volume is being recorded, DFSMSrmm only validates the first file on the volume.

  If DFSMShsm is reading a tape volume, only the last 17 characters of the data set name need to match the data set information in the tape header label.
- For scratch tapes mounted in a fully functional automated tape library, that have incorrect or missing VOL1 labels, DFSMSrmm checks the external and internal volume serial numbers. DFSMSrmm ensures that both the external volume serial number and internal volume serial number, if one exists, are either defined as scratch to DFSMSrmm or are not defined to DFSMSrmm.
- DFSMSrmm ensures that the mounted volume has the correct WWID. DFSMSrmm obtains the WWID for a WORM tape from the tape drive when the volume is mounted and used or when you define the WORM tape to DFSMSrmm using the RMM TSO subcommands.
- When DFSMSrmm has a Write Mount Count for a WORM media, DFSMSrmm ensures that the mounted volume write mount count matches the value recorded in the DFSMSrmm control database.

DFSMSrmm performs tape mount validation based on the DFSMSrmm running mode set in the parmlib member EDGRMMxx with the OPTION command and

OPMODE operand as shown in Table 4.

Table 4. How the DFSMSrmm Running Mode Affects Tape Mount Validation

DFSMSrmm Running Mode	Tape Validation
Manual	DFSMSrmm does not validate volume usage.
Record-only	DFSMSrmm does not validate volume usage.
Warning	DFSMSrmm validates tape volume usage and issues warnings if errors are encountered. DFSMSrmm does not reject volume usage.
Protect	DFSMSrmm validates tape volume usage and rejects volume usage under certain conditions.

Use warning mode as you perform testing during conversion from another tape management system. When you are running DFSMSrmm in warning mode, DFSMSrmm validates your tape volumes but does not prevent their use. See the "Defining System Options: OPTION" on page 134 for information about setting options in the DFSMSrmm parmlib member.

Why Does DFSMSrmm Reject Tape Volumes?

In addition to validation, there are a number of reasons why a tape can be rejected when you are running DFSMSrmm in protect mode. Volumes can be rejected based on installation-controlled parmlib options and volume definitions or based on DFSMSrmm and system rules.

Rejects Caused by Installation Controls

With DFSMSrmm you can define conditions such as where volumes should reside, ranges of shelf locations that should not be used, and volumes that can only be used on certain systems. If you are running DFSMSrmm in protect mode, the volume is rejected under these conditions:

- You have defined the REJECT option for a range of shelf locations, preventing a volume from being used on a particular system. The REJECT option is in parmlib member EDGRMMxx.
- The volume mounted for a scratch request has not been defined to DFSMSrmm.
- The volume mounted for a scratch request is not in a scratch pool associated with this system or does not match installation-defined requirements.
- The volume mounted for a scratch request is from a scratch pool associated with another system.
- The volume is not to be used on MVS systems.
- The volume is not defined to DFSMSrmm and your REJECT options request that all non-defined volumes are to be rejected.
- A user asks for the use of a volume to be ignored, but is not authorized to do so for that volume.
- A volume in an automated tape library is not defined to DFSMSrmm and cannot be defined to DFSMSrmm for some reason.

Rejects Caused by Validation Failure

DFSMSrmm performs validity checking on volumes when you read or write to them if DFSMSrmm is recording information about the data sets on the volumes. If you are running DFSMSrmm in protect mode the volume is rejected under these conditions:

- The wrong volume is mounted for a specific volume request.
- An attempt is made to use a specific scratch volume. In DFSMSrmm, when you want a specific volume, you must request a specific, non-scratch volume, and when you want a scratch volume, you must request a non-specific mount.
- A private volume (master or user) is mounted in response to a scratch request.
- The data set information for the first file on the volume that DFSMSrmm has recorded during open, close, or end-of-volume processing does not match the information on the volume.
- An attempt is made to overwrite a data set on a master volume and the specified data set name does not exactly match the data set name that DFSMSrmm has recorded. You can control the overwriting of data sets on master volumes using the EDGRMMxx OPTION MASTEROVERWRITE operand. If both the data set being written, and the data set DFSMSrmm has recorded are generation data group data sets, DFSMSrmm ignores the generation data group suffix when comparing the data set names.

  If the volume is part of a multivolume sequence containing multiple data sets, DFSMSrmm uses only the first volume, first file data set name for validation; for all other volumes the sequence of volumes is validated to prevent overwrite.
- An attempt is made to read a data set that DFSMSrmm has not recorded, and the volume information was previously recorded by DFSMSrmm.
- An attempt is made to automatically label a non-scratch volume in an automated tape library.

Rejects Caused by DFSMSrmm Rules

DFSMSrmm checks that volumes are labeled correctly and that the volume status and intended usage is acceptable to DFSMSrmm. If you are running DFSMSrmm in protect mode the volume is rejected under these conditions:

- An attempt is made to read a scratch volume.
- An attempt is made to read a volume obtained using the RMM GETVOLUME subcommand and the volume has not yet been written to. You use the RMM GETVOLUME subcommand to request a scratch volume and assign it to an owner defined to DFSMSrmm.
- Bypass label processing (BLP) is being used to write to a scratch or master volume unless you requested BLP processing through EDGRMMxx in the parmlib.
- An attempt is made to read or write to a volume using nonstandard labels, and the volume is defined to DFSMSrmm.
- An attempt is made to overwrite standard labels on a master volume, and the user is not authorized to do so.
- An attempt is made to write standard labels on a master volume that has no labels, and the user is not authorized to do so.
- An attempt is made to overwrite standard labels on a scratch volume by a user that is not authorized to do so.
- A scratch volume is requested for a nonstandard label request. In DFSMSrmm, scratch volumes must have standard labels.
- A volume is in an automated tape library and is defined to DFSMSrmm, but it is defined as not having a standard label or has different internal and external volume labels.
- An attempt is made to read or write to a volume that is waiting to be initialized.
- An attempt is made to read or write to a volume that is pending release.

- An attempt is made to write to a data set on the scratch volume other than the first one.
- An attempt is made to write to a data set that was specified with a sequence number that is not the next in the sequence from the last file DFSMSrmm has recorded. This applies only if DFSMSrmm is recording information about all the data sets on the subject volume.

Who Can Use DFSMSrmm and How?

The following sections describe DFSMSrmm users and the tasks they can perform.

General User

General users need only limited access to DFSMSrmm functions. They might want to manage volumes they own and request information about resources defined to DFSMSrmm.

General users can use DFSMSrmm to perform these tasks:

- Manually request a scratch volume
- Change information about an owned volume or data set
- Update information about your owner ID
- Manually release an owned volume
- Create lists of resources and display information about most resources defined to DFSMSrmm

Tape Librarian

Tape librarians can use DFSMSrmm to perform these tasks:

- Define new volumes
- Add shelf locations to the removable media library and to storage locations
- Add, change, and delete information about resources defined to DFSMSrmm
- Manually release any volume
- Confirm volume movements and actions
- Create lists of resources and display information about any resource defined to DFSMSrmm

Storage Administrator

Storage administrators can use DFSMSrmm to perform these tasks:

- Define retention and storage policies for data sets and volumes
- Change information about any volume they own, their owner ID, and any vital record specification
- Manually request a scratch volume
- Manually release a volume they own
- Delete a vital record specification
- Create lists of resources and display information about resources defined to DFSMSrmm

System Programmer

System programmers can use the DFSMSrmm support menu to perform these tasks:

- Display parmlib options and control data set control information

- Add, change, and delete information about any volume owner
- Manually request a volume and manually release a volume they own
- Create a list of software products and display information about any resource defined to DFSMSrmm

Operator

Operators can use DFSMSrmm to perform these tasks:

- Fetch and mount tapes from specific pools or shelf locations, as specified in mount messages
- Manually erase and initialize tapes
- Manually request a scratch volume
- Manually release a volume they own
- Create lists of scratch tapes available for use

See *z/OS DFSMSrmm Guide and Reference* for a complete description of operator procedures.

Using DFSMSrmm

You define RACF profiles to establish the authorization scheme for using DFSMSrmm functions. The basic authorization scheme recognizes there are different types of users and each user type will request common DFSMSrmm functions. See Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171 for information on how authorization is set up for each user type.

In DFSMSrmm, you can use either the DFSMSrmm ISPF dialog or the set of TSO subcommands to request DFSMSrmm functions. The RACF profiles control whether or not DFSMSrmm responds to requests for functions. If you request a function you are not authorized to use, your request will fail. For descriptions of the TSO subcommands available within DFSMSrmm, see *z/OS DFSMSrmm Guide and Reference*.

DFSMSrmm offers menus in the DFSMSrmm ISPF dialog that are tailored specifically to a user group’s needs and level of access authorization. For example, only tape librarians are authorized to add software products to DFSMSrmm, so only the DFSMSrmm Librarian Menu includes an option to add software products. DFSMSrmm provides a specific menu for general users, storage administrators, tape librarians, and system programmers. It does not provide a menu for operators.

Chapter 2. Implementing DFSMSrmm

This topic describes tasks you need to perform to implement DFSMSrmm. This topic also includes some optional tasks that might not need to be performed for your installation. Review all the steps in this topic as well as information in the *z/OS Migration* document before you begin implementing DFSMSrmm. If you have multiple MVS images or a RMMplex, you must repeat several of these steps for each MVS image. We provide recommendations for steps that should be repeated for each image. The document describes steps that are required before you install DFSMSrmm.

To access IBM Redbooks on topics such as converting to DFSMSrmm from another tape management product, visit www.redbooks.ibm.com.

Here are the steps that you can follow to implement DFSMSrmm.

Step 1: Preparing to Implement DFSMSrmm

Before using this document, you should have installed DFSMSrmm, along with the other DFSMS components, using SMP/E. Refer to the *z/OS Program Directory* you received with the product tape for complete installation instructions.

If you have multiple MVS images or systems on which you want DFSMSrmm to manage tape processing, there are some basic decisions you must make about how to implement DFSMSrmm. If the systems have access to shared DASD, you can set up one DFSMSrmm control data set and share it among your systems.

Systems which are in a sysplex are best managed using a single control data set. When you have multiple sysplexes and have shared DASD between them, you can choose to have a single control data set to be shared or one control data set per sysplex. If you do not have shared DASD on which to place a control data set for all systems to share, consider using DFSMSrmm client server support for z/OS to enable a single control data set to be used. See Chapter 3, “Setting Up DFSMSrmm Client and Server Systems,” on page 57 for additional information.

Recommendation: After you have installed DFSMSrmm using SMP/E, IPL your system without performing any DFSMSrmm implementation tasks and have DFSMSrmm take no part in removable media management. This is especially helpful if you are running another tape management product in production.

See Appendix D, “Evaluating Removable Media Management Needs,” on page 431 for a checklist to help you plan for implementing DFSMSrmm.

Step 2: Running the Installation Verification Procedure (Optional)

If this is the first time you are implementing DFSMSrmm, use the supplied IVP to verify that DFSMSrmm has installed correctly. You can run the IVP at any time, for example, after installing maintenance on your system.

See Appendix A, “DFSMSrmm Installation Verification Procedures,” on page 403 for the IVP procedures.

Step 3: Updating JES3 (Optional)

DFSMSrmm Samples Provided in SAMPLIB

- EDG3IIP1 Sample to Update IATIIP1
- EDG3LVVR Sample to Update IATLVVR
- EDG3UX29 Sample to Update IATUX29
- EDG3UX62 Sample to Update IATUX62
- EDG3UX71 Sample to Update IATUX71

If you use DFSMSrmm and have JES3-managed tape devices that are not in an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495), continue with this step. Otherwise, skip this step because allocation and DFSMSdfp handle all mount and volume verification.

DFSMSrmm provides SMP/E USERMODs that you can apply to the standard supplied JES3 user exits and other JES3 modules. The USERMODs can be used to prevent JES3 from validating certain volume mounts, to update JES3 fetch and mount messages, and to enable the use of no label tape volumes with JES3. If you have installed DFSMSrmm on a JES3 system, you might want to apply the JES3 USERMODs as described in Chapter 13, “Running DFSMSrmm with JES3,” on page 271. You must consider how applying a USERMOD might affect any locally-developed user exits.

Step 4: Updating Installation Exits

DFSMSrmm Samples Provided in SAMPLIB

| • EDGCVRSX Sample to Use the Installation Exit EDGUX100
- EDGUX100 Sample to Use the Installation Exit EDGUX100
- EDGUX200 Sample to Use the Installation Exit EDGUX200

DFSMSrmm Samples Provided in AEDGSRC1 or SMPSTS

- CBRUXCUA Programming Interface to EDGLCSUX
- CBRUXEJC Programming Interface to EDGLCSUX
- CBRUXENT Programming Interface to EDGLCSUX
- CBRUXVNL Programming Interface to EDGLCSUX
- IGXMSGEX Programming Interface to EDGMSGEX

DFSMSrmm provides two sample installation exits, EDGUX100 and EDGUX200. You must install EDGUX100 and EDGUX200 if you want to use them. Chapter 11, “Using DFSMSrmm Installation Exits,” on page 221 describes the functions that the DFSMSrmm installation exits provide and how to use the exits. The installation exits that DFSMSrmm provides for IGXMSGEX and the OAM exits are implemented when DFSMS/MVS is installed. The installation exits that are installed for IGXMSGEX or the OAM exits replace any exits you had previously installed.

Recommendation: Convert your code to run when called by the DFSMSrmm exits and modify the supplied exits to call your code. You might need to modify your DFSMSrmm installation exits to provide the functions available in the exits you are currently using as well as provide functions required by DFSMSrmm.

Related Reading:

- “Processing Fetch and Mount Messages: EDGMSGEX” on page 215 for details about the DFSMSrmm programming interface that you use with IGXMSGEX.
- “Managing System-Managed Tape Library Volumes: EDGLCSUX” on page 202 for details about the DFSMSrmm programming interface that you use with the OAM installation exits.

Step 5: Updating SYS1.PARMLIB Members

Update the SYS1.PARMLIB members IEFSSNxx and IKJTSOxx. Optionally, you can update IFGPSEDI, SMFPRMxx, and GRSRNLxx. You should also ensure that IFAPRD00 is updated correctly during installation of DFSMS.

Perform this step once for each MVS image.

Updating IEFSSNxx

There are two changes to make in the IEFSSNxx member in SYS1.PARMLIB:

- Define a DFSMSrmm subsystem name to MVS
- Add the name of the subsystem interface initialization program, EDGSSSI, to fully enable DFSMSrmm

Recommendation: Implement the changes in two steps. Define the DFSMSrmm system at this time as described in “Defining DFSMSrmm to MVS.” Then you can add the name of the subsystem interface later as described in “Step 21: Initializing the DFSMSrmm Subsystem and Tape Recording” on page 54. If you make only this first change at this time, you can choose whether to start the DFSMSrmm procedure and choose a time to best suit your particular situation. If you implement both changes at once, DFSMSrmm rejects all tape mounts until the subsystem procedure is active or you use the EDGRESET utility to disable the interface.

Defining DFSMSrmm to MVS

Add an entry to IEFSSNxx to define the DFSMSrmm subsystem to MVS. Specify the DFSMSrmm subsystem name shown in Figure 3. Figure 3 shows where to place the subsystem name. Place the DFSMSrmm entry after the JES2 or JES3 entry and before the NetView® entry to ensure that NetView automation receives the write-to-operator messages once DFSMSrmm has updated them. Additionally, if you have any software identified as a subsystem that is dependent on open/close/end-of-volume processing, place the names after the DFSMSrmm entry. DFSMSrmm has no other dependencies on the sequence of subsystem names.

```
SUBSYS SUBNAME(JES2)             /* JES2 PRIMARY SUBSYSTEM START   */
  PRIMARY(YES)  START(YES)
SUBSYS SUBNAME(DFRM)             /* Name of the DFSMSrmm subsystem */
SUBSYS SUBNAME(AOPA)             /* Netview                        */
```

Figure 3. Defining DFSMSrmm to MVS through IEFSSNxx With Subsystem Inactive

where:

DFRM	Specifies the subsystem name. You can use any unique subsystem name, one to four characters long.

Although the subsystem name can be one-to-four characters long, the subsystem name must be four characters and must match the first four characters of the procedure name under these conditions:

- You have not added EDGSSSI, as described in “Step 21: Initializing the DFSMSrmm Subsystem and Tape Recording” on page 54, and you want DFSMSrmm to perform the EDGSSSI processing when it starts by replying RETRY to message EDG0103D.
- You use the EDGRESET utility or the START command

  ```
  S DFRMM,OPT=RESET
  ```

  and plan to restart DFSMSrmm without an IPL.

Run DFSMSrmm as a subsystem that is started under the JES2 or JES3 subsystem, rather than under the master subsystem. When choosing the DFSMSrmm subsystem name, the subsystem name must not exactly match the DFSMSrmm procedure name, unless you specify SUB=JES2 or SUBJ=JES3 on each START command. For example, if you use DFRM as the subsystem name, use DFRMM as the procedure name.

You must run DFSMSrmm under the JES2 or JES3 subsystem to use these functions:

- Use the DFSMSrmm NOTIFY function which automatically notifies volume and product owners when the volumes they own become eligible for release or when product volumes are added.
- Use the SYSOUT facilities like //SORTOUT DD SYSOUT=*.

Dynamically Adding the DFSMSrmm Subsystem

You can dynamically add the DFSMSrmm subsystem without an IPL. Use the MVS system command SETSSI to add the DFRM subsystem. Issue the SETSSI command from a console that has master authority or a console that has sufficient RACF authority.

To activate the DFSMSrmm subsystem, issue the command:

```
SETSSI ADD,SUBNAME=DFRM
```

Enabling DFSMSrmm and Tape Recording

You enable DFSMSrmm later, in “Step 21: Initializing the DFSMSrmm Subsystem and Tape Recording” on page 54. Refer to this section now if you are interested in the changes you will make later to IEFSSNxx to enable the DFSMSrmm tape recording interface.

Updating IKJTSOxx to Authorize DFSMSrmm Commands

Update IKJTSOxx to authorize RMM commands and utilities. Refer to *z/OS TSO/E Customization* if you use the TSO CSECT facility rather than IKJTSOxx for these updates.

To authorize the TSO command RMM, make the specifications as shown in Figure 4:

```
      AUTHCMD NAMES(RMM)
```

Figure 4. Updating IKJTSOxx to Authorize RMM Commands

To authorize the DFSMSrmm utilities that you can use within TSO/E or call through the TSO/E Service Facility, make the specifications as shown in Figure 5:

```
     AUTHTSF NAMES(EDGHSKP
                   EDGUTIL
                   EDGRPTD
                   EDGAUD)
     AUTHPGM NAMES(EDGHSKP
                   EDGUTIL
                   EDGRPTD
                   EDGAUD)
```

Figure 5. Updating IKJTSOxx to Call DFSMSrmm through TSO

To dynamically obtain the updated version of the IKJEFTxx member, use the TSO PARMLIB UPDATE(**) command in your TSO session.

Updating IFGPSEDI When the Enhanced Data Integrity Function is Activated

The purpose of enhanced data integrity in the IFGPSEDI member of SYS1.PARMLIB is to detect programs that cause data loss due to multiple programs updating a sequential data set simultaneously. This is a useful function but if your system programmer chooses to activate it, DFSMSrmm receives spurious warning messages or ABENDs.

The system programmer activates this function by:

1. Creating SYS1.PARMLIB member IFGPSEDI with MODE(WARN) or MODE(ENFORCE) and
2. Either re-IPLing the system or issuing the command:
   ```
   S IFGEDI
   ```

If the member has MODE(DISABLE) or MODE is omitted, then the member has no effect.

If MODE(WARN) is in effect, then during the normal running of DFSMSrmm programs, the system issues message IEC984I or message IEC985I for the DFSMSrmm journal, ACTIVITY, MESSAGE, REPORT, REPTEXT, and XREPTEXT data sets. You can ignore these messages. IBM recommends that you code those names in the DSN parameter in the IFGPSEDI member to suppress these warning messages.

If MODE(ENFORCE) is in effect, then normal running of DFSMSrmm programs causes ABEND 213-FD for the DFSMSrmm journal, ACTIVITY, MESSAGE, REPORT, REPTEXT, and XREPTEXT data sets. You must code the names of these data sets in the DSN parameter in the IFGPSEDI member to suppress these OPEN failures. Alternately you can change MODE(ENFORCE).

Related Reading: For more information on the journal, see “Step 13: Creating the Journal” on page 40. For more information on the data sets used for inventory management, see “Allocating Data Sets for Inventory Management” on page 279. For further information about enhanced data integrity, see *z/OS MVS Initialization and Tuning Reference*.

Updating SMFPRMxx (Optional)

If you want DFSMSrmm security-type SMF records or audit-type SMF records, update the SMFPRMxx member to ensure that the DFSMSrmm SMF records are generated. Define the SMF record numbers in the DFSMSrmm parmlib member EDGRMMxx using the OPTION command and the SMFAUD operand for auditing records and the SMFSEC operand for security records. For more information on these commands, see Chapter 8, “Using the Parmlib Member EDGRMMxx,” on page 127.

Select two SMF record numbers in the range 128 through 255 that your installation is not currently using. Add your record numbers to the member as described in *z/OS MVS System Management Facilities (SMF)*.

Updating GRSRNLxx (Optional)

Before you begin: Skip this step if you are not using global resource serialization (GRS) to serialize access to resources. If you have multiple systems connected with global resource serialization, decide now how you want GRS to handle the reserve on the control data set.

Recommendations: The following recommendations for setting up resource serialization depend on the volume where the control data set resides and if the volume contains critical data.

- If the volume does not contain other critical data and if real reserves do not cause any shared DASD contention problems, convert the associated SYSTEMS enqueue to a local SYSTEM enqueue, while leaving the hardware reserve in effect. This method gives slightly better DFSMSrmm performance.

Example: Add the reserve name to the global resource serialization exclusion list as shown in Figure 6. This leaves the real hardware reserve in effect, and causes the associated SYSTEMS enqueue to be converted to a local SYSTEM one. You must specify both the major name (QNAME) and the minor name (RNAME) as shown in Figure 6 when converting the SYSTEMS enqueue to a local SYSTEM enqueue. If you only specify the major name, all enqueues used by DFSMSrmm are converted and you will be unable to run DFSMSrmm on multiple systems.

```
RNLDEF RNL(EXCL) TYPE(SPECIFIC)
QNAME(SYSZRMM)
RNAME(MASTER.RESERVE)
```

Figure 6. Converting the SYSTEMS Enqueue to a Local SYSTEM Enqueue

- If the volume contains other critical data and real reserves could impact other systems or cause DASD contention problems, convert the RESERVE to a SYSTEMS enqueue. If you do not change your GRSRNL*xx* parmlib member, DFSMSrmm still functions correctly; but any performance gain from using the real hardware reserve feature is negated by the overhead created from sending the global SYSTEMS enqueue around the GRS ring.

  Example: Add the reserve name to the global resource serialization reserve conversion list as shown in Figure 7 to leave the global SYSTEMS enqueue in effect and remove the real hardware reserve. You must specify both the major name (QNAME) and the minor name (RNAME) as shown in Figure 7 when converting the RESERVE to a SYSTEMS enqueue. If you only specify the major name, all enqueues used by DFSMSrmm are converted and you will be unable to run DFSMSrmm on multiple systems.

```
RNLDEF RNL(CON) TYPE(SPECIFIC)
QNAME(SYSZRMM)
RNAME(MASTER.RESERVE)
```

Figure 7. Converting the RESERVE to a SYSTEMS Enqueue

Table 5 is the list of DFSMSrmm resource symbolic names. Do not use GRS to change or alter any of these ENQs. The resource symbolic names are provided for information only because DFSMSrmm processing manages serialization for you.

Table 5. DFSMSrmm Resource Symbolic Names

Major Name	Minor Name	Resource	Scope
SYSZRMM	HSKP.*dsn.volser*	Inventory management data set serialization	SYSTEMS
SYSZRMM	MASTER.RESERVE	DFSMSrmm control data set serialization	SYSTEMS
SYSZRMM	MHKP.ACTIVE	Serialize inventory management functions on the same DFSMSrmm subsystem	SYSTEM
SYSZRMM	MHKP.*dsn.volser*	Inventory management data set serialization	SYSTEMS
SYSZRMM	RMM.ACTIVE	Ensure only one system run per MVS image	SYSTEM
SYSZRMM	BUFFER.CONTROL	Buffer management	STEP

Table 5. DFSMSrmm Resource Symbolic Names (continued)

Major Name	Minor Name	Resource	Scope
SYSZRMM	EDGINERS.*volser*	Serialize volume labeling	SYSTEMS
SYSZRMM	SHUTDOWN	Serialize DFSMSrmm shutdown and refresh processing	SYSTEM
SYSZRMM	INACTIVE	Serialize DFSMSrmm activation enabling only a single WTOR to be issued to the operator	SYSTEM
SYSZRMM	EXIT_IS_ACTIVE	Exit recovery serialization	SYSTEM
SYSZRMM	WTOR_ENQ	Exit recovery serialization	SYSTEM
SYSZRMM	WTORIPC	TCP/IP error recovery serialization on CLIENT systems	SYSTEM
SYSZRMM	EXIT_*id*_UNAVAIL	Exit recovery serialization where id can be 100 or 200 representing the last three characters of the DFSMSrmm installation exits EDGUX100 or EDGUX200.	SYSTEM

See *z/OS MVS Planning: Global Resource Serialization* for additional information about global resource serialization.

Enabling DFSMSrmm

IBM supplies a tailored SYS1.PARMLIB member, IFAPRD00, that enables the elements and features you ordered. Before you can use them, you must copy the contents of IDAPRD00 to an active IFAPRDxx member that you establish through the PROD parameter in IEASYSxx or the SET PROD operator command. The IFAPRD00 member is not active by default.

To update the IFAPRDxx member dynamically, use the MVS SET system command:

```
SET  PROD=xx
```

For more information on IDAPRDxx, see *z/OS MVS Initialization and Tuning Reference*.

Step 6: Using the Problem Determination Aid Facility (Optional)

Perform this step once for each MVS image. You only need to perform this step if you want an external DASD record of trace data.

The problem determination aid (PDA) facility gathers DFSMSrmm processing information to enable analysis to pinpoint module flow and resource usage that is related to DFSMSrmm problems. The PDA facility is required for IBM Support Center because it traces module and resource flow. The PDA facility consists of in-storage trace, optional DASD log data sets, EDGRMM*xx* parmlib member options, and operator commands to control tracing. See Chapter 18, “Using the Problem Determination Aid Facility,” on page 385 for information about setting up the PDA facility.

Step 7: Setting Up DFSMSrmm Disposition Processing (Optional)

DFSMSrmm disposition processing is optional and provides support for these tasks:

- Providing operators with information to assist them in performing tasks like moving a tape to a specific location
- Generating sticky labels
- Updating the location where a volume resides

See Chapter 19, “Setting Up DFSMSrmm Disposition Processing,” on page 389 for information about setting up DFSMSrmm disposition processing.

Step 8: Updating the Procedure Library

DFSMSrmm Sample Provided in SAMPLIB

- EDGDFRMM Sample to Create a Procedure in SYS1.PROCLIB

Perform this step once for each MVS image.

Create a procedure in SYS1.PROCLIB to start the DFSMSrmm subsystem address space.

Tip: Figure 8 shows sample JCL that you can use to create the procedure.

```
//DFRMM PROC M=00,OPT=MAIN
//IEFPROC EXEC PGM=EDG&OPT.,PARM='&M',TIME=1440,REGION=40M
//EDGPDOX DD DISP=SHR,DSN=RMM.&SYSNAME..RMMPDOX
//EDGPDOY DD DISP=SHR,DSN=RMM.&SYSNAME..RMMPDOY
//dispdd OUTPUT DEST=SYSTEMX,FORMS=LABEL,CLASS=L
```

Figure 8. Creating a Procedure in SYS1.PROCLIB Using the Recommended JCL

The sample JCL in Figure 9 on page 32 shows the use of additional parameters for specifying the MASTER DD statement, the JOURNAL DD statement, the PARMLIB DD statement, and the IEFRDER DD statement.

```
//DFRMM PROC M=00,OPT=MAIN
//IEFPROC EXEC PGM=EDG&OPT.,PARM='&M',TIME=1440,REGION=40M
//PARMLIB DD   DDNAME=IEFRDER
//IEFRDER DD   DISP=SHR,DSN=SYS1.PARMLIB
//MASTER  DD   DISP=SHR,DSN=RMM.CONTROL.DSET
//JOURNAL DD   DISP=SHR,DSN=RMM.JOURNAL.DSET
//EDGPDOX DD   DISP=SHR,DSN=?UID..?HOSTID..RMMPDOX
//EDGPDOY DD   DISP=SHR,DSN=?UID..?HOSTID..RMMPDOY
//dispdd OUTPUT DEST=SYSTEMX,FORMS=LABEL,CLASS=L
```

Figure 9. Creating a Procedure in SYS1.PROCLIB Using Additional Parameters

where:

DFRMM
Specifies the procedure name. You can use any procedure name, one-to-eight characters long, subject to the restriction documented under IEFSSN*xx* parmlib member.

dispdd
dispdd is an OUTPUT JCL statement that you code when you want to create label output data. The name of the output statement must match the name you specified in the DFSMSrmm EDGRMM*xx* parmlib OPTION DISPDDNAME operand as described in “Defining System Options: OPTION” on page 134. You can use any JCL keywords supported on the OUTPUT statement as described in *z/OS MVS JCL Reference*.

EDGPDOX DD and EDGPDOY DD
EDGPDOX DD and EDGPDOY DD are required statements for external trace output recording. If you do not specify these DD statements, no logging of the PDA trace output is performed. When DFSMSrmm swaps the PDA log data sets EDGPDOX and EDGPDOY, DFSMSrmm uses an intermediate data set name for the log data sets. The started task must have ALTER access to this intermediate data set. The data set name is ?UID.RMMPDO.TEMP.RMMPDO.H?SYSID. The ?UID is the DFSMSrmm ID, and ?SYSID is the first seven characters of the DFSMSrmm SYSID that is defined in the DFSMSrmm EDGRMM*xx* parmlib OPTION SYSID operand, as described in Chapter 8, “Using the Parmlib Member EDGRMMxx,” on page 127.

IEFRDER DD
IEFRDER is an optional statement. Use the IEFRDER DD statement to enable operators to specify PARMLIB DD keywords on the START command for the DFRMM procedure. If you do not code the IEFRDER DD and code the PARMLIB DD, you will not be able to specify a different PARMLIB data set name when starting the DFRMM procedure. If you do not code SYS1.PARMLIB, it is not necessary to code the IEFRDER DD statement.

JOURNAL DD
JOURNAL is an optional statement. If you code the JOURNAL DD statement to name the DFSMSrmm journal data set, you cannot easily change the journal name by switching to a different PARMLIB member. You must supply the journal data set name in the DFSMSrmm EDGRMM*xx* parmlib member if you do not code the JOURNAL DD statement and if you require journaling. You must catalog the journal.

M Use M on the PROC statement to specify a parmlib member suffix. When you specify `M=00`, DFSMSrmm uses member EDGRMM00.

MASTER DD
MASTER is an optional statement that identifies the DFSMSrmm control data set name. If you code the MASTER DD statement to name the DFSMSrmm

control data set, you cannot easily change the control data set name by switching to a different parmlib member. You must supply the control data set name in the DFSMSrmm EDGRMM*xx* parmlib member if you do not code the MASTER DD statement.

OPT

Use OPT on the PROC statement to specify whether to enable or disable subsystem interface:

`OPT=RESET` to disable the subsystem interface
`OPT=MAIN` to enable the DFSMSrmm subsystem

PARMLIB DD

PARMLIB is an optional statement. Use the PARMLIB DD statement to identify the data set that contains DFSMSrmm startup parameters when you do not use the system PARMLIB concatenation. If you do not code the PARMLIB DD statement, do not code the IEFRDER DD statement because it is not required. See “Step 18: Starting DFSMSrmm” on page 47 for parameters you can specify with the START command if you specify the PARMLIB DD statement.

If you specify the parmlib data set in the DFSMSrmm procedure, the data set remains allocated while DFSMSrmm is active. If you do not specify the parmlib data set name, DFSMSrmm dynamically allocates PARMLIB by using the concatenated parmlib function of the MVS system.

Recommendation: Do not specify the PARMLIB or IEFRDER DD statement in the DFRMM procedure. Let DFSMSrmm dynamically allocate the PARMLIB to ’SYS1.PARMLIB’ or use the concatenated parmlib support.

REGION

As you determine the REGION size for the DFSMSrmm started procedure, the amount of virtual storage that DFSMSrmm uses depends on the resources you have defined. DFSMSrmm virtual storage usage can be affected by any REGION size controls or restrictions that your systems might have in place such as in IEFUSI. The sample DFRMM procedure specifies REGION=40M, which normally provides all the private region below 16MB and 40MB above 16MB. To enable DFSMSrmm to use all available virtual storage, specify REGION=0M. If you want to set a specific region size, consider the following tips along with the current region size of your DFRMM started procedure, to determine if you need to make any changes to the REGION size:

- The VSAM local shared resources (LSR) buffer pool that is built by the DFSMSrmm subsystem for the control data set is obtained above 16 MB.

  DFSMSrmm builds an LSR buffer pool for the DFRMM started procedure, and also for the EDGUTIL utility batch address space, which has a predetermined size. The LSR buffer pool is 800*data CISZ + 200*index CISZ. Assuming 10 240 for the control data set data CISZ and 2048 for the control data set index CISZ, the value is 800*10 240+200*4 096=8.4 MB. If you use larger CI sizes, more buffer space is required. For example, if you use a 26K data CISZ, a 21.2 MB buffer size is required.

  Recommendation: If the buffer space is larger than 8.6 MB, add the difference to the 40 MB region size that is used by DFSMSrmm and use this value as the REGION size for the DFRMM started procedure.

- DFSMSrmm uses DFSORT during inventory management. Increase the DFSMSrmm REGION size to allow DFSORT to use storage rather than SORTWK*xx* DASD files. The use of storage rather than DASD files can potentially decrease the time that is needed to run DFSMSrmm inventory management. If you change the REGION size, use 1 additional MB for every 2 000 data sets on private volumes.

If you code the MASTER DD statement and the JOURNAL DD statement in the started procedure, you can switch to different data set names by specifying the names in the EDGRMM*xx* parmlib member. To make the switch:

1. Quiesce DFSMSrmm by specifying:

   ```
   F DFRMM,QUIESCE
   ```

   DFSMSrmm frees the current file allocations and pauses the DFSMSrmm processing.
2. Specify the new parmlib member name that contains the data set names.
3. Start DFSMSrmm by specifying:

   ```
   F DFRMM,M=xx
   ```

If you decide to use EDGLABEL or EDGXPROC, you must update the procedure library to include them as well. See "Using the LABEL Procedure" on page 382 for information on the supplied EDGLABEL procedure. See "Replenishing Scratch Volumes in a System-Managed Library" on page 381 for information on the supplied EDGXPROC procedure.

Step 9: Assigning DFSMSrmm a RACF User ID

Perform this step once for each MVS image.

When running on a system with RACF installed, assign DFSMSrmm a RACF user ID by adding a definition to the RACF started procedures table, ICHRIN03, or in the RACF STARTED class. The RACF user ID can be the name of the DFSMSrmm procedure you created in "Step 8: Updating the Procedure Library" on page 31 or any installation-selected RACF user ID you specify. As data sets are created for use by the DFSMSrmm procedure, add the RACF user ID to the access list for the data sets. Table 6 lists the data sets to which DFSMSrmm requires access.

Table 6. Data Sets Requiring Access by the DFSMSrmm RACF User ID

DDNAMES	Access Required
ACTIVITY	Update
EDGPDOX	Alter
EDGPDOY	Alter
MASTER	Control
Parmlib member	Read
JOURNAL	Update
MESSAGE	Update
REPORT	Update
REPTEXT	Update
XREPTEXT	Update

If you plan to use the DFSMSrmm procedures EDGXPROC, BACKUPPROC, or LABEL, you must define the procedures in ICHRIN03 or the STARTED class. For more information on updating ICHRIN03 or the RACF STARTED class, see *z/OS Security Server RACF System Programmer's Guide*.

The DFSMSrmm procedures EDGXPROC and EDGBKUP require READ access to STGADMIN.EDG.HOUSEKEEP and ALTER access to the data sets specified in the BACKUP and JRNLBKUP DD statements. The LABEL procedure requires UPDATE access to STGADMIN.EDG.OPERATOR. If you are using the EDGRESET utility, you should make sure it has ALTER access to the STGADMIN.EDG.RESET.SSI access list. For additional information on authorization needed for the DFSMSrmm user ID, see Chapter 9, "Authorizing DFSMSrmm Users and Ensuring Security," on page 171.

To run DFSMSrmm with DFSMShsm, ABARS, Tivoli Storage Manager, or OAM, you must define their procedure names to RACF with the STARTED class. See *z/OS DFSMShsm Storage Administration Guide* for information about defining the procedure names.

You must define any user ID that requires DFSMSrmm services and makes use of OPERATIONS or privileged attributes to RACF.

If you are using an equivalent security product, review the RACF-related information to determine the changes that might be required to run DFSMSrmm with the equivalent security product.

Step 10: Defining Parmlib Member EDGRMMxx

Perform this step once for each MVS image.

| **Related Reading:**

| • "Implementing DFSMSrmm Client and Server Systems" on page 58 for details
| about setting up DFSMSrmm systems.

| Create a parmlib member EDGRMMxx definition for each DFSMSrmm standard
| system, server system, and client system. The member name is in the form
| EDGRMMxx, where *xx* is a two character alphanumeric suffix of your choice. The
| default is EDGRMM00. A sample parmlib member is available as EDGIVPPM in
| SAMPLIB.

Recommendation: Specify the PARMLIB member EDGRMM*xx* as a member of SYS1.PARMLIB or the PARMLIB concatenation. If you follow this recommendation, you can use the DFSMSrmm startup parameters to avoid stopping and restarting DFSMSrmm. For example, you can use the modify command (F DFRMM,M=xx) to implement updates to the data set and avoid stopping and restarting DFSMSrmm. You can also restart DFSMSrmm using another data set with parameters to implement changes.

The RACF ID associated with the DFSMSrmm started procedure requires READ access to the parmlib member EDGRMM*xx* data set.

Table 7 shows several ways to define the parmlib member.

Table 7. Creating DFSMSrmm Parmlib Definitions

Way to Define	Example
Member of SYS1.PARMLIB	SYS1.PARMLIB(EDGRMMxx)
Member of PARMLIB concatenation	SYSC.PARMLIB(EDGRMMxx)
Member of separate data set	MY.PARMLIB(EDGRMMxx)
Separate sequential data set	USER.PARMLIB

Step 11: Tailoring Parmlib Member EDGRMMxx

Perform this step once for each MVS image.

Tailor the options for parmlib member EDGRMMxx. To see the options and parmlib member examples, see Chapter 8, “Using the Parmlib Member EDGRMMxx,” on page 127.

Parmlib member EDGRMMxx contains the installation options for DFSMSrmm. See Chapter 8, “Using the Parmlib Member EDGRMMxx,” on page 127 for information about the options you can define in the parmlib member. Using EDGRMMxx, you can perform the following tasks:

- Define system options, such as the date format for reports and messages, and the mode in which DFSMSrmm runs
- Prevent a range of tapes from being used on specific systems
- Define pools, such as the range of shelves to use for a pool and whether a pool has RACF tape profile processing
- Tailor mount messages, such as the position of the volume serial number and identifier
- Define security classes for data sets and volumes
- Control the use of bypass label processing for tape volumes
- Define storage locations
- Define controls for running DFSMSrmm vital record processing to apply retention and movement policies

Step 12: Creating the DFSMSrmm Control Data Set

DFSMSrmm Sample Provided in SAMPLIB

- DFSMSrmm EDGJMFAL Sample JCL for Allocating the Control Data Set
- DFSMSrmm EDGJUTIL Sample JCL for Creating the Control Data Set

Perform this step once for each RMMplex, or for each MVS image.

Create the DFSMSrmm control data set, a VSAM key-sequenced data set that contains the complete inventory of the removable media library. DFSMSrmm records all changes made to the inventory, such as adding or deleting volumes, in the control data set.

| You can create the control data set for each RMMplex or for each MVS image. If you have a control data set for each MVS image, each control data set only contains information for that system. DFSMSrmm cannot track tapes that are accidentally moved to another system that has a different control data set.

| **Recommendation:** Create one control data set for each RMMplex. Sharing the control data set across systems allows DFSMSrmm to keep all tape usage information in one place.

Roadmap for Creating the Control Data Set

The following table shows the subtasks and associated procedures for creating the control data set.

Subtask	Associated procedure
Define the control data set.	“Defining the DFSMSrmm Control Data Set”
Calculate DASD space for the control data set.	“Calculating DASD Space for the DFSMSrmm Control Data Set”
Place the control data set.	“Placing the DFSMSrmm Control Data Set” on page 38
Allocate space for the control data set.	“Allocating Space for the Control Data Set” on page 39
Protect the control data set.	“Protecting the Control Data Set” on page 40
Initialize the control data set for DFSMSrmm subsystem use by running the EDGUTIL utility.	“Initializing the Control Data Set” on page 40
Back up the control data set.	“Backing Up the Control Data Set” on page 40

Defining the DFSMSrmm Control Data Set

You can define the DFSMSrmm control data set as either an extended format (EF) or a non-extended format VSAM data set. Using a non-extended format data set for the DFSMSrmm control data set, limits the control data set size to a maximum of 4 GB. Using an EF data set enables you to use VSAM functions such as multivolume allocation, compression, or striping. EF also enables you to define a control data set that uses VSAM extended addressability (EA) to enable the control data set to grow above 4 GB. To define an EF control data set, you must include the DATACLASS keyword on the AMS DEFINE command and reference the correct data class. Refer to *z/OS DFSMS: Using Data Sets* for more information on EF data sets, and refer to *z/OS DFSMSdfp Storage Administration Reference* for information on defining data classes with DSNTYPE=EXT and EXTENDED ADDRESSABILITY=Y.

DFSMSrmm requires CONTROL access to the control data set. The control data set cannot be a SYS1.*xx* data set if the control data set is to be shared. Additionally, batch LSR cannot be used with the DFSMSrmm control data set.

Calculating DASD Space for the DFSMSrmm Control Data Set

Table 8 helps you to calculate DASD space requirements for the DFSMSrmm control data set. To determine the number of resources in the library, such as the number of software products you have, see your answers to Appendix D, “Evaluating Removable Media Management Needs,” on page 431.

Table 8. DFSMSrmm Control Data Set DASD Space Requirements

Control Data Set Content	DASD Space
Control record	1 MB (MB equals approximately 1 000 000 bytes)
Data sets	500 KB for every 1000 data sets
Shelf locations in the library that do not contain volumes	140 KB for every 1000 shelf locations
Shelf locations in storage locations	140 KB for every 1000 shelf locations
Owners	38 KB per 1000 volumes
Software products, average five volumes per product	420 KB for every 1000 software products
Volumes	1 MB for every 1000 volumes

Table 8. DFSMSrmm Control Data Set DASD Space Requirements (continued)

Control Data Set Content	DASD Space
Vital record specifications	212 KB for every 1000 vital record specifications

After calculating the previous figures, increase the total by approximately 50% for free space in the VSAM key-sequenced data set. For example, if the calculated size is 3000 KB, increase it by 50% to 4500 KB to allow for free space. Also consider your expected future growth in numbers of tape volumes and data sets, such as acquiring a new virtual tape solution, and build this expected growth into your size calculations. Use this calculated value in the access method services KILOBYTES parameter on the DEFINE CLUSTER command shown in Figure 10 on page 39. See "Monitoring the Space Used by the Control Data Set" on page 330 for information about monitoring the DFSMSrmm control data set space usage.

The size of the records in the DFSMSrmm control data set is variable, and record sizes increase as new function is added. Thus, the space required for your DFSMSrmm control data set increases over time. For example, each time a volume record is updated, such as during inventory management or when a new data set is written on a volume, the associated control data set volume records change in size or increase in size as they migrate to the new level dynamically. The value you use for free space enables DFSMSrmm and VSAM to handle the record size changes and allows you the ability to easily add new resources to DFSMSrmm at any time. Because of the way that DFSMSrmm handles variable length records, it is recommended that you do not use FREESPACE(0 0) when defining the DFSMSrmm control data set.

Placing the DFSMSrmm Control Data Set

For each RMMplex, create one control data set on the main, or most active, system in your complex.

If the volume where you place the control data set is system-managed, select a storage class name that has the GUARANTEED SPACE attribute, and substitute it in the example for the STORAGECLASS parameter in Figure 10 on page 39. If it is not system-managed, remove the STORAGECLASS parameter.

If you are running DFSMSrmm on more than one system, decide which systems in an RMMplex will share the control data set using DASD sharing and which will share the control data set using DFSMSrmm client/server support for z/OS. Each standard system and server system needs to share the control data set using DASD sharing. The control data set must be cataloged in a user catalog shared by all the standard systems and server systems in the RMMplex.

Put the control data set on a different volume than the journal, which is also shared. Separating the two data sets optimizes data integrity, since the journal is a copy of changes made to the control data set.

If you plan to use DFSMSdss to back up the control data set, place the control data set on a concurrent copy capable volume.

To avoid a potential deadlock on the volume where the DFSMSrmm control data set is placed, you should consider the other data that you place on the volume. A deadlock can occur when a program other than DFSMSrmm reserves the volume where the control data set resides and requests DFSMSrmm for service. If the DFSMSrmm control data set requires additional extents because of the request, then a deadlock can occur.

DFSMSrmm uses RESERVE/RELEASE to control access to the control data set and ensure integrity. If your installation is using global resource serialization, see “Step 4: Updating Installation Exits” on page 25 and “Updating GRSRNLxx (Optional)” on page 28.

Place the DFSMSrmm control data set and journal on the highest performing DASD in your installation. DFSMSrmm can benefit from features like caching and DASD fastwrite and support for concurrent copy. Consider using the storage class attribute AVAILABILITY=CONTINUOUS for the control data set. This does not remove the need for journaling in DFSMSrmm, however, as the journal is required when reconstructing the control data set from backups.

Allocating Space for the Control Data Set

When you allocate the DFSMSrmm control data set index and data components, the index and data components must be on the same volume because DFSMSrmm does not support them being allocated on separate volumes.

Use JCL similar to that shown in Figure 10 to allocate space for the control data set on the master system:

```
//IDCAMS   EXEC PGM=IDCAMS,REGION=1M
//SYSPRINT DD   SYSOUT=*
//DASD     DD   DISP=SHR,UNIT=SYSDA,VOL=SER=8E5U04
//SYSIN    DD   *
 DEFINE CLUSTER(NAME(RMM.CONTROL.DSET) -
               FILE(DASD) -
               FREESPACE(15 0) -
               KEYS(56 0) -
               REUSE -
               RECSZ(512 9216) -
               SHR(3 3) -
               KILOBYTES(4500 1500) -
               STORAGECLASS(gspace) -
               VOLUMES(8E5U04)) -
          DATA(NAME(RMM.CONTROL.DSET.DATA) -
               CISZ(26624)) -
         INDEX(NAME(RMM.CONTROL.DSET.INDEX) -
               CISZ(2048))
  /*
```

Figure 10. Allocating DASD Space for the Control Data Set

Recommendation: Use a CISZ of 26624 bytes for the data component of the control data set to help improve inventory management run times through reduced I/O to the control data set. Any suitable CISZ between 10240 and 26624 that meets your needs can be used.

where:

KILOBYTES
: Is the space value you calculated in “Calculating DASD Space for the DFSMSrmm Control Data Set” on page 37. Choose a secondary space value that allows the control data set to grow in the future.

NAME()
: Specifies the name of the control data set. Use the same name you

assigned in the parmlib member EDGRMMxx under OPTION DSNAME(name) or in the MASTER DD statement in the DFSMSrmm procedure.

FREESPACE Specifies how much free space VSAM reserves in the data set for future additions. For more information on changing the FREESPACE values, see *z/OS DFSMS: Using Data Sets*.

Protecting the Control Data Set

Protect the control data set by ensuring that a RACF DATASET profile protects it. To prevent inadvertent updates to the control data set, specify a RACF universal access of NONE. Give CONTROL access only to users authorized to perform functions independent of the subsystem, such as restoring and reorganizing the control data set and using program EDGUTIL. Give CONTROL access to the RACF user ID assigned to the DFSMSrmm procedure in “Step 9: Assigning DFSMSrmm a RACF User ID” on page 34.

Initializing the Control Data Set

To initialize the control data set, run the DFSMSrmm EDGUTIL utility. Figure 11 is sample JCL you can use to create the control data set.

```
//EDGUTIL EXEC PGM=EDGUTIL,PARM='CREATE'
//SYSPRINT DD SYSOUT=*
//MASTER   DD DSN=RMM.CONTROL.DSET,DISP=SHR
//SYSIN    DD CONROL EXTENDEDBIN(YES) STACKEDVOLUME(YES)
CONTROL
/*
```

Figure 11. Initializing the Control Data Set

See “Creating or Updating the Control Data Set Control Record” on page 341 for additional information on SYSIN values you can use.

Backing Up the Control Data Set

Plan to use the DFSMSrmm utilities EDGBKUP and EDGHSKP to back up the control data set. You can automatically back up the control data set as part of inventory management if you use EDGHSKP which also clears the journal. You can also use the parmlib OPTION command JOURNALFULL operand threshold value to start a backup procedure.

Back up the control data set to DASD. You can also back up the control data set to tape using DFSMSdss. Once the backup is complete, you can move or copy the backup file to any storage medium.

Recommendation: Keep the latest backup available on DASD to provide the fastest recovery time for the control data set.

See “Backing Up the Control Data Set” on page 310 for more information.

Step 13: Creating the Journal

DFSMSrmm Sample Provided in SAMPLIB

- EDGJNLAL Sample JCL for Allocating the Journal

Perform Step 13 once for each RMMplex or MVS image, depending on how you created the control data set. Create one control data set and one journal for each RMMplex.

The journal contains a record of all changes made to the control data set since the last backup. Create the journal and use it to forward recover changes made since the last backup. Each time the control data set is backed up successfully using the EDGHKSP utility, the journal data set is cleared.

DFSMSrmm requires UPDATE access to the journal. The journal cannot be an extended sequential data set.

You should plan to back up the journal and maintain multiple generations of it. See Chapter 14, "Performing Inventory Management," on page 275 for additional information.

Roadmap for Creating the Journal

The following table shows the subtasks and associated procedures for creating the journal.

Subtask	Associated procedure
Calculate DASD space for the journal.	"Calculating DASD Space for the Journal"
Place the journal.	"Placing the Journal" on page 42
Allocate space for the journal.	"Allocating Space for the Journal" on page 42
Protect the journal.	"Protecting the Journal" on page 43
Back up the journal.	"Backing Up the Journal" on page 43

Calculating DASD Space for the Journal

Table 9 helps you to calculate DASD space requirements for the journal. To determine the number of resources in your library, such as the number of scratch mounts that you have, see your answers to Appendix D, "Evaluating Removable Media Management Needs," on page 431. Base the following calculations on your requirements from one backup to the next. To ensure that the journal has enough space when there is an unexpected increase in tape activity, increase the calculated amount by 50% or more. The maximum journal size is 65 535 tracks.

Table 9. DFSMSrmm Journal DASD Space Requirements

Journal Content	DASD Space
Changes by users	1.5 KB for each change made
Data sets	1.5 KB for each data set retained by a vital record specification
Data sets no longer retained by a vital record specification	1.5 KB for each data set no longer retained by a vital record specification
Expiring volumes	1.5 KB for each expiring volume
Non-scratch mounts	6.7 KB for each mount
Scratch mounts	8.3 KB for each mount
Volumes	1.5 KB for each volume retained by a vital record specification
Volume checked in/out	2.6 KB for each volume in or out of the library

Table 9. DFSMSrmm Journal DASD Space Requirements (continued)

Journal Content	DASD Space
Volumes returned to scratch	3.0 KB for each volume returned to scratch
Volumes to and from storage locations	3.5 KB for each volume moved to or from a storage location
Volumes no longer retained by a vital record specification	1.5 KB for each volume that has not reached its expiration date and is no longer retained by a vital record specification
Volumes that are exported or imported	1.5 KB for each logical volume exported or imported
Vital record specifications	1.3 KB for each vital record specification created

Convert the final figure into a space allocation. The journal has a record format of variable-length blocked records. You do not need to specify the record format information when allocating the journal, because DFSMSrmm sets the correct values when it opens the journal data set.

To calculate the space required, divide the total KB of space by 4, as allocation will be by average record size using a 4 K value. This calculation gives you the number to use in the space allocation (SPACE=) shown in Figure 12.

Placing the Journal

Place the journal on a different volume than the DFSMSrmm control data set. If the selected volume is system-managed, use the STORCLAS parameter as shown in Figure 12, and select a storage class with the GUARANTEED SPACE attribute. By using the GUARANTEED SPACE attribute, you ensure that the journal is allocated to a specific volume. This volume is different than the one on which the control data set resides. Remove the STORCLAS parameter if the volume is not system-managed.

Place the DFSMSrmm journal on the highest performing DASD in your installation. DFSMSrmm can benefit from features like caching and DASD fastwrite and support for concurrent copy. Consider using the storage class attribute AVAILABILITY=CONTINUOUS for the journal.

When the enhanced data integrity function (EDI) is activated, you must include the journal in the parmlib member IFGPSEDI.

Systems sharing a control data set must also share the same journal.

Allocating Space for the Journal

Figure 12 shows JCL for allocating space for the journal.

```
//JOURNAL EXEC PGM=IEFBR14
//SYSPRINT DD  SYSOUT=*
//JOURNAL  DD  DISP=(NEW,CATLG),DSN=RMM.JOURNAL.DSET,
//             UNIT=SYSDA,VOL=SER=volser,STORCLAS=store_class,
//             AVGREC=U,SPACE=(4096,(pp))
```

Figure 12. Allocating Space for the Journal

where:

pp Specifies the calculated primary space.

DSN= Specifies the name of the journal. Use the same name you assigned in the parmlib member EDGRMMxx for OPTION JRNLNAME(name) or in the JOURNAL DD statement in the DFSMSrmm procedure.

SPACE=(4096,(pp))
Specifies the space allocation for the journal. Use the value you calculated in "Calculating DASD Space for the Journal" on page 41.

You must catalog the journal.

You do not have to specify any DCB information for the data set because the required values are set when DFSMSrmm opens the data set. Any conflicting values that you supply are overridden.

You can create the journal in multiple extents, but increases in size are not permitted once DFSMSrmm uses the data set. If you specify a secondary space allocation for the data set, DFSMSrmm ignores it.

Protecting the Journal

Protect the journal by ensuring that a RACF DATASET profile protects it. To prevent inadvertent updates to the journal, specify a RACF universal access of NONE. Give READ access only to users that are authorized to perform functions independent of the subsystem, such as restoring and reorganizing the control data set. Give UPDATE access to the RACF user ID assigned to the DFSMSrmm procedure in "Step 9: Assigning DFSMSrmm a RACF User ID" on page 34.

Backing Up the Journal

Plan to use the DFSMSrmm utilities EDGBKUP and EDGHSKP to back up the journal. You can automatically back up the journal as part of inventory management if you use EDGHSKP which also clears the journal. You can also use the parmlib OPTION JOURNALFULL to automatically start a backup procedure. DFSMSrmm uses IDCAMS to back up the journal.

Back up the journal to DASD. Once the backup is complete, you can move or copy the backup file to any storage medium. Be sure to maintain multiple generations of the journal.

Step 14: Authorizing Users

| Perform this step once for each RMMplex.

Refer to Chapter 9, "Authorizing DFSMSrmm Users and Ensuring Security," on page 171 to authorize your users to DFSMSrmm resources.

Step 15: Making the DFSMSrmm ISPF Dialog Available to Users

Modify the ISPF environment so you can run the DFSMSrmm ISPF dialog. Use one of the following techniques to make the dialog available:

- Concatenate the DFSMSrmm target ISPF libraries with your existing ISPF libraries and use one of the following methods:
  1. Add DFSMSrmm to an ISPF dialog as described in "Adding DFSMSrmm to an ISPF Selection Panel" on page 44.

| 2. Use the method supplied by DFSMS. DFSMSrmm is selection option 'R' from
| the ISMF primary option menu.

3. Use the RMMISPF EXEC to enter the dialog.

- Use the ISPF LIBDEF facility to make the DFSMSrmm target ISPF libraries available to your users, and use the EDGRMLIB EXEC to enter the dialog.

  If you are using the LIBDEF facility and you are not using the same target library names as listed in the *z/OS Program Directory* and *ServerPac: Installing Your Order*, you must modify the EDGRMLIB EXEC or produce your own similar exec or CLIST.

Adding DFSMSrmm to an ISPF Selection Panel

You can add a selection to an ISPF selection panel so that users can choose DFSMSrmm. To add the selection:

1. Add a selection for DFSMSrmm to the body of the chosen ISPF selection panel. For example, add:
   ```
   R  DFSMSrmm    Invoke DFSMSrmm
   ```
2. Add one of the following statements to the ZSEL processing list in the)PROC section of the chosen ISPF panel:
   - If you are not using LIBDEF:
     ```
     R,'CMD(%RMMISPF) NEWAPPL(EDG)'
     ```
   - If you are using LIBDEF:
     ```
     R,'CMD(%EDGRMLIB)'
     ```

Figure 13 on page 45 shows the ISPF Utility Selection Menu with step 1 and 2 changes made. LIBDEF was not used:

```
%-------------------------  UTILITY SELECTION MENU  ---------------------
%OPTION  ===>_ZCMD
%
%    1 +LIBRARY    - Compress or print data set.  Print index listing.
+                      Print, rename, delete, or browse members
%    2 +DATASET    - Allocate, rename, delete, catalog, uncatalog, or
+                      display information of an entire data set
%    3 +MOVE/COPY  - Move, copy, or promote members or data sets
%    4 +DSLIST     - Print or display (to process) list of data set names
+                      Print or display VTOC information
%    5 +RESET      - Reset statistics for members of ISPF library
%    6 +HARDCOPY   - Initiate hardcopy output
%    8 +OUTLIST    - Display, delete, or print held job output
%    9 +COMMANDS   - Create/change an application command table
%   10 +CONVERT    - Convert old format menus/messages to new format
%   11 +FORMAT     - Format definition for formatted data Edit/Browse
%   12 +SUPERC     - Compare data sets (Standard dialog)
%   13 +SUPERCE    - Compare data sets (Extended dialog)
%   14 +SEARCH-FOR - Search data sets for strings of data
%    R +DFSMSrmm   - Invoke DFSMSrmm
)INIT
  .HELP = ISR30000
)PROC
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
               1,'PGM(ISRUDA) PARM(ISRUDA1)'
               2,'PGM(ISRUDA) PARM(ISRUDA2)'
               3,'PGM(ISRUMC)'
               4,'PGM(ISRUDL) PARM(ISRUDLP)'
               5,'PGM(ISRURS)'
               6,'PGM(ISRUHC)'
               8,'PGM(ISRUOLP)'
               9,'PANEL(ISPUCMA)'
              10,'PGM(ISRQCM) PARM(ISRQCMP)'
              11,'PGM(ISRFMT)'
              12,'PGM(ISRSSM)'
              13,'PGM(ISRSEPRM) NOCHECK'
              14,'PGM(ISRSFM)'
              R,'CMD(%RMMISPF) NEWAPPL(EDG)'
             ' ',' '
               *,'?' )
  &ZTRAIL = .TRAIL
)END
```

Figure 13. Adding DFSMSrmm to ISPF

Modifying an ISPF Selection Panel

You can modify the selection so that only the USER option of the DFSMSrmm dialog is available to the majority of end users.

- If you are not using LIBDEF, add to the)PROC section:

  ```
  R,'CMD(%RMMISPF USER) NEWAPPL(EDG)'
  ```

- If you are using LIBDEF, modify the supplied EDGRMLIB EXEC to include the USER parameter on the RMMISPF EXEC call.

  For example, enter the following REXX statement to replace the one supplied in EDGRMLIB.

  ```
  address "ISPEXEC" "SELECT CMD(%RMMISPF USER) NEWAPPL(EDG) PASSLIB"
  ```

Make the DFSMSrmm panel library, tables, skeletons, messages, and REXX execs available to users. If you are not using LIBDEF, Table 10 on page 46 shows the names of the default libraries that you concatenate to the DD statements in the TSO logon procedure or a user-supplied start-up CLIST. If you are using LIBDEF, the EDGRMLIB EXEC allocates these default libraries to the user. If you changed the names of the target libraries on your system, modify the EDGRMLIB exec to contain the new library names.

Table 10 lists the DFSMSrmm libraries by the default target names.

Table 10. Default Libraries to Concatenate

DFSMSrmm Data Set Name	DD Statement	Content
SYS1.SEDGEXE1	SYSPROC (or SYSEXEC)	REXX execs
SYS1.SEDGMENU	ISPMLIB	English messages
SYS1.SEDGPENU	ISPPLIB	English panels
SYS1.DGTSLIB	ISPSLIB	Skeletons
SYS1.DGTTLIB	ISPTLIB	Tables

The target library SYS1.SEDGEXE1 has a fixed-block record format. The SYS1.DGTSLIB library and the SYS1.DGTTLIB library are shared with other DFSMS/MVS components.

To customize the DFSMSrmm Report Generator library names, you can modify the EDGRMAIN member of SYS1.SEDGEXE1. Refer to *z/OS DFSMSrmm Reporting* for details.

Use the ISPF ISPPREP facility to build preprocessed versions of the panels. For more information on using ISPPREP, see *z/OS ISPF User's Guide Volume I*.

Step 16: Restarting MVS with DFSMSrmm Implemented

You are ready to start the system with DFSMSrmm implemented. You cannot restart MVS without an IPL. You can, however, avoid an IPL when performing one of the following tasks:

- Changing subsystem name information in IEFSSNxx
- Setting SMF information in SMFPRMxx
- Changing GRS RNL definitions in GRSRNLxx
- Using the PARMLIB UPDATE command to implement changes to IKJTSOxx

If you performed an IPL during Step 1, described in “Step 1: Preparing to Implement DFSMSrmm” on page 23, you have to re-IPL only if you cannot dynamically implement changes to the MVS system parmlib members or modified installation exits. You must re-IPL with CLPA to include new levels of DFSMSrmm code that have LPALIB as the target library.

Step 17: Tailoring DFSMSrmm Set Up

At this time, you must perform some additional set up for some of the DFSMSrmm functions. See *z/OS Migration* for information about steps that you need to perform prior to installing DFSMSrmm.

See these sections for more information:

- Using volumes with special expiration dates, see “Managing Volumes with Special Dates” on page 78.
- Using volumes with duplicate volume serial numbers, see “Managing Volumes with Duplicate Volume Serial Numbers” on page 81.

- Using DFSMSrmm to control tape label types that can be used on volumes, see Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171.
- Enabling the use of no label output for scratch volumes with JES3, see Chapter 13, “Running DFSMSrmm with JES3,” on page 271.
- Synchronizing the DFSMSrmm control data set and other user catalogs, see “Running DFSMSrmm Catalog Synchronization” on page 305.
- Enabling DFSMSrmm stacked volume support, see “Setting up DFSMSrmm Stacked Volume Support” on page 346.
- Managing volumes using ACS routines, see “Using SMS Tape Storage Groups for DFSMSrmm Scratch Pooling” on page 71.

Step 18: Starting DFSMSrmm

Set the OPMODE operand in parmlib member EDGRMMxx to M to start DFSMSrmm in manual mode. Tape mounts are processed as they were before you began implementing DFSMSrmm. Refer to “Defining System Options: OPTION” on page 134 for information about setting OPMODE. See *z/OS DFSMSrmm Guide and Reference* for information about operator procedures.

Then, issue the MVS START command to start the DFSMSrmm subsystem, as shown in Figure 14:

```
S DFRMM,M=xx,SUB=jesp
```

Figure 14. Starting DFSMSrmm

where:

DFRMM	Specifies the procedure name.
M=*xx*	Specifies your chosen parmlib member name suffix.
SUB=*jesp*	This is an optional keyword to identify the name of the job entry subsystem. DFSMSrmm must not run under the MSTR subsystem. If the DFSMSrmm procedure name you use matches the subsystem name in IEFSSN*xx* as described in “Updating IEFSSNxx” on page 25, you must specify the SUB keyword when starting the DFSMSrmm procedure. The value *jesp* is the procedure name of your job entry subsystem and can be specified as SUB=JES3 or SUB=JES2.

If your DFSMSrmm procedure has a PARMLIB DD statement with DDNAME=IEFRDER, there are several other parameters that you can specify with the START command. You can specify keyword parameters that are supported on a DD statement. For example, you can specify DSN= to override the control data set name on the IEFRDER statement, as shown in Figure 15:

```
S DFRMM.name,M=xx,DSN=parmlib_name
```

Figure 15. Starting DFSMSrmm with Additional Parameters

where:

name	Specifies a name other than DFRMM by which you can call the DFRMM procedure. You can then use this name on STOP and MODIFY commands.

M=*xx* Specifies a specific parmlib member with which DFSMSrmm should be started instead of the default parmlib member.

DSN=*parmlib_name*
Specifies an alternative data set name to be used as a parameter for this restart of the DFSMSrmm subsystem.

DFSMSrmm is now running in manual mode.

Stopping DFSMSrmm

You can stop the DFSMSrmm subsystem with the MVS STOP command, as shown in Figure 16:

```
P DFRMM
```

Figure 16. Stopping DFSMSrmm

If the operator defers the reply to the WTORs, DFSMSrmm shutdown might wait until the operator enters a reply. If shutdown is delayed, DFSMSrmm issues message EDG0154I to notify the operator that action is required.

When DFSMSrmm is stopped, DFSMSrmm completes the following tasks:

- DFSMSrmm completes any requests being processed at the time DFSMSrmm is stopped.
- DFSMSrmm completes any requests accepted by DFSMSrmm but that are currently waiting to be processed, except for requests to start DFSMSrmm inventory management. DFSMSrmm fails requests to start DFSMSrmm inventory management.
- If DFSMSrmm is already quiesced, DFSMSrmm does not process the waiting requests. DFSMSrmm issues message EDG1105I indicating that the requests are still waiting to be processed. If any of the requests are for catalog activity notifications, DFSMSrmm provides additional message text in message EDG1106I. DFSMSrmm issues a WTOR EDG1107D with options to STOP, QUIESCE, RESTART DFSMSrmm with the same parmlib member or RESTART DFSMSrmm with a new parmlib member.
- DFSMSrmm fails new requests with the RMM not active reason.

To stop DFSMSrmm and still allow tapes to be used, issue the commands shown in Figure 17 to disable the DFSMSrmm subsystem interface until the next time you IPL or start the DFSMSrmm subsystem.

```
S DFRMM,OPT=RESET
P DFRMM
```

Figure 17. Disabling the DFSMSrmm Subsystem Interface

Before DFSMSrmm disables the subsystem interface, it ensures that the user who made the request is authorized. To disable the DFSMSrmm subsystem interface by using the RESET option, you must have a security profile in place. If your installation does not have RACF or an equivalent security product installed, DFSMSrmm allows the reset request. In order to control the request, you can write a RACROUTE exit to test for the security profile and return an acceptable return code. See Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171 for additional information.

Tell the operator to reply to any DFSMSrmm WTORs issued between issuing the commands shown in Figure 17 on page 48. The operator must reply to any DFSMSrmm WTORs before entering the STOP command. If the RESET option is used to allow tape usage to continue, the operator should first enter the command:

```
S DFRMM,OPT=RESET
```

The operator should reply to the outstanding WTORs and then enter the STOP command. If the operator defers the reply to the WTORs, DFSMSrmm shutdown might hang until the operator enters a reply. If shutdown is delayed, DFSMSrmm issues message EDG0154I to notify the operator that action is required.

Quiescing DFSMSrmm

When you quiesce DFSMSrmm:

- DFSMSrmm completes requests that are being processed when the quiesce is requested. Note that DFSMSrmm might fail the request if manual recovery is to be performed or if there are I/O errors on the control data set.
- Any requests accepted by DFSMSrmm are left on work queues until DFSMSrmm recovery and refresh is completed.
- DFSMSrmm fails any new requests with an I/O error during manual recovery or RMM not ACTIVE when DFSMSrmm is quiesced by command.
- In a multihost environment, conditions, which result in an automatic quiesce of DFSMSrmm (such as control data set errors from which DFSMSrmm cannot automatically recover), cause the quiesce on all hosts sharing the control data set. Only after all hosts have successfully quiesced can the control data set be recovered. Manually issuing a DFSMSrmm quiesce affects only the host on which you issue the command. If you want all hosts quiesced, you must issue the command on each host that is sharing the control data set.

To quiesce DFSMSrmm, the operator should issue the command to quiesce the subsystem, allow DFSMSrmm to deallocate the control data set and the journal, and allow recovery processing to be performed.

```
MODIFY DFRMM,QUIESCE
```

```
MODIFY DFRMM,QUIESCE
```

Figure 18. Quiescing the DFSMSrmm Subsystem Interface

Restarting DFSMSrmm

To restart DFSMSrmm after it has been quiesced or to change the DFSMSrmm options, the operator can issue the command specified with a member name.

```
MODIFY DFRMM,M=xx
```

Checking DFSMSrmm Status

Your operator can obtain DFSMSrmm status information by issuing the following command.

```
F DFRMM,QUERY ACTIVE
```

The operator can also use the abbreviations

```
Q ACT
```

and

```
Q A
```

Status information that is available includes:

- The number of requests waiting to be processed
- The number of local requests and server requests
- If DFSMSrmm is active or quiesced
- If the journal is enabled, disabled, or locked
- The status of the listener task, either active or inactive

Step 19: Defining Resources

The following sections help you define these resources to DFSMSrmm: shelves, volumes, owners, and vital record specifications. For detailed information about how to perform these tasks, see *z/OS DFSMSrmm Guide and Reference*. If you are migrating to DFSMSrmm, refer to *Converting to Removable Media Manager: A Practical Guide*, SG24-4998, and the DFSMSrmm SAMPLIB documentation member, EDGCMM01.

Defining Shelf Locations

Shelf locations in the removable media library are called rack numbers. Shelf locations in storage locations are called bin numbers. To define shelf locations to DFSMSrmm, use RMM ADDRACK or ADDBIN subcommands or the DFSMSrmm ISPF dialog.

You can optionally define a rack number for every volume that you plan to define to DFSMSrmm. You do not have to define all possible shelf locations now because you can add them at any time. If you want to logically divide the library into pools, see “Organizing the Library by Pools” on page 63.

If volumes are designated for use in an automated tape library, you must define rack numbers for the volumes when the external and internal volume serial numbers are not the same.

You can optionally define a bin number for every shelf location in a storage location. For shelf locations in DFSMSrmm built-in storage locations, LOCAL, DISTANT, and REMOTE, you provide an initial count and DFSMSrmm assigns the bin numbers.

You can use the LOCDEF command in the EDGRMMxx member of parmlib to define additional locations to DFSMSrmm and to further define existing locations. See “Defining Storage Locations: LOCDEF” on page 127 and Chapter 7, “Managing Storage Locations,” on page 117. For shelf locations in installation defined storage locations, provide a count and an initial bin number.

Defining Owner Information to DFSMSrmm

Use the RMM ADDOWNER subcommand or the Add Owner Details panel in the DFSMSrmm ISPF dialog to define owner information to DFSMSrmm. You must define owner information to DFSMSrmm before defining owner information for non-scratch volumes, software products, and vital record specifications.

If you plan to define volumes using the RMM ADDVOLUME subcommand and want to assign volume ownership, you must predefine those owners to DFSMSrmm before you add the volumes.

You must also define owners before you use the RMM ADDVRS subcommand or the DFSMSrmm Add Data Set VRS, DFSMSrmm Add Name VRS, or DFSMSrmm Add Volume VRS panel in the DFSMSrmm ISPF dialog.

DFSMSrmm automatically adds owner information to the control data set when volumes are read or data is written to volumes. DFSMSrmm uses the RACF user ID for a job, when available, as the volume owner. When there is no RACF user ID for a job, DFSMSrmm uses the job name as the volume owner ID.

To use DFSMSrmm automatic owner notification, you should include the owner's electronic user ID and node when you define an owner to DFSMSrmm. For example, if you want DFSMSrmm to automatically notify owners when their volumes become eligible for release, DFSMSrmm can send the release notification to the owner's electronic address, an MVS or VM user ID and node, that you have defined.

Defining Volumes

Use the RMM ADDVOLUME subcommand or the DFSMSrmm ISPF dialog to add physical volumes, logical volumes, and stacked volumes to DFSMSrmm. The volume serial number and volume status are the basic information needed for DFSMSrmm to recognize a volume. You can add more information using the subcommand or dialog or use the DFSMSrmm running modes to record information about volumes as they are used.

You can add information about volumes that reside in an automated tape library before they are entered into the automated tape library using the RMM ADDVOLUME subcommand or the DFSMSrmm ISPF dialog. For more information about adding volumes to a system-managed tape library and using DFSMSrmm with system-managed tape libraries, see Chapter 5, "Running DFSMSrmm with System-Managed Tape Libraries," on page 85.

You set running modes with the OPMODE operand in the parmlib member EDGRMM*xx*. See "Defining System Options: OPTION" on page 134 for information about the DFSMSrmm running modes. See *z/OS DFSMSrmm Guide and Reference* for details about adding volume information to DFSMSrmm.

Adding Volumes for a New Removable Media Library

If you have a new removable media library or system with all scratch tapes, define all volume information to DFSMSrmm at once.

1. Use the RMM ADDVOLUME subcommand or the DFSMSrmm ISPF dialog to define volume information to DFSMSrmm. Figure 19 shows the minimum information you must add for each volume: volume serial number and volume status.

```
RMM ADDVOLUME volser STATUS(SCRATCH)
```

Figure 19. Defining Minimum Volume Information

   To add volumes that are in a system-managed library, use the command as shown in Figure 20 on page 52. You specify the STATUS(VOLCAT) to obtain volume information from the TCDB.

```
RMM ADDVOLUME volser STATUS(VOLCAT)
```

Figure 20. Defining Volumes in a System-managed Library

To add volumes that are in a manual tape library, use the command as shown in Figure 21. You must specify the MEDIATYPE operand and the RECORDINGFORMAT operand.

```
RMM ADDVOLUME volser LOCATION(mtlname) MEDIATYPE(HPCT)-
      RECORDINGFORMAT(128TRACK) STATUS(SCRATCH)
```

Figure 21. Defining Volumes in a Manual Tape Library

When you are adding a volume to DFSMSrmm, you can specify whether the volume is a logical volume, a physical volume, or a stacked volume. The default volume type is physical volume. If a volume resides in a virtual tape server (VTS), the default is either a logical volume or a stacked volume. If you want DFSMSrmm to manage stacked volumes, you must enable stacked volume support as described in “Setting up DFSMSrmm Stacked Volume Support” on page 346.

You can issue the subcommands in batch.

2. Set the DFSMSrmm running mode in the parmlib member EDGRMMxx to record only mode, warning mode, or protect mode. DFSMSrmm records information about the volumes as they are used. When DFSMSrmm is running in warning mode, DFSMSrmm validates magnetic tapes volumes as you use them. If DFSMSrmm discovers errors, it issues error messages instead of rejecting tapes. When DFSMSrmm is running in protect mode, DFSMSrmm validates magnetic tapes volumes as you use them and rejects magnetic tape volume mounts when errors are encountered. Protect mode is the only mode that provides full verification and validation of volumes.

Adding Volumes from an Existing Removable Media Library

If you have an existing removable media library with no record of tape usage, define basic volume information to DFSMSrmm. Set the DFSMSrmm running mode to record-only mode. DFSMSrmm monitors all tape volume mounts and automatically records and updates information about your defined tape volumes when they are used. DFSMSrmm cannot automatically record optical disk information.

1. Define all required tape volumes to DFSMSrmm as private volumes using the RMM ADDVOLUME subcommand. The following example shows the minimum information you should add:

   ```
   RMM ADDVOLUME volser STATUS(USER) EXPDT(yyyy/ddd)
   ```

   DFSMSrmm uses the parmlib RETPD default if you do not use EXPDT or RETPD to define an expiration date for the volumes. Specifying the EXPDT or RETPD operand allows you to define a time period long enough to gather information about the volumes under DFSMSrmm. Use the information to determine if the volume should be released. Use EXPDT(99/365) if you are unsure how long volumes should be retained. Later, after running DFSMSrmm, defining DFSMSrmm vital record specifications, you should use the information to determine if the volume should be released.

   Specifying STATUS(USER) ensures that DFSMSrmm does not change the expiration date when the first file on the volume is re-written.

2. Set the DFSMSrmm running mode to record-only mode. DFSMSrmm records information about the volumes as they are used but does not validate or reject volumes.
3. When you believe that DFSMSrmm has recorded enough information, set the DFSMSrmm running mode to warning mode. When DFSMSrmm is running in warning mode, DFSMSrmm validates tape volumes. If DFSMSrmm discovers errors, it issues error messages but does not reject tape usage.
4. After DFSMSrmm has been running for a while, check volume usage to determine if some of the volumes you added can be released.

Adding Known Volumes

DFSMSrmm Samples and Execs Provided in SAMPLIB

- EDGCLMS Sample to Convert Volume Information into RMM Commands
- EDGRCSCR Exec to Convert Scratch Pool Information

See the IBM Redbook *Converting to Removable Media Manager: A Practical Guide*, SG24-4998 for information about converting from other products to DFSMSrmm.

1. When you have existing information about your media library, you can build a set of RMM TSO subcommands, one for each volume, and define the information you know. You can also write a REXX exec, CLIST or procedure that converts volume information from your existing tape management system into DFSMSrmm subcommand requests.
2. After adding the information you have available, set the DFSMSrmm running mode to protect mode. Protect mode is the only mode that provides full verification and validation of volumes.

Defining Vital Record Specifications

Vital record specifications are retention and movement policies for data sets and volumes. Define vital record specifications to use DFSMSrmm retention and movement management. The only requirement is that you define them before running your first expiration processing, which you should run once a day.

To define vital record specifications, use the RMM ADDVRS subcommand or the DFSMSrmm ISPF dialog. For more information about ADDVRS and the dialog, see *z/OS DFSMSrmm Guide and Reference*. See “Managing Volumes with Special Dates” on page 78 for information about defining vital record specifications that use special expiration dates for volumes using the EDGUX100 DFSMSrmm installation exit.

To implement the retention and movement policies you define, you must run DFSMSrmm inventory management vital record processing as described in “Running Vital Record Processing” on page 288. You can control the way vital record processing runs using the DFSMSrmm parmlib member EDGRMMxx OPTION command as described in Chapter 8, “Using the Parmlib Member EDGRMMxx,” on page 127.

Step 20: Updating the Operational Procedures

Ensure that operators understand how to use DFSMSrmm by documenting changes in your operational procedures. *z/OS DFSMSrmm Guide and Reference* has an operator procedures section that you should use to educate operators and update your procedures.

Step 21: Initializing the DFSMSrmm Subsystem and Tape Recording

Perform this step once for each MVS image to initialize the DFSMSrmm subsystem interface to enable tape usage.

Enabling the DFSMSrmm Subsystem Interface

Enable the DFSMSrmm subsystem interface to ensure that the interface starts every time you IPL and to ensure that users cannot use tapes before the DFSMSrmm subsystem starts. To enable DFSMSrmm, change the IEFSSNxx member of SYS1.PARMLIB. Add EDGSSSI, as the DFSMSrmm subsystem initialization program, as shown in Figure 22.

```
SUBSYS SUBNAME(JES2)             /* JES2 PRIMARY SUBSYSTEM START   */
  PRIMARY(YES)  START(YES)
SUBSYS SUBNAME(DFRM)             /* Name of the DFSMSrmm subsystem */
  INITRTN(EDGSSSI)               /* RMM initialization routine     */
SUBSYS SUBNAME(AOPA)             /* Netview                        */
```

Figure 22. Changing SYS1.PARMLIB IEFSSNxx

Figure 22 shows the correct relative position of the DFSMSrmm subsystem, updated to include the initialization program.

Changing the DFSMSrmm Running Mode

The DFSMSrmm running modes are set with the parmlib OPTION command OPMODE operand. Each DFSMSrmm running mode provides different levels of information recording and volume validation. If you want to record information about volumes, set the running mode to record-only by setting the OPMODE in the parmlib member EDGRMMxx to R and starting or restarting DFSMSrmm.

Set the running mode to warning mode if you want DFSMSrmm to validate tape volumes. DFSMSrmm issues error messages but does not reject tape usage. If you want to ensure that DFSMSrmm provides full validation and recording functions, set the running mode to protect mode.

After you have verified that DFSMSrmm is providing the required functions, change the DFSMSrmm running mode to protect mode. Protect mode is the only mode that provides full verification and validation of volumes.

If you are converting to DFSMSrmm from another media management system, this step would be part of the final conversion phase. In the final phase, you might be reconverting some data set and volume information. Changing to protect mode ensures that DFSMSrmm controls the management of the volumes defined to it, and controls the use of scratch tapes. When changing to protect mode, either stop running the current media management system or ensure that the two products do not overlap in their function.

Activating the DFSMShsm Interface

To activate the DFSMShsm interface, follow the directions in “Releasing Tapes: EDGTVEXT” on page 199.

Restarting the DFSMSrmm Subsystem

If you changed the DFSMSrmm running mode as described in “Changing the DFSMSrmm Running Mode” and need to activate the DFSMSrmm subsystem interface to implement the change, use the operator MODIFY command shown in Figure 23:

```
F DFRMM,M=xx
```

Figure 23. Restarting the DFSMSrmm Subsystem

where:

M=*xx* Specifies the suffix of parmlib member EDGRMMxx.

The subsystem temporarily stops and reinitializes itself with the new OPMODE. Use this command whenever you update DFSMSrmm parameters and want to implement the change.

During this restart, DFSMSrmm issues the message shown in Figure 24, unless you have IPLed the system since adding EDGSSSI to IEFSSNxx in SYS1.PARMLIB, in which case the interface is already initialized.

```
EDG0103D DFSMSrmm SUBSYSTEM INTERFACE IS INACTIVE -
            ENTER "IGNORE", "CANCEL" OR "RETRY"
```

Figure 24. Message EDG0103D

Reply **RETRY** to this message. In response, DFSMSrmm initializes its subsystem interface. If you reply IGNORE, the DFSMSrmm tape recording function is not activated.

Step 22: Setting Up DFSMSrmm Utilities

There are several DFSMSrmm utilities that you should now set up to run. Run the utilities on the system with the highest level of code to ensure you are taking advantage of DFSMSrmm enhancements and changes to the control data set.

Related Reading:

- See “Scheduling DFSMSrmm Utilities” on page 275 for a sample schedule of all DFSMSrmm utilities. See Chapter 20, “Running DFSMSrmm with the IBM Tivoli Workload Scheduler for z/OS,” on page 395 for information about setting up the IBM Tivoli Workload Scheduler for z/OS to manage the scheduling of DFSMSrmm functions.
- See Chapter 14, “Performing Inventory Management,” on page 275 for information about EDGHSKP to run inventory management activities.
  - Run vital record processing to determine which volumes to retain and what volume moves are required based on vital record specifications. Vital record processing can be run to validate vital record specification information without updating volume and data set information in the DFSMSrmm control data set.
  - Run expiration processing to identify volumes that are ready to be released and returned to scratch.
  - Run storage location management processing to assign shelf locations to volumes that are being moved to storage locations.
  - Run backup of the control data set and the journal to ensure the integrity of the control data set and journal.
- See *z/OS DFSMSrmm Reporting* for information about utilities that help you get information about your removable media library and storage locations, security-related information about volumes and data sets defined to DFSMSrmm,

and audit trail information about volumes, shelf assignments, and user activity. Create movement and inventory reports by producing an extract data set from the control data set and creating a report from the control data set with EDGRPTD.

- See Chapter 15, “Maintaining the Control Data Set,” on page 317 for information about using EDGBKUP and EDGUTIL to back up and recover the control data set, back up the journal, and check the integrity of the information contained in the control data set. Use the EDGBKUP utility to back up and recover the control data set and back up the journal, and the EDGUTIL utility to create, update, and verify the control data set.

  Use DFSMSrmm backup utilities rather than other backup utilities, such as DFSMS access method services EXPORT, because DFSMSrmm provides the necessary serialization and forward recovery functions. Use EDGBKUP to backup and recover the DFSMSrmm control data set when DFSMSrmm is inactive, stopped, or quiesced.
- See Chapter 16, “Initializing and Erasing Tape Volumes,” on page 351 for information about EDGINERS, the DFSMSrmm utility you use to erase and initialize tape volumes either automatically or manually. You can use EDGINERS to replace the DFSMSdfp utility IEHINITT.

Chapter 3. Setting Up DFSMSrmm Client and Server Systems

When a server subsystem starts, identify some basic information for TCP/IP to the DFSMSrmm subsystem using the EDGRMM*xx* parmlib OPTION SERVER operand. See "Defining System Options: OPTION" on page 134 for information on the OPTION command. After the startup of the standard subsystem, DFSMSrmm communicates with TCP/IP and prepares to handle DFSMSrmm requests from client systems. The server verifies that TCP/IP and the specified PORT is available, determines its own default IP address and informs the operator that initialization is successful or issues error messages. As well as normal DFSMSrmm subsystem operation, such as processing of local requests, the server waits for and accepts connection requests, and processes requests from DFSMSrmm client systems. If the server task is unable to start successfully, you can still use DFSMSrmm as a standard subsystem until the server task problem is resolved.

When a client subsystem starts, identify some basic information for TCP/IP to the DFSMSrmm subsystem via the EDGRMM*xx* parmlib OPTION CLIENT operand. See "Defining System Options: OPTION" on page 134 for information on the OPTION command. During startup, the subsystem communicates with TCP/IP and prepares to send DFSMSrmm requests from the client to a server system. The client verifies that TCP/IP is available and that the server can be reached with the specified PORT, determines its own IP address, verifies the control data set ID matches that of the server, and informs the operator that initialization is successful or issues error messages. DFSMSrmm ignores any parmlib options not required for a client system. The client can connect to only one server system at a time. If the defined server is not available, the client issues a WTOR EDG0358D and waits for either the operator to reply with CANCEL, RETRY, or for the server to be available for connection.

After the DFSMSrmm client and server are started, DFSMSrmm fails requests with an I/O error when:

- There is a client or server communication error that cannot be successfully completed.
- The server is restarted using the command: F DFRMM,M=xx. DFSMSrmm recovers from the error when a new server is available and processing continues.

DFSMSrmm issues a WTOR when a TCP/IP error occurs. RETRY processing relies on the operator replying to the WTOR. If the error is resolved and the operator has not replied to the WTOR, DFSMSrmm processing automatically continues and cancels the outstanding WTOR.

Once the client is started, no further verification of the server availability is performed unless a DFSMSrmm request is to be processed. When a request is processed and server communication fails or a time out occurs, and retry still cannot process the request, DFSMSrmm issues message EDG0358D to describe the error and prompts the operator to reply CANCEL or RETRY, and DFSMSrmm automatically continues if the error is resolved.

The DFSMSrmm client system processes most requests by communicating with the server but also by processing those local requests which can be completely processed on the client system. When multiple tasks are being processed, DFSMSrmm maintains a queue in FIFO order. The operator can issue the F DFRMM,Q A command to display the tasks and a summary of the queues.

The DFSMSrmm server processes local requests as if it runs as a standard system. In addition, client requests are accepted and processed synchronously while the requester on the client waits. There is no queue of client requests maintained on the server. The request queues maintained on the server are for local requests only. When you list the active tasks, using 'QUERY ACTIVE', the active local requests are listed together with the accepted client requests.

You must update your firewall to ensure that communication between DFSMSrmm clients and servers is allowed only for the defined IP addresses and ports. The DFSMSrmm subsystem does no authentication, encryption, or verification of connect requests received on the server other than to verify that it is a valid DFSMSrmm request and that control data set IDs match. You should also consider using RACF to protect the use of the IP addresses defined for DFSMSrmm and limit use of the IP address to the DFSMSrmm started task.

Tracing of the IP communication is enabled by the DFSMSrmm support. You will use TCP/IP facilities, such as TCP/IP component trace to gather information about the DFSMSrmm socket processing. See *z/OS DFSMSrmm Guide and Reference* for information on operator procedures.

Implementing DFSMSrmm Client and Server Systems

This topic describes how to implement DFSMSrmm client and server systems.

Related Reading:

- "DFSMSrmm Inventory Management Considerations when Client/Server Support is Enabled" on page 278 for details about DFSMSrmm inventory management considerations when you have set up DFSMSrmm client/server systems.
- See *z/OS DFSMSrmm Guide and Reference* for a complete description of operator procedures.

Before you begin:

- All client and server systems must be at least at this release level.
- DFSMSrmm client/server processing is dependent on Internet Protocol V4.
- You can share the control data set with other systems that run any supported level of DFSMSrmm with any supported level of DFSMSrmm that has appropriate toleration maintenance installed.

You can convert existing DFSMSrmm systems to be either client or server systems, or add new DFSMSrmm systems to the RMMplex. In addition you can merge existing DFSMSrmm systems into the RMMplex by merging the control data sets as described in the redbook DFSMSrmm Primer.

- To implement a client system, follow these steps.
  1. Ensure TCP/IP definition files are updated to identify the server host name, IP address, and port number.
  2. If you have a firewall installed, update your firewall to ensure that communication between DFSMSrmm clients and servers is allowed only for the defined IP addresses and ports. The DFSMSrmm subsystem does no authentication, encryption, or verification of connect requests received on the server other than to verify that it is a valid DFSMSrmm request and that the control data set IDs match. You should also consider using RACF to protect the use of the IP addresses defined for DFSMSrmm and to limit the use of the IP address to the DFSMSrmm started task.

3. Update the parmlib options with the CLIENT operand and select appropriate values for the sub operands. The parmlib operand, CDSID, has to be the same as on the server. If CATSYSID is not already set, add the operand now to define the list of system IDs that share catalogs with the client system or specify that catalogs are shared.
4. Refresh DFSMSrmm with the new EDGRMM*xx* parmlib member.
5. Run EDGHSKP CATSYNCH to synchronize catalogs if needed.
6. Run EDGHSKP with EXPROC regularly to return volumes to scratch status.

- To implement a server system, follow these steps.
  1. Update the parmlib options with the SERVER operand and select appropriate values for the sub operands.
  2. If CATSYSID is not already set, add the operand to define the list of system IDs that share catalogs with the server system or specify that catalogs are shared.
  3. Ensure that TCP/IP definition files are updated to identify the server host name, IP address, and port number.
  4. Ensure your firewall is updated with the client and server IP addresses and port numbers.
  5. Refresh DFSMSrmm with the new EDGRMM*xx* parmlib member.
  6. Run EDGHSKP CATSYNCH to synchronize catalogs if needed.
- You must perform the following steps to implement a standard system that is part of an RMMplex that contains client and server systems.
  1. If CATSYSID is not already set, add the operand to define the list of system IDs that share catalogs with the server system.
  2. Refresh DFSMSrmm with the new EDGRMM*xx* parmlib member.
  3. Run EDGHSKP CATSYNCH to synchronize catalogs if needed.

Using the DFSMSrmm Client and Server Systems

This topic contains recommendations for using the client or server system.

In order to manage a single tape inventory across multiple sysplexes where there is no shared DASD available, you can create one or more DFSMSrmm client systems. Any tape usage on a client system uses the server system to dynamically validate and record tape usage. If the server system is not available for any reason, for example, the IP connection is unavailable or fails, you cannot use tapes on the client system until the server is reconnected or restarted.

All DFSMSrmm users on a client system can use the DFSMSrmm ISPF dialog, RMM TSO subcommands, and batch utilities, including use of DFSMSrmm API and High Level Language API.

Recommendation: Users who need regular access to DFSMSrmm data should log on to the server system. Storage administrators and tape librarians should use a server system or an DFSMSrmm system with direct access to the control data set except when there is a specific reason for using a client system.

- Performing a task for a system-managed tape library that is known to the client and not the server.
  – Ejecting a volume from a system-managed tape library
  – Adding volumes to a system-managed tape library using STATUS(VOLCAT)
  – Changing volume attributes that are also maintained in the TCDB

  - Running expiration processing
  - Confirming moves for exported stacked volumes
  - Running EDGUTIL VERIFY with the TCDB and optionally the Library Manager
- Using DFSMSrmm catalog processing for cataloged data sets that are cataloged only on the client system.
  - Confirming the erasure or initialization of a volume
  - Returning volumes to scratch status and DFSMSrmm is to perform the return to scratch cleanup actions
  - Deleting volumes that contain cataloged data sets

Managing Catalogs in an RMMplex

You can implement DFSMSrmm client/server support with or without sharing catalogs across all of the systems in the RMMplex. You must, however, identify to DFSMSrmm whether catalogs are shared or not using the EDGRMM*xx* parmlib OPTION CATSYSID command described in “OPTION Command Operands” on page 137 in the parmlib member for each system.

DFSMSrmm uncatalogs data sets on a volume, when you specify the EDGRMM*xx* parmlib OPTION UNCATALOG command under these conditions:

- A volume is returned to scratch status, DFSMSrmm uncatalogs all the data sets on the volume.
- The RMM DELETEVOLUME FORCE subcommand is issued for a volume, DFSMSrmm uncatalogs all the data sets on the volume.
- The RMM CHANGEVOLUME DSNAME subcommand is issued for a volume, DFSMSrmm uncatalogs all the data sets on the volume. If the data set name specified on the RMM CHANGEVOLUME subcommand command matches the data set name on the volume, then DFSMSrmm only uncatalogs subsequent data sets.
- The RMM DELETEDATASET subcommand is issued for a data set, DFSMSrmm uncatalogs the data set. Also, DFSMSrmm uncatalogs all data sets recorded on the same volume with higher data set sequence numbers.
- A tape data set is overwritten, DFSMSrmm uncatalogs the data set. Also, all data sets recorded on the same volume with higher data set sequence numbers are uncataloged.
- When the volume on which data sets resides is returned to scratch status, DFSMSrmm uncatalogs data sets.

Recommendation: Use DFSMSrmm RACF processing for TAPEVOL and DATASET profiles maintained in the client’s RACF database when you specify TPRACF(A) or TPRACF(P).

- Confirming the erasure or initialization of a volume
- Returning volumes to scratch status, and DFSMSrmm performs the return to scratch cleanup actions
- Deleting volumes that have RACF profiles

To use the DFSMSrmm catalog processing, you must synchronize the catalogs with the DFSMSrmm control data set. The catalog status for all data sets is maintained in the server system control data set. With unshared catalogs, the UNCATALOG parmlib option, on the server system, cannot be honored for data sets created on the client systems because the server cannot communicate with the client to initiate uncatalog processing. However, when processing is requested from the client, there is special recognition and handling of the request so that any catalog or RACF profile updates can be initiated on the client system. To synchronize the control data set and the catalogs, specify the EDGRMM*xx* parmlib OPTION CATSYSID(list_of_sysids). See "Running DFSMSrmm Catalog Synchronization" on page 305 for additional information. Then run EDGHSKP with CATSYNCH, VERIFY, and EXPROC on the client system.

For example, you have 3 systems; SystemA is to run as a client and has no shared DASD in common with SystemB. SystemC will run in a sysplex and share their own catalogs and the DFSMSrmm control data set.

- For client SystemA, specify the EDGRMM*xx* parmlib OPTION CATSYSID(SystemA) SYSID(SystemA) ..... CLIENT(....) command
- For SystemB, specify the EDGRMM*xx* parmlib OPTION CATSYSID(SystemB,SystemC) SYSID(SystemB) .... SERVER(....) command
- For SystemC, specify the EDGRMM*xx* parmlib OPTION CATSYSID(SystemB,SystemC) SYSID(SystemC) .... command

Run EDGHSKP CATSYNCH once on the client system SystemA, once on one of the systems SystemB or SystemC, and then run EDGUTIL with PARM=UPDATE and SYSIN containing the statement; CONTROL CATSYNCH(YES).

Chapter 4. Organizing the Removable Media Library

You can organize your removable media library by performing the following tasks:

- Create pools of volumes and shelves in your removable media library
- Use Storage Management Subsystem (SMS) management class and storage group to manage volumes in your removable media library
- Use special dates to manage volumes
- Use duplicate volume serial numbers

Organizing the Library by Pools

Related Reading:

- "Defining Pools: VLPOOL" on page 162 for details about the DFSMSrmm EDGRMM*xx* VLPOOL command.

You can use DFSMSrmm EDGRMM*xx* VLPOOL command operands to define pool of volumes and to set installation-defined policies for the volumes in the pool. You can define pools based on criteria like:

- Systems or subsets of systems
- MVS or VM usage
- Use of rack pools to hold foreign tapes
- Media names your installation defines, like 3480, TAPE, CASSETTE
- SMS storage group names
- Release actions for volumes in a pool

Pooling Overview

A *pool* is a group of rack numbers or volumes that share a common prefix. In DFSMSrmm, there are two categories of pools: rack and scratch.

A *rack pool* is shelf space that can be assigned to hold any volumes. Although you can add scratch volumes to these pools, you cannot normally use these volumes to satisfy non-specific mount requests. A rack pool cannot be used with the DFSMSrmm system-based scratch pooling. Rack pools can perform the following functions:

- Hold volumes that are temporarily brought into the library but will be returned to the owner after a period of time
- Hold customer, foreign tapes, and software product volumes
- Contain scratch volumes for use with DFSMSrmm exit-selected scratch pooling

A *scratch pool* is shelf space assigned to hold volumes for use with the DFSMSrmm system-based scratch pooling. The volumes assigned to this shelf space can be used to satisfy scratch requests as long as the volumes are in scratch status. Once the volume has been written to, it becomes a volume with MASTER status, until the volume is returned to scratch status. The volume remains in the same DFSMSrmm system based scratch pool, in that it occupies the same shelf space regardless of status.

The scratch volumes in a system-managed tape library can be from one or more pools. DFSMSrmm does not provide pool selection or validation for volumes in an automated tape library because ACS routines use storage class and storage group to make allocation decisions, and the library manager picks a scratch volume. DFSMSrmm provides pool validation for volumes that reside in a manual tape library. You can pool by storage group, exit-selected pool prefix, or DFSMSrmm system-based pooling when using manual tape libraries.

DFSMSrmm allows you to use these basic types of pooling:

- Pools of shelf space that are based on rack number prefixes. Each range identifies characteristics like management criteria and media name. Use shelf space pools to store volumes that do not match your installation-selected volume ranges and to store duplicate volumes. Define shelf space pools by using the DFSMSrmm EDGRMM*xx* VLPOOL command that is described in "Defining Pools: VLPOOL" on page 162.
- Pools of volumes that are based on the volume serial number prefix. These volumes do not have a rack number or the rack number matches the volume serial number. Each range identifies characteristics like management criteria and media name. Define volume prefix pools by using the DFSMSrmm EDGRMM*xx* VLPOOL command that is described in "Defining Pools: VLPOOL" on page 162.
- Scratch pools. These can be one or more pools of volumes. Scratch pools can be based on name, SMS storage group, prefix, or system. You define scratch pools by using the SYSID, PREFIX, and NAME operands of the DFSMSrmm EDGRMM*xx* VLPOOL command that is described in "Defining Pools: VLPOOL" on page 162.
- Storage groups. When you pool by storage group and use SMS ACS processing to assign a storage group to a tape data set, or the volume is in a system-managed manual tape library, DFSMSrmm ensures that a volume from the correct storage group is mounted. The storage group can be the same across multiple VLPOOL entries. You can use storage group for scratch pooling for system-managed manual tape libraries and non-system managed tape.

Pools can be used in these ways:

- Adding shelf space - DFSMSrmm matches the rack number prefix to the most specific VLPOOL prefix. The rack media name is taken from the matching VLPOOL entry.
- Adding volumes - You can optionally specify RACK or POOL operands to override default processing. Default processing matches the volume serial number prefix to the most specific VLPOOL prefix. If a rack number is found that matches the volume serial number and the specified media name, the volume is stored in the matching shelf pool. If no rack number is found, the volume is in a volume pool. When adding a volume you can specify a storage group name so that the volume can be in a specific scratch pool. If no storage group is specified by the command, DFSMSrmm checks to see if the matching VLPOOL NAME is a storage group, and uses that value as the storage group name. In this case, the scratch pool matches the volume pool.
- Managing access to a volume - For system-managed tapes in a manual tape library, DFSMSrmm validates the mounted volume for the requested pool. You can use the installation exit to ignore the storage group pool and use DFSMSrmm system-based scratch pooling described in "Using Storage Group for Manual Tape Library Pooling" on page 241.
- | Defining actions that should be taken when volumes are ready for release - You can define release actions for volumes on the pool level that might not already be set for individual volumes. For example, you can set the NOTIFY options so that DFSMSrmm sets notification on the release action if it is not already on at the volume level.
- | Selecting scratch pools for new tape data sets - For system-managed tape, SMS ACS processing assigns a storage group. DFSMSrmm uses the storage group name to pool the volumes into a scratch pool. For non-system managed tape,

DFSMSrmm calls SMS ACS processing to allow a storage group to be assigned. If no storage group is assigned, the DFSMSrmm installation exit EDGUX100 is used.

When you pool by pool prefix, selected by EDGUX100 or by SYSID, and the VLPOOL prefix has an associated NAME, DFSMSrmm uses the pool name for mount messages and drive displays but always validates mounted volumes by using the pool prefix and SYSID. Multiple VLPOOL entries can have the same SYSID values and the same NAME values.

Pooling Considerations

If you do not currently use application or user-oriented scratch pools based on job names and data set names, use DFSMSrmm system-based pooling or a general, installation-wide scratch pool. If you currently use application or user-oriented scratch pools that are based on job names and data set names, use SMS ACS routines to assign storage groups. You can use storage groups to pool volumes based on job names and data set names as described in "Using SMS Tape Storage Groups for DFSMSrmm Scratch Pooling" on page 71. You can also implement DFSMSrmm pooling using DFSMSrmm VLPOOL options.

If you are already using some form of scratch pooling based on information such as job name and data set name, review your current use of these pools in order to plan or change your implementation under DFSMSrmm.

Pool Types

You use scratch pools with DFSMSrmm system-based scratch pooling. Define scratch volumes in rack pools and use the RMM GETVOLUME subcommand to claim them or assign them to a user. You can also use EDGUX100 to use rack pools or scratch pools for non-specific tape output requests.

When you define pools for use with specific groups of users or applications, you also need to consider how volumes are defined to DFSMSrmm. You can use the DFSMSrmm ISPF dialog or RMM ADDVOLUME subcommand to define volumes. To add a volume to a specific pool, provide the pool prefix when you issue the RMM ADDVOLUME subcommand or use the DFSMSrmm ISPF dialog. You only need to specify a pool prefix if the volume serial number does not start with the same characters as the pool prefix. When defining scratch volumes you will probably use volume serial numbers that match the rack numbers you include in the pool definition. When adding private volumes which come from another library you will have to specify a pool prefix. You can also use the DFSMSrmm ISPF dialog or RMM TSO CHANGEVOLUME subcommand to move an existing volume to a pool you define, so that the volume becomes part of the pool.

Tape Drive Availability

If you are using specific pools, plan how to set up your scratch tape drives so that a correct scratch tape can be mounted promptly. The benefit of using scratch tapes is that almost any unused tape can be mounted and DFSMSrmm will record new information for the volume. However, as you become more selective about which scratch tapes are used for each non-specific volume request, the benefits of using scratch tapes decrease as your requests become more and more specific.

To make mounting tape volumes easier, consider setting aside certain tape drives for use with identified scratch pools. When using tape cartridge loaders with DFSMSrmm pooling, you need to load each drive with volumes from a single scratch pool, run them in system mode, and direct only the correct non-specific requests to them. Cartridge loaders must not be run in automatic mode because

DFSMSrmm processing depends on the mount message which is not issued when the cartridge loader is in automatic mode and a volume is already loaded. Consider defining different esoteric unit names for each of the tape drives you choose to load with scratch volumes from a single pool.

You can use the EDGUX100 exit to request DFSMSrmm to disable the tape drive cartridge loader to prevent specific scratch pool requests from emptying the loaders through volume rejection. Alternatively, you can direct requests to the correct drives or run the loaders in a mode that prevents them being automatically indexed when a mount is received. Also DFSMSrmm disables the loader if a volume is rejected for a scratch request to prevent the loader from being emptied.

Operator Messages and Tape Drive Displays

To use ACS routines or exit selected scratch pooling, you must define messages IEC501A, IEF233A, IAT5110, and IAT5210 using the DFSMSrmm MNTMSG commands in parmlib as described in "Defining Mount and Fetch Messages: MNTMSG" on page 132. These messages provide the pool identifiers to the operator, the job name, and data set name information to the EDGUX100 exit.

DFSMSrmm can update mount messages and tape drive displays with the pool prefix, storage group name, or pool name. DFSMSrmm can also provide information to some tape automation software installed on your system through the tape mount messages. DFSMSrmm updates messages IEF233A and IEC501A, and IAT5210 for JES3 to include the pool identifier that you select. You can define Basic Tape Library Support (BTLS) category names so that DFSMSrmm can be used to manage volumes in a BTLS-managed tape library. If you use BTLS scratch category names as DFSMSrmm pool names, DFSMSrmm can be used to direct which volumes BTLS will use for each scratch mount.

For JES3, to find out which pool was used to satisfy a request, install the USERMOD EDG3UX71; users can see mount messages with the pool identifier in the output from their batch jobs. Users allowed to view the SYSLOG can see the updated messages in the SYSLOG.

Calculating Pool Size

When planning the size of your pools, remember that the number of characters in the pool prefix determines the maximum pool size. A pool prefix consists of one to five alphanumeric, national, or special characters followed by an *, for example, ABADA*. Table 11 shows the maximum number of volumes each range of pool prefixes can contain:

Table 11. DFSMSrmm Volumes in a Pool Determined by Pool Prefix

Pool Prefix	Maximum Number of Volumes
A*	100 000
AA*	10 000
AAA*	1000
AAAA*	100
AAAAA*	10

The maximum numbers in Table 11 are for numeric suffixes. If you use nonnumeric suffixes, such as pool prefix A with suffixes A00000 through AZZZZZ, you can increase the maximum number of volumes for that pool. For example, using the suffix AAAAA* with alphanumeric characters, $, #, @, and the special characters shown in Table 1 on page xxviii, you can have a maximum of 51 volumes. However, by using nonnumeric suffixes, you limit the use of the COUNT operand on RMM TSO rack and volume related subcommands.

Plan ahead when choosing your pool prefixes, ensuring that the pool prefixes can satisfy all future volume quantities. It is not necessary to define all shelf locations to DFSMSrmm at once. Using pool definitions, you can reserve space for library expansion without taking up space in the control data set. For instance, you could assign a pool prefix ABC*, allowing space for 1000 volumes, but only defining shelves ABC000 - ABC099 initially.

When you use the VLPOOL NAME or SMS storage group name for scratch pooling, you can include multiple VLPOOL PREFIX ranges into the same scratch pool by specifying the same NAME value on multiple VLPOOL definitions or by assigning storage group names for each volume.

Defining Pools in Parmlib Member EDGRMMxx

You must define any pool that you plan to use with DFSMSrmm in the parmlib member EDGRMMxx using the VLPOOL command. This applies to pools used by DFSMSrmm pooling and by the EDGUX100 exit. See “Defining Pools: VLPOOL” on page 162 for information on using VLPOOL command in the parmlib member EDGRMMxx. Also consider shelf space and whether volumes are to be segregated based on media names for any scratch pooling method you use.

Define at least one pool using the VLPOOL command in the DFSMSrmm parmlib member. Use the pools to assign a media name for shelf space and volumes based on a prefix. Define a default pool to ensure that volumes and racks that do not match to a more specific VLPOOL are added with the correct media name and pool name.

Figure 25 shows a VLPOOL command example of the default pool that DFSMSrmm assigns if you do not specify any VLPOOL commands. This default pool includes all shelf locations, all volumes in those shelf locations, and all volumes with no defined shelf location.

```
VLPOOL RACF(N) TYPE(S) -
  EXPDTCHECK(O) MEDIANAME(parmlib_default_medianame) -
  DESCRIPTION('DEFAULT POOL') PREFIX(*)
```

Figure 25. Default VLPOOL Command

In the example:

RACF(N)
Specifies whether to provide RACF tape profile processing for this tape pool. The value N indicates that RACF tape profiles are not created, changed, or deleted for tapes in this pool, regardless of the setting of the system-wide option TPRACF.

TYPE(S)
Specifies the type of pool. The value S means that it is a scratch pool.

EXPDTCHECK(O)
Specifies the processing tape data sets on volumes in this pool that are expiration date protected. The value O means that DFSMSrmm takes no action but allows the operator or automated software to reply as necessary to any write-to-operator messages (IEC507D).

MEDIANAME(*parmlib_default_medianame*)
Specifies a media name for all volumes in this pool. The MEDIANAME value for the default pool is the same value that you specify with the EDGRMM*xx* parmlib OPTION MEDIANAME value.

To separate storage locations by media name, any LOCDEF command media names that you define must be the same as the VLPOOL media names or a subset of the media names. See "Defining Storage Locations: LOCDEF" on page 127 for information on defining storage locations.

DESCRIPTION('DEFAULT POOL')
Describes the pool.

PREFIX(*)
Specifies a generic shelf location prefix that is used in operator messages, TSO subcommands, and the DFSMSrmm ISPF dialog. A single asterisk defines the default pool that contains all rack numbers or volumes not specifically included in any pool. You also have the option to define a pool name that can be used to update operator messages, displays, and can be used as a storage group.

You can also specify a SYSID operand on the VLPOOL command. SYSID associates a scratch pool with a particular system. DFSMSrmm matches the value with the SYSID operand of the OPTION command, also in EDGRMMxx. If the VLPOOL SYSID matches the OPTION SYSID, DFSMSrmm performs the following functions:

- A volume from this pool can be used to satisfy a scratch mount request on this system.
- A volume from any other pool where its SYSID also matches the OPTION SYSID can be used to satisfy a scratch request on this system.

A volume from a pool that has no explicit VLPOOL SYSID can be used to satisfy a scratch request on a system if there are no other VLPOOL definitions that have a SYSID matching the OPTION SYSID. The SYSID values are ignored when a storage group is selected by ACS processing and when a pool is set for a non-specific volume request by EDGUX100 processing. The system level pools can still be used as long as SMS ACS processing does not assign a storage group and the EDGUX100 exit does not set a specific pool to be use.

Changing Pool Definitions

Use the VLPOOL command in the EDGRMM*xx* parmlib member to change pool definitions. See "Defining Pools: VLPOOL" on page 162 for information about using the VLPOOL command.

If you change a pool definition that include racks defined to an existing pool, you might need to redefine the shelf locations or volumes to DFSMSrmm. To redefine the shelf locations or volumes when the media name for the new pool does not match the existing pool, use the RMM subcommands or DFSMSrmm ISPF dialog to change information. For example, to redefine the shelf locations and volumes, perform the following tasks:

1. Add or alter the VLPOOL statements in parmlib and refresh the DFSMSrmm parameters.
2. Add rack numbers for the new VLPOOL prefixes and those with changed MEDIANAMEs.
3. Issue the RMM CHANGEVOLUME volser MEDIANAME for each volume affected.

4. Delete the old rack numbers specifying the old MEDIANAME.

See *z/OS DFSMSrmm Guide and Reference* for more information about using the RMM subcommands or the DFSMSrmm ISPF dialog to change the pool definitions.

Designing Rack Pools

If you have several types of media, define pools based on type of media. For example, you can separate your reels and cartridges from each other by placing them in separate pools.

The volumes in these pools have no relationship to any specific device type. Although you can use device type names rather than media type names to define these pools, DFSMSrmm does not associate device type names with devices.

You might want to provide a pool of scratch volumes for a particular group of users or application. With EDGUX100, you can use an exit-selected scratch pool to satisfy non-specific mount requests.

Users can also obtain volumes from a pool using the DFSMSrmm ISPF dialog or the RMM GETVOLUME subcommand. When you use the GETVOLUME subcommand, all tape mounts are specific mounts, which means that you cannot use a scratch pool and cartridge loaders.

If you have any volumes that do not meet your chosen volume naming conventions, or have a number of volumes that regularly enter and leave the removable media library, define a pool of empty shelf locations in which you can store these volumes.

When volumes arrive at the installation, define the volume to DFSMSrmm using the chosen pool ID. DFSMSrmm assigns the next available empty shelf location to the volume. Place an external label specifying the rack number on the volume to identify it.

If your removable media library is split across rooms or sites, define pools that allow easy segregation and identification of volumes based on shelf location number.

Designing Scratch Pools

You can define scratch pools based on the system where the volumes are used. For example, define a pool prefix A* to cover all volumes in a pool reserved for system MVSA. Define a pool prefix VM* for a pool reserved for volumes used on a VM system.

When you define scratch pools by system, you can only use scratch volumes from those pools on the systems specified for them. If you try to mount a scratch volume on a different system than the one with which it is associated, DFSMSrmm rejects the volume. If a system has no pool defined for it, the system accepts all volumes except those you have defined to a specific system. Make your scratch volumes available for use on all systems if you do not want these limitations.

Recommendation: Use one scratch pool for the entire library. You can use a VLPOOL command similar to the default, shown in Figure 25 on page 67. When defining scratch pools, do not use no label tapes in the pool.

You can define multiple pools for each system by associating the pools with a system using the DFSMSrmm EDGRMM*xx* parmlib VLPOOL command with the SYSID operand as described in “Defining Pools: VLPOOL” on page 162.

DFSMSrmm accepts a volume from any of the pools defined to that system. DFSMSrmm updates mount messages and drive displays to indicate the pool from which the scratch volume should be pulled. The pools are searched in alphabetical order and the first suitable pool encountered is added to mount and fetch messages. You can assign the same name to multiple pools to make it easier to satisfy mount requests.

When you define multiple scratch pools, DFSMSrmm does not use the media name when selecting the pool to use. If you are not using system based scratch pools, the operator can mount any volume. Operators use local conventions and operator procedures to select a scratch volume. For example, the operator should select a reel tape for a mount on a 3420 drive. For a 3490 drive, the operator should select a cartridge tape. In most cases, scratch tapes are already pulled and are available for immediate mounting.

Within the pool for use on a VM system, you can define each volume for use only on VM, or for use on MVS and VM. DFSMSrmm uses this information to reject non-MVS volumes on MVS. You can use VM volumes on VM systems.

You can design pools at the user, group, or application level by using DFSMSrmm calls to SMS ACS to assign storage group names, or you can make use of the EDGUX100 installation exit to control scratch pool selection. See “Using SMS Tape Storage Groups for DFSMSrmm Scratch Pooling” on page 71 and “Using the DFSMSrmm EDGUX100 Installation Exit” on page 221 for more information about using pool names and implementing exit-selected scratch pools.

Requesting and Using Scratch Pools

DFSMSrmm system-based scratch pooling is always available for you to use, however, you cannot use it for automated system-managed tape. To use exit-selected scratch pooling, install the sample EDGUX100 installation exit on your system and customize the installation exit EDGUX100 to set a specific pool for a non-specific volume request. See Chapter 11, “Using DFSMSrmm Installation Exits,” on page 221 for information about how to use the EDGUX100 installation exit.

DFSMSrmm updates the mount message and drive display with the appropriate pool prefix, pool name, or storage group name. DFSMSrmm validates the mounted scratch volume against the selected scratch pool prefix, pool name, or storage group during OPEN processing.

You can set up DFSMSrmm to satisfy each request for a new scratch tape in one of the following ways:

- Set up a default pool using the EDGRMM*xx* parmlib OPTION VLPOOL command. DFSMSrmm uses the VLPOOL definitions to select a default DFSMSrmm pool by using DFSMSrmm system-based pooling when you have not set up other pools. DFSMSrmm sets a pool prefix and a pool name if a specific pool is selected. Set up DFSMSrmm system-based scratch pooling using the DFSMSrmm VLPOOL command with the SYSID operand. This method allows you to control which scratch pools can be used based on the system on which the volumes are used. This enforces pooling based on VLPOOL prefix pools. DFSMSrmm uses the DFSMSrmm system-based scratch pooling when you do not select any other method.
- Use ACS routines for scratch pooling based on tape storage group names. Using ACS processing to set a storage group name overrides all other pool selection methods. DFSMSrmm provides support for non-system-managed tape and for

system-managed manual tape libraries. This support enables pooling at the individual volume level. You assign a storage group name to each volume by using DFSMSrmm TSO subcommands or by using pooling information that you define with the DFSMSrmm EDGRMM*xx* parmlib VLPOOL command. See “Using SMS Tape Storage Groups for DFSMSrmm Scratch Pooling”. DFSMSrmm calls ACS routines passing environment information, including the pool identified by DFSMSrmm system-based pooling. The ACS routine can optionally set a storage group name, which overrides the DFSMSrmm system-based pool.

- Customize the DFSMSrmm EDGUX100 installation exit that is described in Chapter 11, “Using DFSMSrmm Installation Exits,” on page 221 to select a pool prefix. You can use information in the system to determine which pool to use. DFSMSrmm calls the EDGUX100 installation exit if ACS processing does not return a storage group name. The EDGUX100 installation exit can return a pool prefix value that overrides the DFSMSrmm system-based pool. The pool name is set based on the selected pool prefix.
- Use pre-ACS processing to obtain the DFSMSrmm system-based pool or the EDGUX100 installation exit pool prefix as an input value to the ACS routines in the MSPOOL read-only variable. During pre-ACS processing, DFSMSrmm does not make the RMMPOOL environment call to the ACS routine. During pre-ACS processing for new allocations:
  - DFSMSrmm uses the VLPOOL definitions to select a default DFSMSrmm pool using DFSMSrmm system-based pooling. DFSMSrmm sets a pool prefix if a specific pool is selected.
  - DFSMSrmm calls the EDGUX100 installation exit to obtain a pool prefix value. If a pool prefix value is returned, the pool prefix value returned by the EDGUX100 installation exit overrides the DFSMSrmm selected pool.
  - DFSMSrmm returns the selected value in the MSPOOL read-only variable if the MSPOOL variable is not already set by the pre-ACS exit.

Using SMS Tape Storage Groups for DFSMSrmm Scratch Pooling

This section describes how to use storage groups for DFSMSrmm scratch pooling.

Before you begin: To use SMS tape storage groups, first define tape storage groups using ISMF. To define tape storage groups, you must have at least one system-managed tape library defined to ISMF. The library can be an automated tape library or a manual tape library. If you do not have a system-managed tape library or do not want to associate the new SMS tape storage groups with an existing library, define a new dummy library to ISMF. Use the ISMF define library application to define a system-managed tape library. Use a dummy library ID to identify the library and use appropriate values to complete the other data fields required for a tape library definition. When you define a dummy library, OAM issues the CBR3006I and CBR3002E messages at startup time. Ignore these messages when they are issued for your dummy library. Once you have defined the dummy library, you do not need to start OAM or define the OAM1 subsystem to support this dummy system-managed library.

Making an ACS Storage Group Assignment

To assign a storage group, you must have the SMS subsystem active and have a valid SMS configuration.

You use the ACS routines to process the special calls that DFSMSrmm makes to the SMS subsystem for ACS processing. DFSMSrmm requests that the data class, management class, and storage group routines are run. The environment variable is set to *RMMPOOL* so that you can differentiate allocation requests for system-managed data sets from requests by DFSMSrmm for a storage group name. When DFSMSrmm calls the ACS routines with the &ACSENVIR variable set to either RMMPOOL or RMMVRS, the storage class (&STORCLAS variable) is set to the word USERMM.

Table 12 lists the read-only variables that are set for DFSMSrmm requests:

Table 12. SMS Read-only Variables Set by DFSMSrmm

&ACCT_JOB	&ACCT_STEP	&ACSENVIR	&DD
&DSN	&DSORG	&DSTYPE	&EXPDT
&FILENUM	&GROUP	&HLQ	&JOB
&LABEL	&LIBNAME	&LLQ	&NQUAL
&PGM	&STORGRP	&SYSNAME	&SYSPLEX
&UNIT	&USER	&XMODE	

The &DD, &PGM, &EXPDT, and &FILENUM variables are optional for JES3 mount messages. &LIBNAME is optional for JES3 fetch messages.

1. Define your pools by using the DFSMSrmm EDGRMM*xx* parmlib VLPOOL command as shown in Figure 26. See "Defining Pools: VLPOOL" on page 162 for information about the VLPOOL command.

```
 ...
VLPOOL PREFIX(123*)   NAME(POOLA) MEDIANAME(CARTS) -
    DESCRIPTION('PART OF POOL A')
VLPOOL PREFIX(99975*) NAME(POOLA) MEDIANAME(REELS) -
    DESCRIPTION('PART OF POOL A')
VLPOOL PREFIX(C0*)    NAME(TEMP) MEDIANAME(CARTS) -
    DESCRIPTION('TEMPORARY POOL')
VLPOOL PREFIX(*)      NAME(DEFAULT) MEDIANAME(CARTS) -
    DESCRIPTION('DEFAULT POOL')
 ...
```

Figure 26. Defining Pools with the DFSMSrmm EDGRMMxx Parmlib VLPOOL Command

2. Code your selection criteria and rules into the management class and storage group ACS routines as shown in Figure 27 on page 73.

```
PROC 1 MGMTCLAS
 ...
 /**************************************************/
 /*   RMM POOLING FILTER LISTS                     */
 /**************************************************/
 ...
FILTLIST POOLA INCLUDE(USERA.**,USERY.KEEP.*) -
                EXCLUDE(USERA.POOLB.**)
 ...
 IF &ACSENVIR = 'RMMPOOL' THEN
 /**************************************************/
 /*   RMM POOLING DECISIONS                        */
 /**************************************************/
   DO
     SELECT (&DSN)
       WHEN (&POOLA) SET &MGMTCLAS = 'POOLA'
       OTHERWISE     SET &MGMTCLAS = 'DEFAULT'
     END
     /* ALLOCATE TEMP POOL FOR TEMPORARY DATA SETS  */
     IF &DSTYPE = TEMP THEN SET &MGMTCLAS = TEMP
 ...
   END
 ...
END
```

Figure 27. Sample Management Class Routine

If your ACS processing does not set the storage group name, DFSMSrmm processing continues without using storage group names.

The processing that the STORGRP ACS routine performs is limited to the variables it can reference. You might want to make your decisions in the MGMTCLAS ACS routine and use the STORGRP routine to set the STORGRP variable to the selected value as shown in Figure 28.

Any management class value that you set during ACS processing for the RMMPOOL ACS environment call is ignored by DFSMSrmm processing.

```
PROC 1 STORGRP
 ...
 IF &ACSENVIR = 'RMMPOOL' THEN
 /**************************************************/
 /*   RMM POOLING DECISIONS                        */
 /**************************************************/
   DO
     SELECT (&MGMTCLAS)
       WHEN ('POOLA')   SET &STORGRP = 'POOLA'
       WHEN ('DEFAULT') SET &STORGRP = 'DEFAULT'
       WHEN ('TEMP')    SET &STORGRP = 'TEMP'
     END
 ...
   END
 ...
 END
```

Figure 28. Storage Group Routine Sample

A Pooling Example

This example describes how pooling might be implemented for an installation that has:

- An automated tape library and no non-system-managed drives.
- A manual tape library.

- Ranges of scratch tapes using a numbering system with consecutive numbers for volumes spread across a few well identified ranges.

  Volume serial numbers, 210000 - 219999, 710000-719999, and 800000 - 805999.
- Foreign tapes from IBM and other vendors with standard labels and random volume serial numbers that use alphanumeric, national, or special character prefixes, such as DZ1100 or DP3100.
- Tapes sent from another site for input processing. The other site uses a range of tapes numbered 401000 - 401999. The tapes are scratch at the other site, but are private volumes here.

The installation defines pools using the VLPOOL command as shown in Figure 29:

```
VLPOOL PREFIX(21*) TYPE(S) MEDIANAME(CART) RACF(Y) EXPDTCHECK(N) -
   DESCRIPTION('scratch pool of cartridge system tape')
VLPOOL PREFIX(71*) TYPE(S) MEDIANAME(CART) RACF(Y) EXPDTCHECK(N)-
   DESCRIPTION('scratch pool manual tape library') -
   NAME(SGMTL01)
 VLPOOL PREFIX(80*) TYPE(S) MEDIANAME(CART) RACF(Y) EXPDTCHECK(N)-
   DESCRIPTION('scratch pool enhanced capacity cartridge') -
   NAME(TWOTONE)
VLPOOL PREFIX(401*) TYPE(R) MEDIANAME(CART) RACF(Y) EXPDTCHECK(Y) -
   NAME(SITE2) -
   DESCRIPTION('tapes sent from site 2')
VLPOOL PREFIX(*) TYPE(R) MEDIANAME(CART) RACF(Y) EXPDTCHECK(Y) -
   NAME(SITE2) -
   DESCRIPTION('miscellaneous check-in and check-out tapes')
```

Figure 29. Defining Pools with VLPOOL Commands

The installation can now define volumes in the 21*, 71* and 80* pools which DFSMSrmm manages as scratch pools. The storage group SGMTL01 used as a VLPOOL NAME must be defined as a valid storage group name to SMS. The installation can also define some empty racks for the 401* volumes. Each time a foreign volume comes in, it is defined to DFSMSrmm, with RLSE(RETURN) RETPD(*nn*), labeled with a sticky label, and entered into the system-managed tape library.

RACF and EXPDTCHECK provide the following:

- RACF(Y) is specified for all pools to ensure that tape volume security is managed by DFSMSrmm as described in Chapter 8, “Using the Parmlib Member EDGRMMxx,” on page 127.
- EXPDTCHECK(N) is used for scratch pools, so DFSMSrmm automatically replies to any IEC507D expiration date protection messages allowing scratch volumes to be reused. DFSMSrmm does not reply to the IEC507D message issued for tape volumes when you use the EDGUX100 installation exit to ignore a volume.
- EXPDTCHECK(Y) is used for all non-scratch volumes to ensure that DFSMSrmm automatically replies to IEC507D to prevent overwrite of expiration date protected data.

For the remainder of the volumes, when a volume comes in, the installation defines a rack number that matches the VOL1 label, and then defines the volume using RLSE(RETURN) RETPD(nnn). This approach should be used since the external and internal volume serial number must be the same.

In this example, all scratch volumes are system-managed. DFSMSrmm validates scratch pools for requests in the manual system-managed library, but does not validate scratch pools for automated tape libraries. DFSMSrmm always ensures that a scratch volume is mounted in response to a non-specific mount request. Defining a rack pool is an alternative to using a scratch pool as described here. The rack pool should have a finite size based on known requirements, and accommodate any volume serial number as long as it does not conflict with the installation's naming conventions.

Anytime the installation wants to have more scratch volumes, existing pools can be made larger or new pools reserving a new range of numbers can be defined.

This approach for system-managed libraries differs from the way pools might be used with a non-system-managed library. With a non-system-managed library, you need to have physical shelving. You also have the option of using different internal to external labels.

For installations with mixed environments, a combination of these approaches provides a workable solution.

Managing Pools with Job Name and Data Set Name

You can manage scratch tape pools based on job names and data set names.

Recommendation: Use SMS ACS Storage Group assignment to manage scratch pooling with jobname and data set name for system-managed volumes. For information, see "Making an ACS Storage Group Assignment" on page 71. For non-system-managed volumes, use the DFSMSrmm EDGUX100 installation exit. See Chapter 11, "Using DFSMSrmm Installation Exits," on page 221 for information about using the EDGUX100 installation exit.

Use the DFSMSrmm installation exit EDGUX100 to select the correct scratch pool for non-specific requests for non-system-managed volumes and to decide whether the cartridge loader should be used. You can also use EDGUX100 with criteria other than job names and data set names, but you have to modify the supplied EDGUX100 installation exit in order to do this.

Assigning Policies

DFSMSrmm provides these methods for the assignment of retention and management policies:

1. Use ACS routines to assign a management class name that matches a DFSMSrmm data set vital record specification. DFSMSrmm calls ACS routines directly to provide support for non-system-managed tape. This DFSMSrmm processing enables you to use SMS management class to manage system-managed and non-system-managed tape data.
2. Define DFSMSrmm vital record specifications with data set name masks to identify the retention and movement policies.
3. Define a vital record specification management value that DFSMSrmm can use to apply retention and movement policies to datasets. Use the DFSMSrmm EDGUX100 installation exit described in Chapter 11, "Using DFSMSrmm Installation Exits," on page 221 to select a vital record specification management value. You can use information in the system to determine which vital record specification management value to use.
4. During pre-ACS processing, use the DFSMSrmm EDGUX100 installation exit to obtain information that is used by the ACS routines. The EDGUX100 installation exit can provide a vital record specification management value in the

MSPOLICY read-only variable, which is used by the ACS routines. During pre-ACS processing, DFSMSrmm does not make the RMMVRS environment call to the ACS routine.

Using SMS ACS processing you can consolidate policy assignment decisions in a single place whether you use system-managed tape or not. You can use SMS ACS routines to assign management class for your data sets instead of using EDGUX100 exit assigned vital record specification management values. You can assign a management policy by name to either a non-system-managed or a system-managed tape data set.

For non-system-managed tape, DFSMSrmm calls the ACS routines to obtain a management class. The management class is used in place of the vital record specification management value assigned by the DFSMSrmm EDGUX100 installation exit. When a management class name is assigned using ACS routines, the EDGUX100 installation exit is not called for a vital record specification management value. The decision to call the EDGUX100 installation exit is made each time a new data set is created on a tape based on whether a construct is assigned by ACS processing. You have the flexibility to identify one request to be handled by ACS and the next request to be handled by the EDGUX100 installation exit. For compatibility, DFSMSrmm passes the vital record specification management value that is determined by the EDGUX100 installation exit by using the pre-ACS interface in the MSPOLICY variable. You might use the vital record specification management value in the MSPOLICY variable as the base for MC assignment for system-managed tape. Even when you use SMS ACS support to assign management class names you can have separate policies for retention and movement by using a primary data set name vital record specification and a secondary management class vital record specification.

Use the ACS routine to assign the management class as the secondary vital record specification and the DFSMSrmm data set name vital record specification to assign the primary vital record specification. You must specify the EDGRMMxx parmlib OPTION VRSEL(NEW) operand to enable this support.

You can still use the EDGUX100 installation exit to check for either EXPDT= or ACCODE= specifying special values and override them to ensure correct retention processing by DFSMSrmm.

Using SMS Management Class to Retain Non-System-Managed Volumes

The DFSMSrmm EDGRMM*xx* parmlib OPTION VRSEL operand determines how DFSMSrmm assigns retention and movement policies. When VRSEL(NEW) is used, you can define policies using a vital record specification data set name mask, and a management class, or a vital record specification management value can also be assigned to manage a data set.

For each new tape data set, DFSMSrmm performs the following tasks:

1. During pre-ACS processing for new tape data sets:
   a. DFSMSrmm calls the EDGUX100 installation exit for a vital record specification management value. The vital record specification management value provides management policy and retention policy information for datasets.
   b. DFSMSrmm returns the selected value in the MSPOLICY read-only variable.
2. During OPEN processing:

a. DFSMSrmm calls ACS routines and passes environment information. Your ACS routine can set a management class name.
b. If ACS processing does not return a management class name, DFSMSrmm calls the EDGUX100 installation exit to obtain a vital record specification management value.
c. At OPEN time, DFSMSrmm records the specified management class or vital record specification management value at the data set level.

Making An ACS Management Class Assignment

To assign a management class, you must have the SMS subsystem active and have a valid SMS configuration.

You use the ACS routines to process the special calls that DFSMSrmm makes to the SMS subsystem for ACS processing. DFSMSrmm requests that the management class routine is run. The environment variable is set to RMMVRS so that you can differentiate allocation requests for system-managed data sets from requests by DFSMSrmm for a management class name. When DFSMSrmm calls the ACS routines with the &ACSENVIR variable set to either RMMPOOL or RMMVRS, the storage class (&STORCLAS variable) is set to the word USERMM.

Table 13 lists the read-only variables that are set for DFSMSrmm requests:

Table 13. SMS Read-only Variables Set by DFSMSrmm

&ACCT_JOB	&ACCT_STEP	&ACSENVIR	&DD
&DSN	&DSORG	&DSTYPE	&EXPDT
&FILENUM	&GROUP	&HLQ	&JOB
&LABEL	&LIBNAME	&LLQ	&NQUAL
&PGM	&STORGRP	&SYSNAME	&SYSPLEX
&UNIT	&USER	&XMODE	

1. Define vital record specifications for the management class names you plan to use as shown in Figure 30.

```
 ...
RMM ADDVRS DSNAME('CATALOG') WHILECATALOG
RMM ADDVRS DSNAME('ONSITE') DAYS COUNT(31)
RMM ADDVRS DSNAME('OFFSITE') CYCLES COUNT(5) LOCATION(STORE1) -
    NEXTVRS(DAYS90)
RMM ADDVRS NAME(DAYS90) DAYS COUNT(90)
 ...
```

Figure 30. Defining Vital Record Specifications for Non-System-Managed Volumes

2. You code your selection criteria and rules into the management class ACS routine as shown in Figure 31 on page 78.

```
PROC 1 MGMTCLAS
 ...
 /***************************************************/
 /*   RMM POLICY FILTER LISTS                       */
 /***************************************************/
 ...
 FILTLIST POOLA INCLUDE(USERA.**,USERY.KEEP.*) -
               EXCLUDE(USERA.POOLB.**)
 ...
  IF &ACSENVIR = 'RMMVRS' THEN
 /***************************************************/
 /*   RMM VRS ASSIGNMENT DECISIONS                  */
 /***************************************************/
   DO
     SELECT (&DSN)
       WHEN (&OFFSITE) SET &MGMTCLAS = 'OFFSITE'
       WHEN (&ONSITE)  SET &MGMTCLAS = 'ONSITE'
     END
     /* CATCH EXPDT=99000                           */
     IF &EXPDT = '1999000' THEN SET &MGMTCLAS = 'CATALOG'
 ...
   END
 ...
END
```

Figure 31. Sample Management Class Routine for Managing Non-System-Managed Volumes

If ACS processing does not set the management class name, DFSMSrmm processing continues without using management class names.

When your ACS routine sets a management class, no call is made to the EDGUX100 installation exit to obtain a vital record specification management value. However, EDGUX100 is still called to update the DFSMSrmm copy of the JFCB expiration date if necessary.

Managing Volumes with Special Dates

If your current tape management system allows special expiration dates (such as EXPDT=99000) in JCL to identify management and retention requirements for system-managed volumes, you can manage those volumes by using management class to handle the special dates.

You can also use vital record specification management values to manage both system-managed volumes and non-system-managed tape volumes with JCL expiration dates that are specified with special dates. A vital record specification management value is a one-to-eight character name defined by your installation and used to assign management and retention values to tape data sets.

You define the management policies for management classes and for vital record specification management values using vital record specifications.

Use the DFSMSrmm EDGUX100 installation exit as described in Chapter 11, “Using DFSMSrmm Installation Exits,” on page 221 to assign vital record specification management values to new tape data sets.

During inventory management VRSEL processing, DFSMSrmm uses the vital record specification management value as determined by the VRSEL parmlib option.

Using Volumes with Special Expiration Dates

If you have volumes with JCL expiration dates specified with special dates, you can use DFSMSrmm to manage those volumes. You use management classes for system-managed volumes. You use vital record specification management values for both system-managed and non-system-managed tape volumes.

For system-managed tape volumes, perform these tasks:

- Define management classes to cover all the special dates used by your installation.
- Update your ACS routine to assign those management classes to data sets on tape.
- Define vital record specifications identifying the management policies for management classes.
- Run DFSMSrmm inventory management vital record processing to identify the volumes that should be retained based on the management classes that you define.

For both system-managed and non-system-managed tape volumes, perform these tasks:

- Define vital record specification management values to cover all the special dates used by your installation.
- Tailor the EDGUX100 installation exit to assign the vital record specification management values you define to new tape data sets.
- Define vital record specifications identifying the management policies for vital record specification management values.
- Run DFSMSrmm inventory management vital record processing to identify the volumes that should be retained based on the vital record specification management values that you define.

See “Managing Volumes with Special Dates” on page 78 for more information.

Using Management Class to Retain System-Managed Volumes

This section presents the steps you need to take to use management class to retain system-managed volumes.

Step 1: Define Management Class Names

You need to define management class names that cover all the special dates used by your installation. For example, you can define a management class, M99000, for the special date 99000 and then define a vital record specification using the data set name M99000.

The management class name can be one to eight alphanumeric characters and must begin with an alphabetic character to be acceptable to the MVS data set naming standards that apply to DFSMSrmm vital record specifications.

Step 2: Update ACS Routine

Update your ACS routine to use an appropriate filter list and logic to assign management classes to data sets on tape. When a volume is opened, DFSMSrmm records the management class name you assign to new tape data sets using your ACS routine.

Figure 32 on page 80 shows a management class ACS routine that maps the expiration date 99000 to a management class name M99000, and the expiration dates 99001 through 99009 to the management class name M99009.

```
PROC 1 &MGMTCLAS
/********************************************************/
/*  DEFINE FILTER FOR KEEP WHILE CATALOGED  EXPDT=99000 */
/********************************************************/
  FILTLIST SPECIAL_DATE9 INCLUDE('1999000')
/********************************************************/
/*  DEFINE FILTER FOR UP TO 9 CYCLES                    */
/********************************************************/
  FILTLIST SPECIAL_DATEC9 INCLUDE(199900*)
/********************************************************/
/*  DEFINE FILTER FOR TAPE UNITS                        */
/********************************************************/
  FILTLIST TAPEUNITS INCLUDE(TAPE*,T3420*,T3480*,T3490*,
                             '3420','3480','3490',
                             T3590*,'3590')
/********************************************************/
/*  SELECT VALID MANAGEMENT CLASS                       */
/********************************************************/
  SELECT
    WHEN (&EXPDT = &SPECIAL_DATE9 && &UNIT = &TAPEUNITS)
      SET &MGMTCLAS = 'M99000'
    WHEN (&EXPDT = &SPECIAL_DATEC9 && &UNIT = &TAPEUNITS)
      SET &MGMTCLAS = 'M99009'
    OTHERWISE
      SET &MGMTCLAS = 'NORMAL'
  END
END
```

Figure 32. Assigning Management Class to Data Sets on Tape

Step 3: Define Retention Policies for Management Class Names

You define policies for management classes by defining vital record specifications. Use the RMM subcommand ADDVRS with the DSNAME operand, or the DFSMSrmm Add Data Set VRS panel in the DFSMSrmm ISPF dialog to define vital record specifications. Use the data set name masks that match the management class names you have defined. See *z/OS DFSMSrmm Guide and Reference* for more information on defining vital record specifications.

Issue the command shown in Figure 33 to define a vital record specification that manages the special date 99000.

```
RMM ADDVRS DSNAME('M99000') WHILECATALOG
```

Figure 33. Special Date 99000 Vital Record Specification

You can define a vital record specification data set name mask that matches multiple management class names. For example you could define a data set name mask of M99* and use this mask to cover several management classes. Because the RMM ADDVRS subcommand with DSNAME operand can also be used to define a vital record specification for an individual data set, the management class name mask you specify for ADDVRS DSNAME is first used to match a data set name before any other mask.

Figure 34 on page 81 uses a data set name mask to define a vital record specification to manage any management class starting with M99. For example, with this data set name mask you could manage special dates in the range 99001 through 99365.

```
RMM ADDVRS DSNAME('M99*')
```

Figure 34. Managing Management Classes Using a Data Set Name Mask

Step 4: Run DFSMSrmm Inventory Management Vital Record Processing

Run DFSMSrmm inventory management vital record processing to identify the volumes that should be retained based on the management classes you have defined. DFSMSrmm uses the management class assigned to the data set, rather than the data set name to select the appropriate vital record specifications. See “How Vital Record Processing Works” on page 295 for information about vital record processing.

Using the SMS Pre-ACS Interface

Using the SMS pre-ACS calls to DFSMSrmm, you can use any existing DFSMSrmm and EDGUX100 exit scratch pool and EDGUX100 exit policy assignment decisions as input to your SMS ACS logic to enable you to direct new data set allocations to the correct media. You can use SMS ACS to decide if data sets are to be system-managed or not system-managed. For system-managed data sets, you can decide if they are to be DASD or tape. For system-managed tape data sets, you can decide to which library or library type you would like the allocation directed. This can help you with VTS implementation.

To use DFSMSrmm and EDGUX100 decisions within your ACS processing, use the pre-ACS processing. Implement or customize the sample EDGUX100 exit (or the EDGCVRSX sample of EDGUX100) to feed pool and VRS management value information into ACS. Base your ACS processing decisions on the MSPOOL and MSPOLICY values that come from DFSMSrmm scratch pool processing and EDGUX100 exit processing. You can plan to move the EDGUX100 exit decisions into your SMS ACS routines to enable the EDGUX100 exit processing to be ignored or removed some time in the future.

DFSMSrmm always attempts to pass values for the MSPOOL and MSPOLICY ACS read-only variables. If you do not use an EDGUX100 exit the MSPOOL variable is set to the DFSMSrmm system-based scratch pool decision. If you do not use the EDGUX100 exit, there is never a value set for MSPOLICY. DFSMSrmm sets a pre-ACS variable only if the variable has not yet been set using the pre-ACS exit. Since DFSMSrmm gets control after the installation exit, any vendor decisions or any customer decisions take precedence.

Managing Volumes with Duplicate Volume Serial Numbers

In DFSMSrmm, a duplicate volume has these attributes:

- Is defined to DFSMSrmm with a unique external volume serial number.
- Has a VOL1 label.
- The VOL1 label does not match its own external volume serial number.

You can manage volumes with duplicate volume serial numbers by performing these tasks:

- Define the volumes with duplicate volume serial numbers so that you can use them without changing the volume serial number as described in “Using Volumes with Duplicate Volume Serial Numbers” on page 82.

- Change the volume serial numbers as described in “Changing Duplicate Volume Serial Numbers” on page 83.
- Use the volumes with duplicate volume serial numbers and request that they not be managed by DFSMSrmm. See “Using EDGUX100 to Ignore Duplicate or Undefined Volume Serial Numbers” on page 225 for information about using EDGUX100 to ignore duplicate volume serial numbers.

Using Volumes with Duplicate Volume Serial Numbers

Define a VLPOOL rack pool to reserve shelf space for duplicate volumes. You can use the next available empty rack number as the unique volume serial number for a new duplicate volume. For example, define a VLPOOL with the PREFIX(D*) to manage duplicate volumes. Volume ABC001 is a duplicate volume because it has standard labels that contain the volser ABC001 and it matches the volume ABC001 which is defined to DFSMSrmm. You can define ABC001 to DFSMSrmm as a duplicate volume by using a unique external volume serial number.

Use the RMM VOLUME subcommands with the VOL1 operand to define a unique volume serial number for each volume that has a duplicate VOL1 volume serial number. Change the external label, and optionally the barcode, to match the new unique volume serial number. In this example, the next available rack in the D* pool is D00010. Use the D00010 value to add the duplicate volume.

Example:

```
RMM ADDVOLUME D00010 STATUS(MASTER) VOL1(ABC001)
```

When you use the DFSMSrmm ISPF dialog to define volumes, DFSMSrmm automatically detects when you have added a volume that duplicates a volume that was already defined to DFSMSrmm. The dialog prompts you with options so that you can add the new volume as a duplicate, or change the existing volume to a duplicate and add the new volume using its existing name.

Example: Obtain information about duplicate volumes defined to DFSMSrmm by issuing the RMM SEARCHVOLUME TSO subcommand. You can also use the DFSMSrmm ISPF dialog by selecting the VOL1 field of the Search Volume panel.

```
RMM SEARCHVOLUME VOLUME(*) VOL1(*) LIMIT(*) OWNER(*)
```

To label a volume as a duplicate volume, use the EDGINERS utility manual processing with the VOL1 operand on the command in the SYSIN file.Then you can label the volume. If DFSMSrmm already knows the correct VOL1 label volume serial number, you do not need the VOL1 operand on the EDGINERS INIT command. Specify the VOL1 operand on the INIT command to override the value recorded in the DFSMSrmm control data set. DFSMSrmm writes this value to the tape label.

Example: Label a volume as a duplicate volume.

```
//MANUAL   EXEC PGM=EDGINERS
//SYSPRINT DD SYSOUT=*
//TAPE     DD UNIT=(tape,,DEFER)
//SYSIN    DD *
INIT VOL(D00001) LABEL(SL) VOL1(ABC123)
/*
```

If the volume is not defined to DFSMSrmm, specify the VOL1 operand to label the volume with a value different from the volume serial number used to define the volume to DFSMSrmm. Add the volume as a MASTER volume rather than a SCRATCH volume. The following example shows how to label a duplicate volume using the EDGINERS utility.

Example: Define a volume to DFSMSrmm with the external volume serial number D00001 and label it with the VOL1 label volume serial number ABC123.

```
INIT VOLUME(ABC123,D00001) VOL1(ABC123) LABEL(SL) POOL(F*) -
   MEDIANAME(3590)
```

When you use EDGINERS automatic processing, the new volume label is written so that the VOL1 label volume serial number matches the volume serial number and the VOL1 label volume serial number value recorded by DFSMSrmm is cleared.

Changing Duplicate Volume Serial Numbers

If you want DFSMSrmm to manage volumes with duplicate volume serial numbers and you cannot remove one of the duplicate volumes serial numbers, you must change volume serial numbers.

Add the duplicate volume serial number, as follows:

- For volumes with no label (NL), file the duplicate volume under a different unique volume serial number. Assign the volume a volume serial number equal to the shelf location, change its external label, and inform the owners of the change.
- For IBM standard label (SL) volumes and ISO/ANSI label (AL) volumes, move all data sets on the volume to a different, unique volume. Change the external labels and RACF profiles for the new volume. Inform the owners of the change. The owners must catalog the data sets on the new volume if the data sets were cataloged on the original volume.

  You might need to perform this step if you have received volumes that contain software products.

Example: Set the initialize action for a volume that is defined to DFSMSrmm.

```
RMM CHANGEVOLUME D00001 INIT(Y)
```

Managing Undefined Volume Serial Numbers

DFSMSrmm does not manage volumes that are not defined to DFSMSrmm. These types of volumes are also known as foreign tape volumes. DFSMSrmm always rejects foreign volumes when they are mounted in response to a request for a scratch tape. To control the use of foreign tapes for non-scratch requests, you can set up DFSMSrmm so that DFSMSrmm rejects foreign volumes when they are opened. Specify the REJECT ANYUSE(*) and the REJECT OUTPUT(*) commands in the DFSMSrmm EDGRMM*xx* parmlib member as described in “Defining Tapes Not Available on Systems: REJECT” on page 157. You must use the prefix value ’*’.

To prevent DFSMSrmm from rejecting undefined or foreign volumes, use the DFSMSrmm EDGUX100 installation exit to request that DFSMSrmm ignore the undefined volumes. When DFSMSrmm ignores a volume DFSMSrmm does not perform the following tasks:

- Record information about the volume in the DFSMSrmm control data set.
- Validate that the correct volume has been mounted.
- Perform management functions for the volume.

See “Using EDGUX100 to Ignore Duplicate or Undefined Volume Serial Numbers” on page 225 for information about using EDGUX100 to ignore duplicate volume serial numbers.

Segregating WORM tapes in separate scratch pools

When you use scratch pooling for non-system-managed tape or use an manual tape library, ensure that WORM tapes are in a separate scratch pool. If you mix WORM tape and non-WORM tape in a scratch pool, you cannot control the type of tape that will be mounted for a non-specific volume request.

Chapter 5. Running DFSMSrmm with System-Managed Tape Libraries

DFSMSrmm provides the following support for system-managed tape libraries:
- Cartridge entry processing
- Cartridge eject processing
- Expiration management
- Volume-not-in-library support
- Movement between libraries and storage locations
- Partitioning the libraries with other systems
- VTS import/export processing
- Tracking and managing logical, physical, and stacked volumes

If you have volumes in an automated tape library, you need to reserve shelf space outside the automated tape library if you plan to eject the volumes and retain them in a manual tape library or non-system-managed tape library. Volumes in a manual tape library can be logically ejected, left in the same shelf location and used as non-system-managed volumes.

DFSMSrmm identifies all volumes including volumes that reside in system-managed tape libraries by using the volume serial number and the rack number.
- The volume serial number is the internal volume serial number recorded in VOL1 label on SL and AL type labels.
- The rack number is the external volume serial number

For volumes that reside in system-managed tape libraries, a rack number is optional. If you specify a rack number for a volume residing in a system-managed tape library, the volume serial number and the rack number must be the same. You cannot change the rack number for volumes that are associated with system-managed libraries. You cannot define a rack number for logical volumes.

You can manage volume movement between system-managed libraries and storage locations, between system-managed libraries, and between non-system managed libraries and system-managed libraries by defining vital record specifications.

This topic describes how to use DFSMSrmm with system-managed tape volumes. Refer to the EDGRMM*xx* OPTION command SMSTAPE operand described in "Defining System Options: OPTION" on page 134 for details about controlling the functions that are provided by DFSMSrmm.

Using DFSMSrmm with System-Managed Tape Libraries

DFSMSrmm automatically adds volumes to the DFSMSrmm control data set during entry processing and when you use volumes. If a volume defined to DFSMSrmm is entered into a system-managed tape library without using the RMM CHANGEVOLUME command to set the volume location to the library name, DFSMSrmm updates the DFSMSrmm control data set volume record with the library name, library type, and volume entry status. DFSMSrmm does not update the home location name for the volume.

If the volume is not defined to DFSMSrmm and is entered into an automated tape library, DFSMSrmm automatically defines the volume in the DFSMSrmm control data set with information it obtains from the OAM entry user exit parameter list as follows: LOCATION, STATUS, STORGRP, MEDIATYPE, RECORDINGFORMAT, SPECIALATTRIBUTES, COMPACTION, READDATE, WRITEDATE, EXPDT, LIBRARY NAME, TYPE, ENTRY STATUS, and HOME LOCATION. This means that all volumes entered into an automated tape library must be managed by DFSMSrmm, unless the library is to be partitioned and certain volumes left undefined. See “Partitioning System-Managed Tape Libraries” on page 109 for details on partitioning the system-managed tape library.

Recommendation: Define all volumes to DFSMSrmm before entering them into a library. Define private volumes prior to entry into the automated tape library and set the ISMF default cartridge entry status to scratch.

If you are inserting cartridges without predefining them to DFSMSrmm, and the default entry use attribute is private, then DFSMSrmm assigns a default owner. The default owner is set to the RACF User ID associated with the DFRMM started procedure name.

Associating Volumes and System-Managed Libraries

Specify a system-managed tape library name for a volume by using the DFSMSrmm ISPF dialog or the RMM ADDVOLUME or RMM CHANGEVOLUME subcommand with the LOCATION operand. Figure 35 shows how to use the ADDVOLUME subcommand to specify the name of an automated tape library, ATL1, where volume A00001 resides:

```
RMM ADDVOLUME A00001 LOCATION(ATL1)
```

Figure 35. Specifying a Library Name for a Volume

You can provide the library name even if the volume does not yet reside in a system-managed tape library. When you provide a library name, DFSMSrmm checks if the library name is already defined in the TCDB. If the library name is defined in the TCDB, processing continues. If the library name is not defined, then processing fails.

DFSMSrmm records the library name and the library type; manual tape library or automated tape library. DFSMSrmm also records whether the volume is currently in the library.

If the volume is not defined in the TCDB, and the library name specified is an automated tape library, then DFSMSrmm sets the volume destination to the library name and waits for the volume to be entered. If the volume is not defined in the TCDB, and the library name specified is a manual tape library, then DFSMSrmm uses manual cartridge entry processing so the TCDB is updated for the volume.

Cartridge Entry Processing

DFSMSrmm is called at cartridge entry time by the CBRUXENT installation exit which DFSMSrmm supplies. During cartridge entry processing in an automated tape library, the installation exit passes the external volume serial number. DFSMSrmm checks for the existence of a volume or a rack with the same number. When there is a conflict, the cartridge entry is denied. DFSMSrmm sets a return code telling OAM not to process the volume on this system under the following conditions.

- If a volume is defined to DFSMSrmm as ″not for use on MVS″.
- If a volume is not defined to DFSMSrmm and it matches the REJECT ANYUSE prefixes, and is entered into a tape library.

If a volume is entered into a library that does not match the library name already defined for it, DFSMSrmm updates the library name in the control data set. The home location name is not updated. If a volume is 'intransit' and the destination is different than the library the volume is entering, the cartridge entry is prevented.

Manual Cartridge Entry Processing

DFSMSrmm uses manual cartridge entry processing to add information about a volume residing in a manual tape library into the TCDB. The OAM macro CBRXLCS is used to get the volume information added to the TCDB when a volume is to reside in a manual tape library. The information provided by DFSMSrmm using CBRUXENT, is the same as for automated libraries, and includes: storage group name, recording format, compaction, special attributes, owner name, and expiration date.

DFSMSrmm uses the volume's media type as defined to DFSMSrmm when requesting manual cartridge entry; you must ensure that DFSMSrmm has the correct media type to avoid allocation and mount problems when the volumes are later used. You use the RMM ADDVOLUME or CHANGEVOLUME subcommands with the MEDIATYPE operand to set the media type.

When volumes are entered into a manual tape library, DFSMSrmm ensures that both private and scratch volumes meet these DFSMS requirements: that only supported label types are used and that internal volume serial number matches external volume serial number.

Managing Scratch Pools

The DFSMSrmm scratch pool function in VLPOOL definitions is not honored for scratch mount requests on automated system-managed library resident drives, even if the storage group assigned matches a VLPOOL name. In an automated system-managed library, DFSMS and the library manager control the pooling of scratch volumes with one pool for each media type. There is currently no method to direct mount processing to select a subset of the volumes in a scratch category.

Ejecting Volumes from System-Managed Libraries

You can eject physical volumes from a system-managed library by using various methods. One way is to list a number of volumes and then enter eject commands from the list. You normally use export processing to eject logical volumes. For information about export processing, see “Using DFSMSrmm with the IBM TotalStorage Peer-to-Peer Virtual Tape Server (PtP VTS)” on page 93.

You can also eject volumes from a system-managed library by changing the library name with the RMM CHANGEVOLUME subcommand. When you issue the command, DFSMSrmm requests that the volume is ejected from the original library. Information about the volume is changed in the control data set, and the TCDB record is optionally deleted. If the target library is an automated tape library, the operator must physically transfer the volume. If the source and target libraries are both manual tape libraries, then physical movement is only needed if the volume is to be moved to another shelf location.

You can also mark a volume as ejected from a system-managed library or change the location name to SHELF when the library is not known or offline by using the RMM CHANGEVOLUME subcommand with the FORCE operand. To use the FORCE operand, you must have access to the STGADMIN.EDG.MASTER and STGADMIN.EDG.FORCE RACF profiles. When you use the FORCE operand to update a volume in a system-managed library, DFSMSrmm attempts to update the TCDB. Your request to update DFSMSrmm is completed even when the TCDB is not updated.

DFSMSrmm does not automatically eject volumes from system-managed libraries after completing vital record processing and storage location management processing. See “How Storage Location Management Processing Works” on page 300 for information about adding a job step to eject volumes at a time you determine.

The DFSMSrmm control data set is updated even if ejects are not driven by DFSMSrmm. The eject is detected using the OAM eject installation exit, and DFSMSrmm can control the disposition of the TCDB volume information. During eject processing, DFSMSrmm checks the setting of the DFSMSrmm EDGRMMxx OPTION SMSTAPE(PURGE) operand to determine whether to request deletion of volume information from the TCDB. This optionally overrides the ISMF default cartridge eject option you might have specified. If the volume record is not deleted, DFSMSrmm sets the volume destination information into the TCDB volume shelf location field.

Figure 36 shows how you can eject a volume from an automatic tape library or from a manual tape library to a convenience output station using the RMM CHANGEVOLUME subcommand or the RMM DELETEVOLUME subcommand.

```
RMM CHANGEVOLUME A12345 EJECT(CONVENIENCE)
RMM CHANGEVOLUME A12345 LOCATION(new_lib)
RMM CHANGEVOLUME A12345 LOANLOC(set_loc)
RMM DELETEVOLUME A12345 REMOVE
RMM DELETEVOLUME A12345 FORCE
```

Figure 36. Requesting the Eject of a Volume

You can specify the EJECT operand on the RMM CHANGEVOLUME and DELETEVOLUME subcommands. Specify the EJECT(BULK) operand to direct the ejected volume to the high capacity input/output station. Specify the EJECT(CONVENIENCE) operand to direct the ejected volume to the convenience output station. If you specify EJECT(BULK) and the high capacity output station is not installed, then the eject is routed to a convenience output station. If there is no convenience output station, the eject is then routed to a single cell facility. In a manual tape library, DFSMSrmm ignores the BULK values and the CONVENIENCE values and the cartridge is ejected.

The RMM CHANGEVOLUME subcommand with the HOME operand does not cause an eject or a move as it is only used to update the home location name.

Returning Volumes to the System-Managed Library

Normally system-managed volumes remain resident in the system-managed library unless they are removed for movement to an offsite location, for physical inspection, or for error recovery. When volumes are to be returned to an automated system-managed library, enter the volumes into the appropriate entry station. DFSMSrmm is notified that the move has taken place. When volumes are returned to a manual system-managed library, you must confirm the movement to DFSMSrmm so that DFSMSrmm can initiate entry processing.

There might be times when a volume is not in a system-managed library but is needed for processing to continue. z/OS includes a function called

Volume-Not-In-Library (VNL) processing that allows a volume to be identified for re-entry to a system-managed library at various stages of system processing:
- Job step setup
- Device allocation
- Library mount

The CBRUXVNL OAM installation exit provides support so that a volume that is needed for processing to continue can be entered back into the library. Logical volumes that are exported must be imported to enable them to be used for processing.

For volumes that are defined in the DFSMSrmm control data set, information is provided to determine if the volume can be entered into the library. DFSMSrmm issues messages that provide location, movement, and status information during VNL processing.

If the volume entering the library is marked as "intransit" to other than a system-managed library, ensure that the move for the volume is confirmed as completed. Otherwise the volume entry will be rejected.

Refer to "Managing System-Managed Tape Library Volumes: EDGLCSUX" on page 202 for information on using CBRUXVNL and other OAM installation exits.

Volume-Not-In-Library Processing

When a volume is requested for a job and is not in a system-managed tape library, the DFSMSrmm supplied CBRUXVNL exit is called so the volume can be inserted into a library to prevent job failures from occurring. The CBRUXVNL exit can be modified to suit your requirements.

CBRUXVNL calls DFSMSrmm once to retrieve volume information, and then for a limited set of circumstances, it calls DFSMSrmm a second time to request the operator to enter the volume into a system-managed tape library. The second call is made under the following conditions:
- CBRUXVNL is called from job setup, and the exit determines that the volume needs to be directed to a system-managed tape library.
- CBRUXVNL is called from other than job setup.

CBRUXVNL checks for the requested volume as follows:
- If the volume resides in a loan location and the volume is system-managed, the exit fails the request.
- The exit attempts to get a volume entered into a system-managed library if the volume can be used on the system and when:
  - The volume is in location SHELF and is not moving to a storage location, and its home location is system-managed.
  - DFSMSrmm volume information indicates the volume resides in a system-managed library.
  - The volume destination is a system-managed location.

DFSMSrmm uses the information to build a WTOR message that includes the volume location and that prompts the operator to enter the volume into the system-managed library. For volumes that reside in a VTS, DFSMSrmm returns additional information for a volume. DFSMSrmm indicates if the volume is a physical volume or a logical volume and provides the stacked volume on which the logical volume is exported.

The CBRUXVNL exit does not allow a volume to be used under the following conditions:

- The volume is pending release or is in scratch status.
- The initialize volume action is pending.
- The volume is not to be used on MVS.

In all other cases the CBRUXVNL exit allows the volume to be used, but a non-system managed tape drive is allocated by the system.

Confirming Volume Movement for System Managed Libraries

DFSMSrmm automatically confirms volume moves during cartridge entry processing. When you manually confirm the moves for volumes to an automated tape library, DFSMSrmm checks for the volume and library name in the TCDB and only confirms movements if the volume is library resident. DFSMSrmm If there is a mismatch between the information in the TCDB and the DFSMSrmm control data set, use the RMM CHANGEVOLUME subcommand with the LOCATION operand to update the destination information in the control data set to match the library name in the TCDB.

If a volume is returning to a manual tape library, there is no automatic confirmation. You must explicitly confirm that the move has taken place when the volume is entered into the library. When you confirm that a volume has moved to a manual tape library, DFSMSrmm requests that the volume location in the TCDB is updated with the new location. DFSMSrmm also changes the volume location information in the TCDB when you issue the DFSMSrmm CHANGEVOLUME *volser* LOCATION (*mtl_name*) to change the volume location to a manual tape library.

If you specify the RMM CHANGEVOLUME subcommand with the EJECT operand for a volume for which no move has been identified, DFSMSrmm does not set a destination for the volume but the volume is set to be ’intransit’. Once the volume is ejected and cannot be entered again, you should specify the location where the volume is placed and then confirm the move.

When you specify the RMM CHANGEVOLUME subcommand with the LOCATION operand, DFSMSrmm sets a destination for the volume. You only need to confirm the move after the eject.

Defining System-Managed Volume Information

System-managed tape volumes are defined in the tape configuration database (TCDB). The TCDB is a catalog that is marked as a volume catalog (VOLCAT) containing tape volume and tape library records. For more information about the TCDB, see *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries*.

System-managed tapes should be defined to the DFSMSrmm control data set for the volumes to be managed by DFSMSrmm. To define volumes to both DFSMSrmm and the TCDB, use the RMM ADDVOLUME subcommand for volumes not yet known to DFSMSrmm, and the RMM CHANGEVOLUME subcommand for volumes already defined to DFSMSrmm but not the TCDB. For information on using DFSMSrmm commands, see *z/OS DFSMSrmm Guide and Reference*.

When you use DFSMSrmm to request that a volume should move to a system-managed tape library, DFSMSrmm ensures volume information is recorded in the TCDB.

Keeping System-Managed Volume Information Consistent

DFSMSrmm uses information in the TCDB to verify requests or to obtain information about volumes that are being added to DFSMSrmm. When you define volumes to DFSMSrmm by using the RMM ADDVOLUME subcommand with the STATUS(VOLCAT) operand, DFSMSrmm uses TCDB information to update the DFSMSrmm control data set. When there is a conflict between information in the TCDB and information in the control data set, DFSMSrmm uses the value in the TCDB.

To control the way DFSMSrmm updates the information in the TCDB, specify the DFSMSrmm EDGRMM*xx* parmlib OPTION command SMSTAPE operand and the OPTION command OPMODE operand. See “Defining System Options: OPTION” on page 134 for information about the DFSMSrmm EDGRMM*xx* parmlib OPTION command. When you run DFSMSrmm in PROTECT mode, DFSMSrmm updates the TCDB. If you are running DFSMSrmm in other modes, you can decide if and when you want DFSMSrmm to update the TCDB. DFSMSrmm updates the following TCDB information:

Volume owner information
DFSMSrmm uses the first 8 characters of the 64 bytes of owner information in the TCDB to store the DFSMSrmm volume owner information. DFSMSrmm does not overlay any information you might have entered into the first 8 bytes. Keep the first bytes of the field blank and DFSMSrmm maintains the volume information for you.

Expiration date
The expiration date is updated in the tape volume record whenever the DFSMSrmm control data set is updated.

Volume use attribute (status)
DFSMSrmm updates this information when an RMM TSO subcommand is issued and a volume’s status is changed. For example, if you issue the RMM CHANGEVOLUME subcommand with the STATUS(USER) operand or the RMM GETVOLUME subcommand, the TCDB volume use attribute is changed to PRIVATE. Volume status is also changed when volumes are returned to scratch status during inventory management.

Storage group name
DFSMSrmm updates this information when an RMM TSO subcommand is issued to change TCDB information.

You can use the DFSMSrmm EDGUTIL utility to perform the following tasks:

- Check the consistency of the control data set with the TCDB and the library manager database and use EDGUTIL MEND(SMSTAPE) to synchronize the TCDB and library manager database from the DFSMSrmm control data set.
- Mend information in the control data set based on information in the TCDB and the library manager database.

For more information about EDGUTIL, see “Using EDGUTIL for Tasks Such as Creating and Verifying the Control Data Set” on page 336.

Use the RMM CHANGEVOLUME subcommand with the LOCATION(*mtl_name*) operand to rebuild manual tape library information in the TCDB.

If you use DFSMSrmm, ISMF, or operator commands to eject a volume, DFSMSrmm notes that the volume is ’intransit’. During eject processing, DFSMSrmm checks the setting of the DFSMSrmm EDGRMMxx OPTION SMSTAPE(PURGE) operand to determine whether to request deletion of volume information from the TCDB. This optionally overrides the ISMF default cartridge eject option you might have specified. If the volume record is not deleted, DFSMSrmm sets the volume destination information into the TCDB volume shelf location field.

Initializing Scratch Volumes in System-Managed Libraries

You can use these methods to label volumes in an automated tape library:

- Use EDGINERS or IEHINITT to label them before they are entered.
- For scratch volumes, let the volume be labeled the first time that the volume is used for output after it enters the automated tape library. DFSMSdfp performs labeling during OPEN processing.
- Use EDGINERS or IEHINITT for volumes in all system-managed tape libraries.

To label volumes by using the EDGINERS initialize function, set the DFSMSrmm initialize action by using the DFSMSrmm ISPF dialog or the RMM TSO subcommand.

If you set the initialize action for a scratch volume before it enters an automated tape library and then enter the volume into the automated tape library, DFSMSrmm defers initialization to DFSMSdfp labeling support. If you later eject the scratch volume before it is used DFSMSrmm reinstates the initialize action.

If you set the initialize action for a scratch volume while it is resident in an automated tape library, the initialize action is set outstanding and you must use the DFSMSrmm EDGINERS utility to create the labels. See “Using EDGINERS with System-Managed Tape Libraries” on page 362 for additional information.

When you use the DFSMSdfp labeling support in an automated tape library, DFSMSdfp obtains the volume serial number from the library vision system, but still issues the WTOR IEC704A to prompt the operator to provide optional owner information. The IEC704A message text includes an indicator that the volume is system managed. Normally, DFSMSrmm provides information to bypass the issuing of the message. If DFSMSrmm is running in record mode or a volume is rejected in warning mode, the WTOR is issued and must be replied to by the operator.

The DFSMSdfp labeling support for manual tape libraries cannot obtain the volume serial number from the vision system. DFSMSdfp still uses the WTOR IEC704A to prompt the operator for it.

You can decide to exploit DFSMSdfp automatic labeling to avoid unnecessary mounting of volumes or use EDGINERS before volumes enter the tape library or after volumes are resident in the tape library.

Using Storage Group Names

You can specify a storage group name for each volume defined in the control data set. Setting the name is optional.

DFSMSrmm validates the storage group name by using services that are provided by the Storage Management Subsystem. For system-managed volumes, DFSMSrmm passes the storage group name to OAM unless the volume is in scratch status. For non-system managed volumes, you can use the assigned storage group names for scratch pooling. For more information about using SG for scratch pooling, see “Using SMS Tape Storage Groups for DFSMSrmm Scratch Pooling” on page 71.

When a volume is moved from one system-managed tape library to another, DFSMSrmm does not change the storage group name. OAM validates the storage group name during entry to the new library and fails the entry if the storage group is not defined for use in that new library.

DFSMSrmm uses the OAM installation exits to record any changes that ISMF and Automatic Class Selection Routine processing make to the storage group name. This function enables DFSMSrmm to provide the storage group name at cartridge re-entry time, if the volume information had been removed from the TCDB.

For information on how DFSMSrmm records data class, management class, storage class, and storage group information, see *z/OS DFSMSrmm Guide and Reference*.

Using DFSMSrmm with the IBM TotalStorage Peer-to-Peer Virtual Tape Server (PtP VTS)

DFSMSrmm supports Virtual Tape Server (VTS) libraries such as automated system-managed libraries with extensions to that support to handle the special requirements for different volume types and functional capability. DFSMSrmm supports virtual tape server import/export in different ways depending on whether or not DFSMSrmm stacked volume support is enabled. When you enable stacked volume support, DFSMSrmm tracks logical volumes and stacked volumes.

When you do not enable stacked volume support, DFSMSrmm does not track the stacked volumes. DFSMSrmm supports the IBM Peer-to-Peer VTS by ensuring that you cannot use the names of distributed VTS libraries with DFSMSrmm. You must only use the names of consolidated libraries with DFSMSrmm.

Related Reading:

- See *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries* for information about using OAM for virtual tape server support.
- See *z/OS DFSMSrmm Reporting* for information about the DFSMSrmm EDGRPTD utility that you can use to obtain information about volumes that reside in virtual tape server libraries.
- See *IBM TotalStorage Enterprise Automated Tape Library (3494) Introduction and Planning Guide*, *IBM TotalStorage Enterprise Automated (3494) Tape Library Operator Guide*, *TotalStorage Automated Tape Library (3495) Introduction*, and *IBM TotalStorage Enterprise Automated Tape Library (3495) Operator's Guide* for information about the format for the import and export lists.

Defining Logical Volumes in a Virtual Tape Server Library

A logical volume is a volume that resides in a virtual tape server library either on DASD (in the tape volume cache as a virtual volume) or on a stacked volume (as a logical volume) and is referenced from the host as if it were a physical volume.

The way you define and enter logical volumes is common regardless of how you set up DFSMSrmm. Other functions are dependent on how you request stacked volumes to be managed. For details of how DFSMSrmm supports export and import processing for logical volumes refer to "DFSMSrmm Support for Stacked Volumes When Stacked Volume Support Is Enabled" on page 97 and "DFSMSrmm Support for Stacked Volumes When Stacked Volume Support Is Not Enabled" on page 101.

When volumes are automatically added to DFSMSrmm, DFSMSrmm sets the volume type based on where the volume resides. If the volume resides in a VTS, DFSMSrmm sets the volume type to logical. If the volume does not reside in a VTS, DFSMSrmm sets the volume type to physical. You can chose to define the volumes to DFSMSrmm rather than having the volumes automatically defined when they are entered into the library. Specify the correct volume type for each volume that you define. If you do not specify the correct volume type for volumes residing in system-managed tape libraries, these volumes will not be processed successfully.

To ensure that all logical volumes are correctly identified to DFSMSrmm as logical volumes, use the DFSMSrmm TSO subcommand example shown in Figure 37 to change the volume type of previously defined volumes. DFSMSrmm allows you to identify any volume as a logical volume as long as it is not resident in a system-managed automated tape library.

```
RMM SEARCHVOLUME VOLUME(*) LIMIT(*) OWNER(*) LOCATION(vts) -
 CLIST('RMM CHANGEVOLUME ',' TYPE(LOGICAL)')
 EXEC EXEC.RMM.CLIST
```

Figure 37. Changing Volume Type

Logical Volume Cartridge Entry Processing

DFSMSrmm processes new logical volumes and imported volumes differently at entry time.

- DFSMSrmm automatically defines both new logical volumes and imported logical volumes if they are not defined to DFSMSrmm already. DFSMSrmm does not define rack numbers for these volumes.
- DFSMSrmm checks the rack number for a logical volume that is already defined to DFSMSrmm. If the rack number does not match the logical volume's volume serial number, DFSMSrmm fails the entry request and issues message EDG8189I. If the rack number matches the volume serial number, DFSMSrmm continues processing the request. DFSMSrmm frees the rack number for the logical volume so the logical volume is no longer associated with the rack number.
- DFSMSrmm checks that an imported logical volume that is already defined to DFSMSrmm, is also a logical volume and does not reside in a virtual tape server library. When a logical volume is being imported, DFSMSrmm checks that the volume, if it is already defined to DFSMSrmm, is correctly defined as an exported logical volume. DFSMSrmm issues message EDG8183I when the entry request fails.
- DFSMSrmm accepts new scratch logical volumes that match existing scratch logical volumes. DFSMSrmm updates the volume information with information for the new volume. If the volume serial numbers match but the volume is not defined as a scratch volume, DFSMSrmm fails the entry request and issues message EDG8182I.

Table 14 describes the entry processing decisions DFSMSrmm makes.

Table 14. DFSMSrmm Entry Processing Decisions

Case	Physical Volume	Stacked Volume	Logical Volume	Imported Volume
Volume is not defined; no REJECT.	Added	Not applicable	Added scratch or private	Added private
Volume is not defined; REJECT.	Ignored	Not applicable	Ignored	Ignored
Volume is defined for use with MVS.	Updated	Not applicable	Updated	Updated
Volume is defined for use with VM.	Ignored	Not applicable	Ignored	Ignored

Table 14. DFSMSrmm Entry Processing Decisions (continued)

Case	Physical Volume	Stacked Volume	Logical Volume	Imported Volume
Rack number associated with the volume is the same as the volume serial number.	Updated	Not applicable	Updated; DFSMSrmm clears the rack number.	Updated; DFSMSrmm clears the rack number.
Rack number associated with the volume is not the same as the volume serial number.	Entry fails; DFSMSrmm issues EDG8189I.	Not applicable	Entry fails; DFSMSrmm issues EDG8189I.	Import fails; DFSMSrmm issues EDG8189I.
No rack number is present.	Updated	Not applicable	Updated	Updated
Volume is a scratch and not logical volume.	Updated	Not applicable	Entry fails; DFSMSrmm issues EDG8182I.	Import fails; DFSMSrmm issues EDG8182I.
Volume is private and not a logical volume.	Updated	Not applicable	Entry fails; DFSMSrmm issues EDG8182I.	Import fails; DFSMSrmm issues EDG8182I.
Volume is a logical volume and not an exported volume.	Entry fails; DFSMSrmm issues EDG8184I.	Not applicable	Update	Import fails; DFSMSrmm issues EDG8183I.
Volume is a logical exported volume.	Entry fails; DFSMSrmm issues EDG8184I.	Not applicable	Entry fails; DFSMSrmm issues EDG8184I.	Updated
Volume destination is not the library.	Entry fails; DFSMSrmm issues EDG8192I.	Not applicable	Entry fails; DFSMSrmm issues EDG8192I.	Import fails; DFSMSrmm issues EDG8192I.
Volume owner is not valid.	Entry fails; DFSMSrmm issues EDG8195I.	Not applicable	Entry fails; DFSMSrmm issues EDG8195I.	Import fails; DFSMSrmm issues EDG8195I.
Defined rack number=vol is not available.	Entry fails; DFSMSrmm issues EDG8198I.	Not applicable	Entry fails; DFSMSrmm issues EDG8198I.	Import fails; DFSMSrmm issues EDG8198I.
Maximum retention period is exceeded.	Entry fails; DFSMSrmm issues EDG8196I.	Not applicable	Not applicable	Import fails; DFSMSrmm issues EDG8196I.

Managing Stacked Volumes

A *stacked volume* is a volume in a virtual tape server library that is used to store one or more logical volumes.

DFSMSrmm can manage volume movement based on the logical volume or the stacked volume. You control the way DFSMSrmm manages stacked volumes by enabling stacked volume support as described in “Enabling Stacked Volume Support” on page 104. If you do not enable stacked volume support, you can always use the DFSMSrmm TSO subcommands to add, change, or delete information about stacked volumes. You can also perform export and import processing of logical volumes.

If you do not enable stacked volume support, DFSMSrmm bases volume movement on the logical volume. DFSMSrmm records the stacked volume as the in container value but does not use the stacked volumes to determine volume movement. See “DFSMSrmm Support for Stacked Volumes When Stacked Volume Support Is Not

Enabled" on page 101 for details about how DFSMSrmm supports stacked volumes when stacked volume support is not enabled.

If you enable stacked volume support, DFSMSrmm tracks stacked volumes and bases volume movement on the stacked volume. See "DFSMSrmm Support for Stacked Volumes When Stacked Volume Support Is Enabled" on page 97 for details about how DFSMSrmm supports stacked volumes when stacked volume support is enabled.

Stacked volumes normally start and end their movement in a VTS automated tape library. The host system is not notified when stacked volumes enter or leave an automated tape library so stacked volumes cannot be managed the same way that physical volumes in an automated tape library are managed.

When you use a DFSMSrmm command for a stacked volume, DFSMSrmm checks both the TCDB and the library manager database to see if the volume is known. If the volume is not known to the library manager, you cannot define the volume as being in the VTS library, but the destination is set to the VTS library name. Once the volume is entered into the VTS, you can use the CHANGEVOLUME subcommand with CONFIRMMOVE to inform DFSMSrmm that the volume is now in the VTS.

Defining Stacked Volumes to DFSMSrmm

Before you begin: Use the library manager console to ensure that the ranges of stacked volumes for containers are defined correctly. By doing so, you can avoid accidentally entering stacked volumes into the library as physical volumes.

Recommendation: Always define stacked volumes to DFSMSrmm. This ensures that the volumes cannot be used outside of the VTS. It also ensures that DFSMSrmm checks at cartridge entry processing time that a physical volume or logical volume does not duplicate a stacked volume.

You can define stacked volumes whether stacked volume support is enabled or not. DFSMSrmm defines the stacked volumes automatically at export time when stacked volume support is enabled. There is no automatic host notification when stacked volumes enter the VTS, leave the VTS, or change category, so you must use DFSMSrmm commands if you want to have all stacked volumes defined to DFSMSrmm.

To define stacked volumes to DFSMSrmm and ensure that they can only be used as stacked volumes inside a VTS, perform the following steps:

1. Issue the RMM ADDVOLUME subcommand with the TYPE(STACKED) operand to manually define stacked volumes to DFSMSrmm.
2. Take one of the following steps to prevent the use of stacked volumes outside the VTS:
   - Specify the DFSMSrmm EDGRMM*xx* parmlib command REJECT ANYUSE(*) to prevent the use of any volumes not defined to DFSMSrmm. See "Defining Tapes Not Available on Systems: REJECT" on page 157.
   - Specify the DFSMSrmm EDGRMM*xx* parmlib command REJECT ANYUSE(*prefix**) to prevent the use of ranges of stacked volumes.
   - Specify the RMM ADDVOLUME subcommand with the USE(VM) operand and the TYPE(STACKED) operand to manually define stacked volumes which cannot be used on MVS to DFSMSrmm.

Changing Stacked Volume Information

Once you have defined a stacked volume to DFSMSrmm you do not normally need to change information about that volume. Once stacked volume support is enabled, DFSMSrmm manages the information about stacked volumes and logical volumes and the movement of stacked volumes as they are exported from the VTS library.

You do need to use the CHANGEVOLUME command to confirm the movement of the stacked volume as you would for a physical volume that was moving.

Assigning a Shelf Location for a Stacked Volume

You can optionally assign a shelf location to a stacked volume. When a stacked volume is resident in a VTS library, no shelf location is needed; however, if it is stored outside the VTS assigned to location SHELF you can assign a rack number using either the RACK or POOL operands. Specify RACK with a specific rack number or use POOL to allow DFSMSrmm to select the first empty rack number in that pool of shelf locations. During normal processing, DFSMSrmm storage location management processing assigns shelf locations to a stacked volume that must move to a shelf-managed storage location.

Deleting Stacked Volume Information

You can only delete an empty stacked volume. Ensure that all logical volumes have been imported or removed from the stacked volume. To remove a volume from a stacked volume, use the RMM CHANGEVOLUME subcommand with the CONTAINER(' ') operand to clear the container name.

DFSMSrmm Support for Stacked Volumes When Stacked Volume Support Is Enabled

With DFSMSrmm stacked volume support enabled, you can manage the movement of logical volumes by using stacked volumes. To enable stacked volume support, see “Enabling Stacked Volume Support” on page 104.

To ensure that stacked volumes are managed correctly, DFSMSrmm inventory management processing checks that stacked volumes reside in a library. At the completion of export processing for a single stacked volume, DFSMSrmm uses the required location of the stacked volume to attempt to start its movement. If the required location is not shelf-managed, DFSMSrmm sets the destination, but does not mark the stacked volume as being ’intransit’. If the required location is shelf-managed, DFSMSrmm storage location management processing starts the volume move. Inventory management checks to see if closed, not-empty stacked volumes still reside in their location, and marks those no longer resident as ’intransit’. Storage location management processing starts movement of stacked volumes by assigning bin numbers for storage locations and setting the destination for the volume.

The next time inventory management is run, DFSMSrmm checks the library to see if the stacked volume has been ejected, and if so, the stacked volume is marked as being ’intransit’. Those stacked volumes which are ’intransit’ to a destination VTS are checked for library residence at their destination and the move confirmed if found resident. Stacked volumes that are ’intransit’ to storage locations are confirmed only if a global confirm move has been requested for the location and destination pair. When stacked volumes move from storage locations they are set ’intransit’ by storage location management processing. The result is that DFSMSrmm can automatically determine some of the moves for stacked volumes and needs you to confirm those to storage locations.

Resolving Movement Conflicts

If there is a movement conflict for multiple logical volumes in a stacked volume, DFSMSrmm resolves the conflict by using location priority to determine where the stacked volume should be moved. See “Moving Volumes” on page 298 for information about how DFSMSrmm prioritizes volume movement.

Confirming Stacked Volume Movement

You confirm the movement of stacked volume containers to storage locations when their movement is completed. Use the RMM CHANGEVOLUME * CMOVE subcommand. The READYTOSCRATCH option is not applied to stacked volume moves and container moves are confirmed either with the ALL option or the NOTREADYTOSCRATCH option.

The stacked volume is used to identify the media that is used for the export container volume, to track its location, and to aggregate the logical volumes contained in the volume. All movement for exported logical volumes is managed and tracked using the stacked volume. You can move a stacked volume manually, but normally you control the movement automatically using vital record specifications. The vital record specifications identify movement for data sets on logical volumes. The movement specified for the logical volumes in the stacked volume container direct the movement of the stacked volume. DFSMSrmm automatically defines stacked volumes at EXPORT time. DFSMSrmm keeps track of the ″in container″ information only while the logical volume is exported.

DFSMSrmm Support for Export Processing When Stacked Volume Support Is Enabled

To use DFSMSrmm for export processing, you must first make sure that the volume information reflects the most current required location information. Perform DFSMSrmm inventory management vital record processing to set the required location for any volume moves that are required.

As logical volumes are exported, DFSMSrmm tracks the ″in container″ stacked volume and drives the stacked volume movement based on the required location of the contained volumes.

Use DFSMSrmm export processing to remove logical volumes from a VTS.

1. Define vital record specifications that specify locations to which data sets in the VTS should be moved. Use the vital record specification location name as the key for managing export. You could associate location names with different types of retention to have more control over the volumes that you want to export. All the volumes with the same required location name, in the same export request, get exported together.
2. Run DFSMSrmm inventory management vital record processing to determine which logical volumes should be moved and to set the required location for each volume to be moved.
3. Create a list of volumes to export after you run vital record processing. The required moves are volume moves that are identified during vital record processing or by changes made when you issue the RMM CHANGEVOLUME subcommand. Use the RMM SEARCHVOLUME subcommand with the REQUIRED and CLIST operands for each storage location to obtain volume information for the list as shown in Figure 38 on page 99. DFSMSrmm returns a list of volumes to export that you can use as input for creating the export list logical volume file 1. Consider using the RETDATE operand to group the

volumes that are expiring in the same time period into the same export request. You can use the EDGJIMPC sample to reformat the CLIST file for use in the export list.

```
RMM SEARCHVOLUME VOLUME(*) OWNER(*) LIMIT(*) REQUIRED(STORE1) LOCATION(vts) -
   CLIST(,' STORE1') RETDATE(2001/020)
```

Figure 38. Searching the Required Location for Logical Volumes

See the *z/OS DFSMSrmm Guide and Reference* for information about the RMM SEARCHVOLUME subcommand. See these examples in SYS1.SAMPLIB for the format and required files for the export list volume:

- CBRSPSXP is the sample JCL for export list volume scratch request.
- CBRSPPXP is the sample JCL for export list volume private request.

When stacked volume support is enabled, you can no longer track the movement of logical volumes using the LOCATION and DESTINATION fields of the volume information. Any volume exported on a stacked volume resides in a container and has no assigned location name. DFSMSrmm continues to manage the volume movement using vital record specifications for data sets and volumes but uses this information only to set the REQUIRED location. DFSMSrmm never assigns a destination to a logical volume.

4. Start the export process as described in the *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries* using the LIBRARY EXPORT command or use the CBRSPLCS sample program.

   The VTS export process runs asynchronously to DFSMSrmm processing. Once the export request has been initiated, VTS library signals trigger DFSMSrmm actions related to export processing.

   During the VTS export process, logical volumes are copied to a stacked volume and the stacked volume is completed. For each logical volume copied to the stacked volume, DFSMSrmm is notified of both the logical volume and stacked volume. DFSMSrmm updates the DFSMSrmm control data set with the volume serial number of the stacked volume as the ’in container’.

   During export processing, DFSMSrmm updates the ’In container’ value for each logical volume exported. DFSMSrmm updates the stacked volume and tracks export completion, marking the stacked volume closed.
5. In a single run of DFSMSrmm inventory management, perform the following tasks:
   - Set the required location for the volumes by running vital record processing.
   - Set the destinations for the volumes by running storage location management processing.
   - Create an extract data set after VTS export processing is completed.
6. Create movement report and pick lists by using the DFSMSrmm EDGRPTD report utility.
7. Use the movement reports to eject the stacked volumes from the VTS using the library manager and move them to the destination storage location. Transfer the stacked volumes in the export hold category to the exit station using the library manager console. Physically move the stacked volumes listed in your movement report from the VTS exit station to the destination storage location.
8. Confirm that volumes have been moved. When the volumes have been moved, use the RMM CHANGEVOLUME subcommand, as shown in Figure 39 on page 100 to confirm the completion of the movement of volumes.

```
RMM CHANGEVOLUME * CONFIRMMOVE(vts,ALL)
```

Figure 39. Confirming Volume Moves for Exported Volumes

When the stacked volume is moved, you must confirm the move for all the logical volumes that reside on the volume. You can use the RMM CHANGEVOLUME subcommand with the CONFIRMMOVE operand to confirm the move. You must confirm the move of stacked volumes from the VTS. When DFSMSrmm stacked volume support is enabled, confirm that the stacked volume has moved.

DFSMSrmm Support for Import Processing When Stacked Volume Support Is Enabled

DFSMSrmm supports the importing of logical volumes that are defined to DFSMSrmm or not defined to DFSMSrmm. During import processing for a logical volume, DFSMSrmm automatically adds the volume information to the DFSMSrmm control data set or leaves the volume to be processed on another system. DFSMSrmm does not automatically add rack numbers for logical volumes because rack numbers are not supported for logical volumes.

You can initiate the import of logical volumes as follows.

1. Use the library manager console to transfer stacked volumes to be imported from the Unassign category to the Import category. See *TotalStorage Automated Tape Library (3495) Introduction* and *TotalStorage Automated Tape Library (3495) Operator’s Guide* for more information about the library manager.
2. Create an import list. See these examples in SYS1.SAMPLIB for the format and required files for the import list volume.
   - CBRSPSIM is the sample JCL for import list volume scratch request.
   - CBRSPPIM is the sample JCL for import list volume private request.

   To create the volume list for file 1 of the import list logical volume, you can search in the DFSMSrmm control data set, tailor the DFSMSrmm report extract data set, or use any other method to identify the volumes to be imported.

   You can use the RMM SEARCHVOLUME subcommand to build the list of logical volumes with their containing stacked volume and status as shown in the following examples. When you specify the TYPE(LOGICAL) operand, DFSMSrmm returns the container volume serial number, logical volume serial number, and volume status between your specified CLIST prefix and suffix strings. The SEARCHVOLUME output file can be used as input for creating the import list logical volume file 1 after it is reformatted using the EDGJIMPC sample.

   When stacked volumes are returned to the library, you can build import lists by either using the logical volumes or using the stacked volumes. To build a list of stacked volumes to be imported, you can issue the DFSMSrmm TSO subcommand shown in Figure 40.

```
RMM SEARCHVOLUME VOLUME(*) OWNER(*) LIMIT(*) TYPE(STACKED) DESTINATION(vts) -
   CLIST
```

Figure 40. Building a List of Stacked Volumes to be Imported from a Single Stacked Volume

To build a list of logical volumes to import from a single stacked volume, you can issue the DFSMSrmm TSO subcommand shown in Figure 41 on page 101.

```
RMM SEARCHVOLUME CONTAINER(S12345) VOL(*) OWN(*) LIMIT(*) TYPE(LOGICAL) CLIST
```

Figure 41. Building a List of Logical Volumes to be Imported from a Single Stacked Volume

To build a list of logical volumes to import from multiple stacked volumes, you can issue the DFSMSrmm TSO subcommand shown in Figure 42.

```
RMM SEARCHVOLUME VOLUME(*) LIMIT(*) OWNER(*) CONTAINER(*) -
   REQUIRED(vts) TYPE(LOGICAL) CLIST
```

Figure 42. Building a List of Logical Volumes to be Imported from Multiple Stacked Volumes

3. Request import processing by using the CBRXLCS macro, the OAM CBRSPLCS sample programs, or the LIBRARY command.

   During import processing, the imported volumes can be automatically defined to DFSMSrmm or the volumes can be left for processing by another system if the library is partitioned. If the volume is already known to DFSMSrmm, the volume is accepted for processing if it is defined as an exported logical volume, otherwise the volume is rejected.
4. To complete the import process, DFSMSrmm uses the cartridge entry processing installation exit to track logical volumes that are imported by the VTS. OAM calls the exit once for each logical volume imported. For logical volumes, DFSMSrmm removes the stacked volume association. The stacked volume is updated to remove information for each imported volume.
5. When all logical volumes on a stacked volume are imported, the count of contained volumes is set to 0. You now must decide what to do with the stacked volume. For example, you can return the scratch volume to the pool of empty stacked volumes for use by the VTS; use the library manager console to transfer empty stacked volumes to other categories within the VTS so they ready for reuse.

DFSMSrmm Support for Stacked Volumes When Stacked Volume Support Is Not Enabled

A stacked volume is a volume in a virtual tape server library that is used to store one or more logical volumes. When stacked volume support is not enabled, DFSMSrmm does not track stacked volumes, but allows you to define them to DFSMSrmm. DFSMSrmm records the name of the exported stacked volume as the ’In container’ information for each logical volume. DFSMSrmm keeps track of the ’In container’ information only while the logical volume is exported.

DFSMSrmm Support for Export Processing When Stacked Volume Support Is Not Enabled

To use DFSMSrmm for export processing, you must first make sure that volume information reflects the most current location information. Perform DFSMSrmm inventory management storage location processing to set the destinations for any volume moves that are required.

As logical volumes are exported, DFSMSrmm tracks the ’in container’ stacked volume and drives movement based on the destination of logical volumes. Use DFSMSrmm export processing to remove logical volumes from a VTS.

1. Define vital record specifications that specify locations to which data sets in the VTS should be moved.
2. Run DFSMSrmm inventory management vital records and storage location management processing to determine which logical volumes should be moved.

DFSMSrmm allocates a bin number to the volume if the volume's destination is shelf-managed. There will be unused bin numbers for logical volumes that are moving to shelf-managed storage locations. You will waste bin numbers because EDGRPTD ignores the assigned bin numbers for logical volumes.

3. Create a list of volumes to export. Create the list after you run storage location management. The moves are those that are identified during vital record processing or those changes that are made when you issue the RMM CHANGEVOLUME subcommand. Use the RMM SEARCHVOLUME subcommand to obtain volume information for the list as shown in Figure 43.

```
RMM SEARCHVOLUME VOLUME(*) LIMIT(*) LOCATION(vts)-
   DESTINATION(dest) CLIST(,',dest') -
   INTRANSIT(N)
```

Figure 43. Creating a Volume Export List

   DFSMSrmm returns a list of volumes to export that you can use as input for creating the export list logical volume file 1. See these examples in SYS1.SAMPLIB for the format and required files for the export list volume.
   - CBRSPSXP — sample JCL for export list volume scratch request.
   - CBRSPPXP — sample JCL for export list volume private request.

   You can track the movement of logical volumes by using the LOCATION field and DESTINATION field of the volume information.

4. Start the export process as described in the *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries* using the LIBRARY EXPORT command or use the CBRSPLCS sample program.

   The VTS export process runs asynchronously to DFSMSrmm processing. Once the export request has been initiated, VTS library signals trigger DFSMSrmm actions related to export processing.

   During the VTS export process, logical volumes are copied to a stacked volume, and the stacked volume is completed. For each logical volume copied to the stacked volume, DFSMSrmm is notified of both the logical volume and stacked volume. DFSMSrmm updates the DFSMSrmm control data set with the volume serial number of the stacked volume as the 'in container'.

5. After export VTS processing is completed, you can run DFSMSrmm inventory management report extract processing to obtain information about the volumes that were processed.
6. Create movement report and pick lists by using the DFSMSrmm EDGRPTD report utility.
7. Use the movement reports to eject the stacked volumes from the VTS using the library manager and move them to the destination storage location. Transfer the stacked volumes in the export hold category to the exit station using the library manager console.
8. Physically move the stacked volumes listed in your movement report from the VTS exit station to the destination storage location.
9. Confirm that volumes have been moved. When the stacked volume is moved, you must confirm the move for all the logical volumes that reside on the volume. Use the RMM CHANGEVOLUME subcommand as shown in Figure 44 on page 103 to confirm the completion of the movement of volumes.

```
RMM CHANGEVOLUME * CONFIRMMOVE(vts,ALL)
```

Figure 44. Confirming Volume Moves for Exported Volumes

DFSMSrmm Support for Import Processing When Stacked Volume Support Is Not Enabled

DFSMSrmm supports the importing of logical volumes that are defined to DFSMSrmm or not defined to DFSMSrmm. During entry processing for a logical volume, DFSMSrmm automatically adds the volume information to the DFSMSrmm control data set or leaves the volume to be processed on another system. DFSMSrmm does not automatically add rack numbers for logical volumes because rack numbers are not supported for logical volumes.

You can initiate the import of logical volumes independently of DFSMSrmm by using the library manager console and creating an import list volume.

1. Use the library manager console to transfer stacked volumes to be imported from the Unassign category to the Import category. See *TotalStorage Automated Tape Library (3495) Introduction* and *TotalStorage Automated Tape Library (3495) Operator's Guide* for more information about the library manager.
2. Create an import list volume. See these examples in SYS1.SAMPLIB for the format and required files for the import list volume.
   - CBRSPSIM -- sample JCL for import list volume scratch request.
   - CBRSPPIM -- sample JCL for import list volume private request.

   To create the volume list for file 1 of the import list logical volume, you can search in the DFSMSrmm control data set, tailor the DFSMSrmm report extract file tailoring, or use any other method to identify the volumes to be imported.

   You can use the RMM SEARCHVOLUME subcommand to build the list of logical volumes with their containing stacked volume and status as shown in Figure 45. Specify the TYPE(LOGICAL) operand and DFSMSrmm returns the container volume serial number, logical volume serial number, and volume status between your specified CLIST prefix and suffix strings. The resultant output file can be used as input for creating the import list logical volume file 1 after it is reformatted using the EDGJIMPC sample.

```
RMM SEARCHVOLUME VOLUME(*) LIMIT(*) OWNER(*) DESTINATION(vts) -
   TYPE(LOGICAL) CLIST
```

Figure 45. Creating a Volume Import List

3. Request import processing by using the CBRXLCS macro, the OAM CBRSPLCS sample programs, or the LIBRARY command.

   During import processing, the imported volumes can be automatically defined to DFSMSrmm or the volumes can be left for processing by another system if the library is partitioned. If the volume is already known to DFSMSrmm, the volume is accepted for processing if it is defined as an exported logical volume, otherwise the volume is rejected.
4. To complete the import, DFSMSrmm uses the cartridge entry processing installation exit to track logical volumes that are imported by the VTS. OAM calls the exit once for each logical volume imported. DFSMSrmm removes the stacked volume association for logical volumes. DFSMSrmm removes the 'in container' value for the logical volumes.

5. When all logical volumes on a stacked volume are imported, you must decide what to do with the stacked volume. For example, you can return the scratch volume to the pool of empty stacked volumes for use by the VTS; use the Library Manager console to transfer empty stacked volumes to other categories within the VTS so they are ready for reuse.

Enabling Stacked Volume Support

With DFSMSrmm stacked volume support enabled, you can manage the movement of logical volumes using stacked volumes. Without stacked volume support enabled, you can define and list stacked volumes using the DFSMSrmm ADDVOLUME TSO subcommand with the TYPE(STACKED) operand. DFSMSrmm inventory management ignores any stacked volumes that you have defined and uses just the container name in the volume records.

Prior to enabling stacked volume support, consider these conditions:

- You cannot disable stacked volume support once it is enabled.
- Do not enable support unless all systems using a control data set support stacked volumes. If the control data set is shared with a lower level and support is enabled you can create inconsistent information in the control data set. Correct inconsistencies by using the DFSMSrmm EDGUTIL MEND utility before you can run inventory management.

To enable stacked volume support, perform the following tasks:

1. Update all systems sharing a control data set to the DFSMSrmm level of code containing stacked volume support.
2. If you have stacked volumes defined to DFSMSrmm, but not as stacked volumes, you need to change volume information. Use the RMM SEARCHVOLUME subcommand to set the correct location information and volume type for non-exported stacked volumes. You need to do this before running EDGUTIL MEND. Although the EDGUTIL processing changes the volume type to stacked, it does not set the correct location information unless the stacked volume is exported and contains volumes.

   You can use the RMM SEARCHVOLUME subcommand to build the command to change the volumes:

   ```
   RMM SEARCHVOLUME VOLUME(ST*) OWNER(*) LIMIT(*) -
    CLIST('RMM CHANGEVOLUME ',' TYPE(STACKED) LOCATION(vts_name) NORACK')
   ```
3. Use EDGUTIL UPDATE with the STACKEDVOLUME(YES) operand on the CONTROL statement of the SYSIN file.

   Once you have enabled support, you can use EDGUTIL with VERIFY(VOLUME) to check if the container information is consistent.
4. Use the RMM LISTCONTROL CNTL subcommand to display the status of support.

   If support shows MIXED, run the EDGUTIL MEND utility to make the container information consistent. The inconsistency is the result of having container information in volume records in the control data set.

   During EDGUTIL MEND processing, DFSMSrmm creates the necessary stacked volumes if you have not previously defined them to DFSMSrmm using the DFSMSrmm TSO subcommands.

   During EDGUTIL MEND processing, DFSMSrmm converts each volume in a container as being in the container instead of a DFSMSrmm location. The location and bin number information in volumes which are in a container is removed, and bins returned to empty status. A bin number is assigned to the stacked volume if the location it is in is bin managed. The bin number is selected from one of the contained volumes by using location priority. The bin number choice should reflect the processing used by EDGRPTD in producing movement reports before stacked volume support is enabled.

From now on, when you run storage location processing, DFSMSrmm moves the stacked volumes based on the required location and priority of the contained logical volumes.

5. Update your procedures used to export or import logical volumes to use the required location of the logical volume instead of using the destination. See “DFSMSrmm Support for Stacked Volumes When Stacked Volume Support Is Enabled” on page 97.

Performing a Virtual Export of Logical Volumes

You can use DFSMSrmm subcommands to perform a virtual export for a private logical volume to an existing exported stacked volume container. A virtual export is when you use the DFSMSrmm subcommands rather than VTS export processing to export a volume. This is possible if you have imported a logical volume from a stacked volume, processed the logical volume only for input, and now want to re-associate the logical volume with the previously exported volumes on the original stacked volume container. You use the CHANGEVOLUME subcommand with the CONTAINER operand to do the virtual export. For example:

```
RMM CHANGEVOLUME V12345 CONTAINER(S26901)
```

DFSMSrmm changes the status of the logical volume to scratch in the library manager database prior to ejecting the volume. This results in a logical eject which deletes the logical volume from the library manager database. DFSMSrmm then adds the volume logically back into the container.

For the logical eject to work and delete the logical volume from the TCDB and the library manager data base, the scratch categories in the VTS must be defined with the ’Fast Ready’ attribute. If you do not use the fast ready attribute for your scratch categories, do not use virtual export. If you do so, you will receive messages CBR3650I and EDG3726I with error code ’06’ and the volume will not be deleted from the TCDB and the library manager data base.

Recovering a Logical Volume from an Exported Stacked Volume

DITTO is used to recover a logical volume from an exported stacked volume in situations where no VTS is available. DITTO does not go through OPEN processing nor issue regular mount messages. DITTO recovers the logical volume as a physical volume. As a result, managing these recovered logical volumes requires some additional consideration and processing.

Here are some considerations:

- When you specify DFSMSrmm EDGRMMxx REJECT command values to manage volumes, DFSMSrmm relies on information available at OPEN. These REJECT command values do not apply because DITTO does not issue OPEN requests.
- DFSMSrmm does not update existing volume information when DITTO copies the data.
- The DITTO output volume is a physical volume but might have the same volume serial number as the original logical volume. If you use the original logical volume serial number for the recovered volume, you can convert the logical volume in

the DFSMSrmm control data set to a physical volume by issuing the DFSMSrmm CHANGEVOLUME subcommand as shown in Figure 46

```
RMM CHANGEVOLUME volser TYPE(PHYSICAL) CONTAINER(' ')-
LOCATION(SHELF) CONFIRMMOVE FORCE
```

Figure 46. Converting A Logical Volume to a Physical Volume

Use the recovered volume as a normal physical volume. If the output volume serial number is different from the original logical volume serial number , the volume information, the data set information and other details can be reused by issuing the DFSMSrmm CHANGEVOLUME subcommand as shown in Figure 47.

```
RMM CHANGEVOLUME volser TYPE(PHYSICAL) CONTAINER(' ')-
LOCATION(SHELF) CONFIRMMOVE NEWVOLUME(pvolser)
```

Figure 47. Reusing Volume Information for a Recovered Volume

- You can also add data set information for the DITTO output volume from the original logical volume.

  To copy details for the original logical volume to the new physical volumes, use DFSMSrmm subcommands to obtain the data set and volume details for the volumes you want to redefine.

Setting Up DFSMSrmm for the System-Managed Tape Library

To get started with the system-managed tape library, information about the volumes in the system-managed tape library must be defined to DFSMSrmm. This section describes scenarios for getting volumes defined to DFSMSrmm.

Using the System-Managed Tape Library With New Volumes

If you intend to populate the system-managed tape library with new scratch volumes, you do not need to explicitly define them to DFSMSrmm. During entry processing, with DFSMSrmm active, DFSMSrmm automatically records information about each volume in its control data set. DFSMSrmm uses the defaults you specified in ISMF for the library entry values; you should set the default entry status to scratch. See “Partitioning System-Managed Tape Libraries” on page 109 for additional information.

Using the System-Managed Tape Library with Volumes Already Defined in DFSMSrmm

Volumes that are already defined to DFSMSrmm have location and other information known to DFSMSrmm. If you plan to use these volumes with the system-managed tape library, you need to update the volume location and home location to a system-managed library name. Here are two methods to implement the system-managed tape library with DFSMSrmm.

Method 1

1. For volumes that are going to a system-managed tape library, use the RMM CHANGEVOLUME subcommand with the LOCATION operand to change the location to the correct automated tape library name. This sets the automated tape library name as the destination name and home location name for each

volume going to an automated tape library. For volumes that are going to a manual tape library, specify a manual tape library name and you are now ready to use the volumes.

You can use the RMM SEARCHVOLUME subcommand with the CLIST operand to build a data set containing the required CHANGEVOLUME LOCATION requests.

```
RMM SEARCHVOLUME VOLUME(A*) LIMIT(*) -
    OWNER(*) LOCATION(SHELF) -
    CLIST('RMM CHANGEVOLUME ',' LOCATION(ATL1)')
```

This subcommand builds a command list that you can execute. Repeat this step for each storage location from which you intend to move volumes.

2. Run EDGHSKP with the PARM='REPTEXT' and use EDGRPTD to generate a list of volumes to insert into the automated tape libraries.
3. Insert the volumes. You are now ready to use the volumes.
4. To return volumes from storage locations to the system-managed tape library when the volumes no longer need to be retained, use the RMM CHANGEVOLUME subcommand to change the volume HOME location from SHELF to the library name.

   Example:

```
RMM SEARCHVOLUME VOLUME(A*) LIMIT(*) -
    OWNER(*) LOCATION(LOCAL) -
    CLIST('RMM CHANGEVOLUME ',' HOME(ATL1)')
```

Method 2

1. Pull all volumes that are to go into the system-managed tape library and physically load them.
2. Run the inventory process of the system-managed tape library and let it drive the adding of volumes to the volume catalog. This also flags the volume location as an automated tape library, but does not change the home location name.
3. Use the RMM SEARCHVOLUME subcommand with the CLIST operand to create a CLIST to create CHANGEVOLUME commands for all the volumes in the library to set the HOME location to the automated tape library. For example, specify:

```
RMM SEARCHVOLUME VOLUME(A*) LIMIT(*) -
    OWNER(*) LOCATION(ATL1) -
    CLIST('RMM CHANGEVOLUME ',' HOME(ATL1)')
```

   If you do not change the home location name, all private volumes are eligible for moving out of the automated tape library when you next run vital record processing as part of inventory management.
4. If you have volumes in storage locations that you want returned to the system-managed tape library when they no longer need to be retained, change the volume HOME location from SHELF to the library name, using the RMM CHANGEVOLUME subcommand. For example, specify:

```
RMM SEARCHVOLUME VOLUME(A*) LIMIT(*) -
    OWNER(*) LOCATION(LOCAL) -
    CLIST('RMM CHANGEVOLUME ',' HOME(ATL1)')
```

Using the System-Managed Tape Library with Existing Volumes

To use the system-managed tape library with existing volumes that are not defined to DFSMSrmm or if you are usingDFSMSrmm to manage these volumes for the first time, ensure that information about your volumes is recorded in the DFSMSrmm control data set.

You need to ensure that the correct status is recorded for each private and scratch volume you want to use with the system-managed tape library. One way to do this is to predefine all the private volumes. You can use any existing information you have for the volumes either from an existing tape management system or based on data set catalog entries. You use this information to build a list of RMM ADDVOLUME subcommands to define the volumes to DFSMSrmm. For example, for a private volume:

```
RMM ADDVOLUME volser LOCATION(ATL1) STATUS(MASTER) –
    OWNER(owner) STORGRP(storagegroup)
```

Adding rack numbers is optional. You can define shelf space first by using the ADDRACK subcommand so there are empty rack numbers for each volume you add.

Depending on the ISMF library entry defaults, you might be able to avoid defining scratch volumes to DFSMSrmm as this can happen automatically during entry processing.

Using DFSMSrmm with an Existing Automated Tape Library

If you have been using an automated tape library without DFSMSrmm, volume status information and retention requirements from the TCDB must be recorded in the DFSMSrmm control data set for private and scratch volumes in the automated tape library.

Assuming that you are using a single range of volume serial numbers that start with S00000 in the automated tape library, follow these steps to obtain the information:

1. Adding shelf space is optional for physical volumes and not supported for logical volumes. Define shelf space for the volumes to DFSMSrmm using the RMM ADDRACK subcommand:

   ```
   RMM ADDRACK S00000 COUNT(5000)
   ```

2. Use the RMM ADDVOLUME subcommand to define the volumes to DFSMSrmm and get the correct status information from the volume catalog. You can provide an owner on the RMM ADDVOLUME subcommand. If you do not specify the EXPDT operand, DFSMSrmm obtains the expiration date from the volume catalog. If there is no date in the volume catalog, DFSMSrmm uses the DFSMSrmm parmlib RETPD default.

   ```
    RMM ADDVOLUME S00000 COUNT(5000) STATUS(VOLCAT) OWNER(owner) -
        STATUS(USER) EXPDT(yyyy/ddd)
   ```

   If you do not provide an owner, DFSMSrmm assigns a default owner as described in *z/OS DFSMSrmm Guide and Reference*. If you are not using a single range of volumes in the automated tape library, issue multiple ADDVOLUME subcommand requests to define all volumes.

It is likely that while implementing DFSMSrmm with the automated tape library you are converting from an existing tape management system. If so, add this step to your overall conversion plan. Instead of using the RMM ADDVOLUME subcommand to define the volumes, use information from your existing tape management system to define the volumes to DFSMSrmm during the conversion.

Returning Volumes to Scratch Status

When a volume in a system-managed tape library is returning to scratch status, DFSMSrmm can inform OAM during expiration processing for each volume being changed to scratch status. This function also allows DFSMShsm volumes to return to the scratch category.

DFSMSrmm informs OAM to update the TCDB volume status during expiration processing or when the status of a volume is changed using the RMM CHANGEVOLUME subcommand based on the DFSMSrmm EDGRMMxx parmlib OPTION SMSTAPE operand value you specify. See “Defining System Options: OPTION” on page 134 for information about the DFSMSrmm EDGRMMxx parmlib OPTION SMSTAPE operand.

DFSMSrmm uses the OAM status change exit, CBRUXCUA, to allow DFSMSrmm to be notified of all volume status changes made by others, for example, by ISMF. CBRUXCUA performs the following processing:

- If a volume is not defined to DFSMSrmm, but is resident in an automated tape library, it is automatically defined to DFSMSrmm. If any errors are encountered, as for cartridge entry processing, the status change is rejected.
- If a volume is defined to DFSMSrmm:
  - All changes from PRIVATE to SCRATCH status are rejected unless the DFSMSrmm control data set already shows SCRATCH status.
  - All changes from PRIVATE to PRIVATE and SCRATCH to SCRATCH are supported to enable OAM to correct discrepancies that might exist with the library manager inventory.
  - All changes from SCRATCH to PRIVATE status are accepted. This includes open processing.
  - Storage group name changes are recorded in the DFSMSrmm control data set.
  - Any change to make TCDB information consistent with DFSMSrmm information is accepted.

  The objective is to keep the TCDB and the DFSMSrmm control data set information consistent.

Recommendation: Use the STGADMIN.IGG.LIBRARY resources to protect the new catalog facilities define, alter, and delete of library entries, and volume entries to ensure changes go through the OAM installation exits. Use IDCAMS as an error recovery tool.

Partitioning System-Managed Tape Libraries

You can partition a system-managed library including a VTS by performing the following tasks:

- Specify the USE operand value on the RMM ADDVOLUME or RMM CHANGEVOLUME subcommands. You can set this value to MVS or VM or both. If you do not specify MVS for a volume, DFSMSrmm prevents the volume from being defined in the volume catalog on this system.
- Define parmlib member EDGRMMxx REJECT prefixes as described in “Defining Tapes Not Available on Systems: REJECT” on page 157. You can use REJECT to prevent a volume not defined to DFSMSrmm from being defined in a system-managed tape library. The REJECT ANYUSE(*prefix*) operand prevents a volume from being defined in the system-managed tape library on the current

system. The REJECT OUTPUT(*prefix*) operand allows you to define the volume to the system-managed tape library but only use the volume for input processing.

When you enter a volume into a system-managed tape library, if the volume is defined to DFSMSrmm and you have specified the USE operand without MVS, or the volume matches a specified REJECT ANYUSE(*prefix*), EDGLCSUX sets a return code of 12 to pass to OAM. OAM leaves the volume in the system-managed tape library in the INSERT category; it does not create a volume entry in the TCDB. The volume is then available for another sharing system to process the insert request. The sharing system could be another VM or MVS system.

If DFSMSrmm allows the volume entry to be performed, OAM creates an entry in the TCDB. If the volume matches a specified REJECT OUTPUT(*prefix*), at OPEN time DFSMSrmm fails any requests for output processing while allowing requests for input processing.

To tailor partitioning when the REJECT ANYUSE option cannot be used, add code to the DFSMSrmm-supplied OAM installation exit to check the volume serial number and set a return code of 12 if the volume is not for use on this system. See “Managing System-Managed Tape Library Volumes: EDGLCSUX” on page 202 for information about the DFSMSrmm EDGLCSUX installation exit.

Sharing a System-Managed Library and a BTLS-Managed Library

When you share an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) between DFSMS and BTLS, there are restrictions on the sharing of volumes between the systems. For example, although a private volume is defined in the TCDB on DFSMS, it cannot be shared unless it is also defined in the BTLS catalog.

Consider partitioning the IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) using some volumes under DFSMS and other volumes under BTLS. This is important when you plan to use scratch volumes. Volumes that are part of scratch pools cannot be effectively shared.

Only volumes that are long-term private volumes can be shared effectively. You can define each private volume to both systems, but if the volume changes status, it is likely that the BTLS catalog and the TCDB will not match the volume status defined in the DFSMSrmm control data set. You can modify the CBRUXENT exit supplied with DFSMSrmm to force the Library Control System to not process the volumes intended for BTLS management.

If you want to designate specific scratch volumes for use on DFSMS and others for use with BTLS:

1. Modify the DFSMSrmm supplied CBRUXENT exit by setting the return code to 12 (UXEIGNOR) for all volumes that are to be managed by BTLS.
2. Use the AMS LIBRARY SETCATEGORY command to set the appropriate BTLS private or scratch category for all volumes entering the IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) that are ignored by the Library Control System.

If you modify the CBRUXENT exit, issue a WTO that you trap in Netview or equivalent. Use this event to trigger the start of the AMS LIBRARY SETCATEGORY command on the system where BTLS resides.

An easier implementation would be possible using separate control data sets, one for DFSMSrmm with system-managed tapes, and one for DFSMSrmm with BTLS volumes. The disadvantage of this is extra administration and total segregation of volumes.

Moving from a Non-System-Managed to a System-Managed IBM TotalStorage Enterprise Automated Tape Library (3494) or an IBM TotalStorage Enterprise Automated Tape Library (3495)

When moving volumes from an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) controlled by BTLS to a system-managed IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) managed with DFSMSrmm, update the DFSMSrmm control data set to reflect the new locations for the volumes. Issue the RMM CHANGEVOLUME subcommand on the DFSMSrmm system to update both the home location and current location of the volumes.

```
RMM CHANGEVOLUME volser LOCATION(sms_lib_name)
```

To return volumes from a storage location to the system-managed IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) instead of the non-system-managed IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495), use the RMM CHANGEVOLUME subcommand to change the home location for the volumes.

```
RMM CHANGEVOLUME volser HOME(sms_lib_name)
```

See “Defining Pools: VLPOOL” on page 162 for information about retaining and moving volumes.

Chapter 6. Running DFSMSrmm with BTLS

You can use DFSMSrmm with Basic Tape Library Support (BTLS). BTLS is an IBM program offering that provides basic automation support for the IBM TotalStorage Enterprise Automated Tape Library (3494) and IBM TotalStorage Enterprise Automated Tape Library (3495) in a non-system-managed library environment. DFSMSrmm does not interact directly with BTLS or the IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495), so you must update the BTLS catalog to reflect changes to volumes that are managed by DFSMSrmm.

DFSMSrmm adds information to the DFSMSrmm control data set when you define volumes to DFSMSrmm. If you plan to use any of the volumes defined to DFSMSrmm with BTLS, use the access methods services LIBRARY command to define the volumes in the BTLS catalog. Refer to the *Basic Tape Library Support Version 1 Release 1 User's Guide and Reference* for more information.

If you plan to use DFSMSrmm for volumes managed by BTLS, set up procedures to return scratch volumes to scratch status in the BTLS catalog after DFSMSrmm expiration processing.

Here is a summary of the steps you follow to use BTLS with DFSMSrmm:

1. Use the NAME operand on the VLPOOL parmlib command to identify the BTLS scratch pools to be used.
2. Optionally, if you use data set name and jobname for your volume pools, use the EDGUX100 installation exit to select a pool for new tape data sets.
3. Set up the procedures to return BTLS-managed volumes to scratch status after you run inventory management.
4. Set up the procedures to update BTLS when volumes are added to or removed from the installation media library.

Setting Up Scratch Pools for BTLS-Managed Volumes

"Defining Pools: VLPOOL" on page 162 provides information on defining VLPOOL and other EDGRMMxx options.

Both DFRMM and DFSMSrmm use VLPOOL scratch pool definitions to perform the following functions:

- Update mount messages.
- Update 3480 and 3490 drive displays if the MSGDISP installation exit IGXMSGEX is called.
- Reject scratch volumes which are not from the correct pools.

If you plan to use DFSMSrmm with BTLS, you must define and use scratch pools with special care to prevent DFSMSrmm from rejecting volumes needlessly. DFSMSrmm does not know that volumes managed by BTLS reside in an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) and attempts to control scratch tape assignment for mounts inside the IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495). DFSMSrmm, on the other hand, does not attempt to control scratch tape assignment for mounts inside a system-managed IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495).

If you have scratch volumes that reside in an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) managed by BTLS, define pool definitions to include all the volumes that you want to use. DFSMSrmm accepts or rejects volumes that are based on the VLPOOL definitions that you provide in parmlib member, EDGRMMxx. For volumes that reside in an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) and are managed by BTLS, ensure that the VLPOOL definitions do not cause conflicts when scratch volumes are used.

For example, consider these VLPOOL definitions:

```
VLPOOL PREFIX(A*) SYSTEM(A) TYPE(S)
VLPOOL PREFIX(B*) SYSTEM(B) TYPE(S)
```

On system *A* when a scratch volume is requested, if the IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) contains volumes from both pools A* and B*, it can select a scratch volume from either pool. If the Library Manager selects a volume in pool B*, DFSMSrmm rejects it. The Library Manager might *never* select a volume that can be used because all available scratch volumes are selected until a volume that is acceptable to DFSMSrmm is found.

If you want DFSMSrmm to manage pool selection when using BTLS, use the NAME operand on the VLPOOL definition for the pool to specify a pool name. For example, you could define VLPOOL definitions as follows:

```
VLPOOL PREFIX(A*) NAME(SCRTCH5) TYPE(S) SYSID(SY1) DESC('BTLS pool 5')
VLPOOL PREFIX(B*) NAME(SCRTCH3) TYPE(R) DESC('BTLS pool 3')
```

DFSMSrmm uses the value in the NAME operand to update messages and tape drive displays for non-specific mount requests. Do not use the NAME operand if your operations depend on a pool prefix rather than a pool name.

For scratch pools of volumes that are contained in a IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) that are managed by BTLS, use the NAME operand when specifying the SYSID operand on VLPOOL definitions. For more information, refer to “Returning BTLS-managed Volumes to Scratch” on page 116.

Running DFSMSrmm Inventory Management with BTLS

Use the EDGHSKP utility to run inventory management activities which include: vital record processing, expiration processing, storage location management processing, backing up the control data set and journal and creating an extract data set. See Chapter 14, “Performing Inventory Management,” on page 275 for more information.

Consider how to perform inventory management if you are using DFSMSrmm with BTLS. You need to complete the update to the BTLS catalog after inventory management is completed. Use the Access Method Services (AMS) LIBRARY SETCATEGORY command to update the status of the volumes managed by BTLS.

For example, to keep track of volumes that are managed by BTLS and to update information about volumes that are returning to scratch in the BTLS catalog, perform the following tasks:

1. Put all your BTLS-managed volumes in racks with a specific media name, for example: BTLS. Any other method to identify BTLS volumes, such as rack number or volume prefix could be used instead.
2. After inventory management is complete, issue the following command to create a list of BTLS racks in scratch status.
   ```
   RMM SEARCHRACK RACK(*) LIMIT(*) MEDIANAME(BTLS) SCRATCH -
       CLIST NOLIST
   ```
3. Use the resulting data set with the scratch volume list as the LIBIN DD input to an IDCAMS job with the following command:
   ```
   LIBRARY SETCATEGORY UNIT(xxx) CATEGORY(SCRTCH)
   ```

Running EDGINERS for BTLS-managed Volumes

For volumes that are managed only by BTLS, use EDGINERS. Use the TAPE DD to allocate a tape drive in the IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495). Use the POOL parameter or any other execution parameter to restrict processing to specific volumes that are managed by BTLS.

Restrictions

DFSMSrmm does not ensure that the volumes you use with BTLS in the IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) meet the MVS labeling restrictions that apply for system-managed tape volumes. You must make sure that only MVS standard label volumes with the same external and internal volume serial number are entered into the IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495). Failure to do so can result in incorrect processing by DFSMSrmm. For example, DFSMSrmm records the internal volume serial number of the mounted volume in the DFSMSrmm control data set, while BTLS uses the external volume serial number to request a mount.

When you migrate from BTLS management to system-managed tape, migration is easier if you use only MVS standard label volumes. Also, under system-managed tape processing, DFSMSrmm ensures that the volume serial number and rack number match, rejecting volumes that do not meet this requirement. Considering these restrictions during BTLS implementation will make migration to system-managed tape easier.

Defining Volume Information for BTLS-managed Volumes

DFSMSrmm does not interface with BTLS so volumes are not automatically defined to the BTLS catalog. If you plan to use any of the volumes defined to DFSMSrmm in an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) managed by BTLS, you must also define the volumes in the BTLS catalog using the AMS LIBRARY SETCATEGORY command.

When you delete volumes from DFSMSrmm, if the volumes are managed by BTLS you also need to ensure the volumes are deleted from the BTLS catalog and ejected from the IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495), if appropriate.

Returning BTLS-managed Volumes to Scratch

When volumes that reside in an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) are returned to scratch by DFSMSrmm, information needs to be updated in the TCDB or the BTLS catalog. DFSMSrmm provides no support to automatically update the BTLS catalog.

If you perform scratch management of volumes that reside in an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) that are managed by BTLS, ensure that BTLS volume status information matches DFSMSrmm volume status information. After you run expiration processing, prepare a list of volumes that are in scratch status. Issue the LIBRARY SETCATEGORY command to update the status of each volume to scratch. Issue the following command for each volume even if the status of the volume is scratch:

```
LIBRARY SETCATEGORY UNIT(addr) VOLSER(volser) CATEGORY(SCRTCH)
```

You can use DFSMSrmm to help build the input to the SETCATEGORY function. The TMP step shown in Figure 48 uses the RMM SEARCHVOLUME subcommand to produce a simple list of scratch volumes in the CLIST data set. This list is then input to IDCAMS in the LIBIN DD file. You can tailor the RMM SEARCHVOLUME subcommand to list volumes that are based on more selective criteria.

```
//jobname JOB ......
//TMP      EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
RMM SEARCHVOLUME VOL(*) OWN(*) STATUS(SCRATCH) LIMIT(*) -
 CLIST
/*
//AMS      EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//LIBIN DD DSN=userid.EXEC.RMM.CLIST,DISP=SHR
//SYSIN DD *
  LIBRARY SETCATEGORY UNIT(addr) CATEGORY(SCRTCH)
/*
```

Figure 48. Sample JCL to Return Volumes to Scratch

You need to modify this example if you are using system-managed tape with DFSMS and BTLS-managed volumes. Limit the search to volumes that are in the BTLS scratch pool. Otherwise all scratch volumes will become BTLS scratch and will not be available for use as scratch on the DFSMS system. Refer to *Basic Tape Library Support Version 1 Release 1 User's Guide and Reference* for more information on using the LIBRARY command.

Chapter 7. Managing Storage Locations

Storage locations are those places outside the removable media library where you send removable media. Storage locations can be destinations for disaster recovery related activities or any other purpose your installation chooses.

DFSMSrmm provides shelf-management of storage locations by assigning bin numbers to shelf locations within a storage location. You can specify that DFSMSrmm make bins available for reuse when volume moves have started or make bins available only when volume moves have been confirmed.

DFSMSrmm automatically provides shelf-management for the built-in locations. This means that DFSMSrmm assigns bin numbers to each volume in a built-in storage location. You can request shelf-management of installation defined storage locations when you create your location definitions using the parmlib LOCDEF command.

You can also specify media names in the parmlib LOCDEF command to provide a way to segregate shelf locations in an installation defined storage location. The media names you specify for a storage location should be the same as or a subset of the media names you specify for pools in your removable media library. See "Defining Pools in Parmlib Member EDGRMMxx" on page 67 for information about pooling.

The movement and retention of volumes in and out of storage locations is done during DFSMSrmm inventory management. During vital record processing, DFSMSrmm sets the required location for each volume using information from vital record specifications. The volume does not move to the required location during vital record processing. During storage location management processing, DFSMSrmm sets the destination for a volume using the required location information if that volume can be moved. If a move is required, and the move is to a shelf-managed storage location, DFSMSrmm assigns an empty bin in the target location to the volume based on the media name and LOCDEF information. If no empty bin numbers of the required media name are available, DFSMSrmm issues message EDG2403E, and inventory management processing continues.

When the volume is returned from the storage location, the bin number is identified for reuse as part of move confirmation processing. See Chapter 14, "Performing Inventory Management," on page 275 for information about inventory management vital record processing and storage location management.

When stacked volume support is enabled, DFSMSrmm sets the destination for a stacked volume at the completion of export processing. There is no destination set for logical volumes. Logical volumes use the stacked volume destination.

You can also override the inventory management processing of specific volumes by manually assigning destinations.

Types of Storage Locations

DFSMSrmm recognizes two types of storage locations: DFSMSrmm built-in storage locations and installation defined storage locations. There are three DFSMSrmm built-in storage locations named: LOCAL, DISTANT, and REMOTE.

Installation defined storage locations can be used for volumes for disaster or vital records, or for controlling any media moving outside your installation. Using DFSMSrmm installation defined storage locations, you can perform these tasks:

- Define more than three storage locations.
- Use any name up to 8 characters in length to name the storage location.
- Change the movement priority for the installation defined storage locations.
- Continue to use the DFSMSrmm built-in storage locations.
- Select shelf-management or no shelf-management for the installation defined storage locations.

Defining Storage Locations

Use the LOCDEF parmlib command to define installation defined storage locations. Storage locations can be any 1 to 8 character named location except for the DFSMSrmm reserved location names ALL, HOME, and CURRENT. You can also use the DFSMSrmm built-in storage location names as installation defined storage locations. You can also define characteristics of system-managed storage locations using the LOCDEF parmlib command. Table 15 shows the differences between built-in and installation defined storage locations.

Table 15. Differences between Built-in and Installation Defined Storage Locations

To	For Built-in Storage Locations	For Installation-defined Storage Locations
Segregate shelf locations by media name	You cannot segregate built-in storage locations by media name or type of media.	You can segregate shelf locations by specifying media names
Define storage locations	You do not need to define built-in storage location names. The three are:LOCAL, DISTANT, and REMOTE.	You can define an unlimited number of storage location names.
Define bin numbers	DFSMSrmm uses bin numbers 1 through 999999.	You can use any 6 character value.
Shelf-manage	DFSMSrmm automatically provides shelf-management.	You can decide.
Determine location priority	DFSMSrmm uses the default priority.	You can define the priority in the LOCDEF location definitions.

Implementing Installation Defined Storage Locations

You might implement installation defined storage locations for the following reasons:

- To choose the storage location names you want
- To change the dominance priority of the locations for inventory management vital record processing for use when moves conflict
- To separate volumes by shape when sending them to storage locations
- To use storage locations without shelf management
- To use more than 3 storage locations

To implement installation defined storage locations, follow these steps:

1. Identify:
   a. The number of storage locations you require. DFSMSrmm provides you with three built-in storage locations. If you use more than three storage locations, use the LOCDEF parmlib command to define additional storage locations. You can identify any location as an installation defined storage location except the locations ALL, HOME, and CURRENT which are DFSMSrmm reserved location names.
   b. The location names you will use.
   c. The priority you want to use for each location. Priority is used to resolve movement conflicts that occur when more than one policy applies to a volume or when multiple logical volumes reside on a stacked volume. The relative priority of the locations is used to determine where a volume is sent. Include the location name SHELF and any system-managed libraries to develop your location priority.
   d. The media names you will use in your installation. If you have different media shapes in your installation, you can set movement policies based on the different shapes. For example, you might have separate shelving for tape reels, cartridge tape, and optical media due to differences in their shape.
   e. If you require storage locations without shelf management.
2. Define LOCDEF parameters in parmlib. See “Defining Storage Locations: LOCDEF” on page 127.
3. Restart or refresh the DFSMSrmm procedure to use the updated parmlib member.
4. Define the bin numbers for the shelf-managed storage locations using the RMM ADDBIN subcommand. See *z/OS DFSMSrmm Guide and Reference* for information.

   When the new location and bin numbers are defined, they are available for assignment to volumes moving to the storage location. In order for DFSMSrmm to use the new storage locations and bin numbers for storage management, you must continue with step 5. Inventory management sequentially assigns the bin numbers to moving volumes. If you do not define vital record specifications and run inventory management, the locations and bin numbers can only be assigned manually by using the RMM CHANGEVOLUME subcommand.
5. Create new vital record specifications or update existing vital record specifications specifying the location names and media names you defined using LOCDEF.
6. Run inventory management vital record processing to produce a Vital Records Retention Report. Check the report to ensure that the correct destination is selected for each data set and volume retained by a vital record specification.
7. If you plan to export logical volumes, run export processing before running DFSMSrmm storage location management.
8. Once the retention report is correct, run inventory management storage location management and report extract processing to assign destinations and bin numbers and to prepare an extract data set which you use as input to EDGRPTD.
9. Run EDGRPTD to produce the movement and inventory reports for use to pull and ship the volumes to the correct locations.
10. Once volumes have been moved, use the RMM CHANGEVOLUME subcommand to confirm volume movement.

Implementing Storage Locations As Home Locations

When you use storage locations as the home location for volumes you can decide how volumes are shelf managed. Shelf management is required if you want volumes stored in a specific slot such as a rack number or a bin number. A shelf location is not required if the volume is stored in a robotic tape library.

To avoid using shelf locations, define the storage location using MANAGEMENTTYPE(NOBINS) and do not define rack numbers that match to the volume serial numbers.

To use the rack number as the shelf location, define the storage location using MANAGEMENTTYPE(NOBINS) and either, define rack numbers that match to the volume serial numbers or use the POOL operand when adding or moving volumes.

To use the bin number as a shelf location define the storage location using MANAGEMENTTYPE(BINS) and do not define rack numbers that match to the volume serial numbers.

To implement storage locations as home locations, follow these steps:

1. Update the DFSMSrmm PARMLIB LOCDEF commands to include TYPE(STORAGE,HOME) for those storage locations you want to also define as home locations.
2. Refresh DFSMSrmm parameters by issuing the command
   ```
   F DFRMM,M=xx
   ```
3. Use the RMM CHANGEVOLUME subcommand to set the home location of volumes to be assigned to a specific storage home location.
   ```
   RMM CHANGEVOLUME volser HOME(storname)
   ```
4. Use the RMM CHANGEVOLUME subcommand to set the current location of volumes already at the storage home location.
   ```
    RMM CHANGEVOLUME volser LOCATION(storname)
   ```

   If the storage location is not shelf-managed, you can include the CONFIRMMOVE operand to confirm the move is completed. If the storage location is shelf-managed, you must run EDGHSKP storage location management processing to assign shelf locations to the volumes. If you want specific shelf locations assigned, you can include the BIN operand on the RMM CHANGEVOLUME subcommand.

Managing Shelf Space for Home Locations

To assign or to change the assignment of shelf locations for volumes in the SHELF location or a system-managed library, use the RMM ADDVOLUME subcommand and the RMM CHANGEVOLUME subcommand with the RACK operand or the POOL operand. DFSMSrmm does not automatically initiate assignment of rack numbers as the shelf location for these volumes.

To automatically manage shelf space in a home location, use a shelf-managed storage location as the volume's home location. When a volume moves from or to a storage location, DFSMSrmm automatically assigns a bin number as the shelf location for the volume.

You can use either the rack number or the bin number as the shelf location for a volume. DFSMSrmm issues message EDG4013I at mount time when a volume is in a storage location, so that the location and bin number is available for the operator.

Reusing Bins in Storage Locations

If you enable extended bin support, you can specify that DFSMSrmm make bins available for reuse when volume moves have started. To make bins available for reuse when a volume move is started, specify the DFSMSrmm parmlib OPTION command REUSEBIN(STARTMOVE) operand. The default operand is REUSEBIN(CONFIRMMOVE), whether extended bin support is enabled or is not enabled. See “Defining System Options: OPTION” on page 134 for detailed information.

To enable extended bin support, create or update the control data set control record using EDGUTIL with the EXTENDEDBIN(YES) option. See “Creating or Updating the Control Data Set Control Record” on page 341 for detailed information.

Moving Volumes to Storage Locations

You can move volumes to storage locations using the following methods:

- By location. See “Moving Volumes by Location.”
- By media shape. See “Moving Volumes by Media Shape.”
- Manually. See “Moving Volumes Manually” on page 123.

Moving Volumes by Location

You can request that DFSMSrmm move volumes from specific locations by running the EDGHSKP utility and using the LOCATION parameter, specifying both the originating location and destination. If you use the INSEQUENCE parameter, DFSMSrmm assigns volumes to bins in sequential volume serial number order and bin number order.

Moving Volumes by Media Shape

You can describe removable media by their shape. For example, you can identify all round media, all square media, all small size media, or all cartridge media. You can use the VLPOOL parmlib command to define pools of media that are based on shape and to set the default media name. You can use media shape to identify the type of media that is allowed in a storage location. For example, you can keep tape reels, cartridges, and optical disks in different ranges of shelf space, where each type of media requires a differently shaped slot for storage.

When you decide which media names to use, consider the different media you have or might have in the future. You should the same media names for storage locations that you use for pools. Use the LOCDEF command MEDIANAME operand to define each media name for each location to allow or restrict storage using media name. You can also aggregate similar media to use the same range of shelf space by using a media name of *.

Consider an installation with the following types of media:

Mini tape reels
Tape reels
Cartridge system tape
Enhanced capacity cartridge system tape

There are four different types of media that fall into three basic media shapes: mini reels, reels, and cartridges. The number of reels in use in the installation is declining. Table 16 shows how the VLPOOL MEDIANAME and the LOCDEF MEDIANAME values can be defined. A different VLPOOL MEDIANAME has been defined for each different type of media. The media names describe the shape of the volumes or a physical characteristic like TWOTONE for enhanced capacity CST. In Table 16 (Case 1), the same LOCDEF media names are used to describe the media that can reside in the storage location. All volumes are segregated by their media name. In (Case 2), the cartridge system tape and enhanced capacity cartridge system tape are defined with the same media name and can be stored together because they are the same shape.

Table 16. Storing Media of the Same Shape

Type of Media	**VLPOOL MEDIANAME**	**LOCDEF MEDIANAME (Case 1)**	**LOCDEF MEDIANAME (Case 2)**
Mini tape reels	MINI	MINI	MINI
Tape reels	REELS	REELS	REELS
Cartridge system tape	CART	CART	*
Enhanced capacity CST	TWOTONE	TWOTONE	*

In Table 17, the media names REELS and CARTRDGE are used in the VLPOOL command to describe media shape. Each basic type of media has been allocated a different media name that gives information about the volumes. When the volumes are moved to a storage location, as shown in Table 17 (Case 3), all volumes are segregated by their media name. In (Case 4), all volumes are segregated by their media name but use is made of the * media name.

Table 17. Storing Media of Different Shapes

Type of Media	**VLPOOL MEDIANAME**	**LOCDEF MEDIANAME (Case 3)**	**LOCDEF MEDIANAME (Case 4)**
Mini tape reels	REELS	REELS	REELS
Tape reels	REELS	REELS	REELS
Cartridge system tape	CARTRDGE	CARTRDGE	*
Enhanced capacity CST	CARTRDGE	CARTRDGE	*

By careful selection of media names you can segregate shelf space in your removable media library using the VLPOOL parmlib command and in your storage locations using LOCDEF. See “Organizing the Library by Pools” on page 63 for information about the use of VLPOOL.

You can also use media name to control movement of volumes to non-shelf-managed storage locations. For example, you might have a customer that can only accept cartridge system tape. You can control the type of media sent to that customer by defining the LOCDEF command with a specific media name. In Figure 49 on page 123, only volumes with a media name of CART can be moved to the CUST1 location.

```
LOCDEF LOCATION(CUST1) TYPE(STORAGE) MANAGEMENTTYPE(NOBINS) -
   MEDIANAME(CART)
```

Figure 49. Using Media Name to Control Volume Movement

During inventory management DFSMSrmm checks the media name for the location and prevents volume movement when the media name does not match. DFSMSrmm issues message EDG2412E for each volume that cannot be moved because its media name is not supported at the location.

Moving Volumes Manually

You can override automatic processing and control volume movement manually by using the RMM CHANGEVOLUME subcommand with the MANUALMOVE operand. To return the volume to automatic movement control, use the RMM CHANGEVOLUME subcommand with the AUTOMOVE operand.

When you put a volume under manual move control, DFSMSrmm does not move the volume anywhere automatically, even when it expires and is pending release. Volume movement occurs only if you request it using the RMM CHANGEVOLUME subcommand with the LOCATION operand.

To allow release processing, you must remove the volume from manual move control unless the volume is in its home location. When a volume is in its home location, DFSMSrmm performs release processing even if the volume is under manual move control.

You might use manual move control to keep a volume on-site even though the volume is flagged to be sent off-site for disaster recovery. To keep the volume on-site or to request that the volume be moved back to its home location, you could use the following command:

```
RMM CHANGEVOLUME volser MANUALMOVE LOCATION(HOME)
```

Figure 50. Keeping Volumes On-site

When a volume is put under manual move control, any outstanding move is canceled. Moves can also be canceled by issuing the RMM CHANGEVOLUME command with the LOCATION operand. The operand LOCATION(HOME) is specified in Figure 50 to cancel any pending moves because the volume is in its home location.

You might use manual move control for volumes you create on one system and then send to other systems for processing. Define the other systems as locations using the parmlib LOCDEF command. When a volume is ready to be sent to the other system, you can confirm the volume move and put the volume under manual move control at the same time. For example, to send a volume to another system defined on a LOCDEF command as OTHER1, you could issue the following command:

```
RMM CHANGEVOLUME volser LOCATION(OTHER1) CONFIRMMOVE MANUALMOVE
```

Figure 51. Sending a Volume to Another System

The CONFIRMMOVE operand shown in Figure 51 confirms that the volume move has completed. The MANUALMOVE operand shown in Figure 51 puts the volume under manual move control and prevents the volume from being moved automatically. When the volume is returned from the other system, remove the volume from manual move control. Then confirm that the volume is back in its home location by issuing the following command.

```
RMM CHANGEVOLUME volser LOCATION(HOME) CONFIRMMOVE AUTOMOVE
```

Figure 52. Returning a Volume from Another System

Assigning Bins in Storage Locations

If you use the REASSIGN parameter, DFSMSrmm reassigns volumes to bins during storage location management processing. See “EXEC Parameters for EDGHSKP” on page 283 for a detailed description. Use both INSEQUENCE and REASSIGN and specify the DFSMSrmm parmlib OPTION command REUSEBIN(STARTMOVE) operand to maximize reuse of bins.

Changing Storage Locations

To change the bin management for a storage location information, follow these steps:

1. Identify the storage locations information you want to change and the vital record specifications that specify the storage location names.
2. Build RMM ADDVRS subcommands using the information from the vital record specifications you want to change. Later you will issue these subcommands to add the vital record specifications you delete. Build RMM DELETEVRS subcommands for each vital record specification that contains the storage location name you want to change.
3. Issue the RMM DELETEVRS subcommands in the background to delete all vital record specifications that use the storage location you want to change.
4. Update the DFSMSrmm EDGRMMxx parmlib member LOCDEF command MANAGEMENTTYPE operand for a storage location.
5. Restart the DFSMSrmm subsystem (F DFRMM,M=xx.) described in “Step 16: Restarting MVS with DFSMSrmm Implemented” on page 46.
6. Use the RMM ADDBIN subcommand to add bins required for the storage location if the new MANAGEMENTTYPE is BIN.
7. Issue the RMM ADDVRS subcommands in the background to add the vital record specifications that you previously deleted.
8. Perform inventory management vital record processing and storage location management to move volumes to the updated storage location.

Deleting Storage Locations

You can delete storage locations by removing the EDGRMMxx parmlib LOCDEF definition for the location. Before removing the LOCDEF definition in parmlib, check that:

- All bin numbers have been deleted in the location you want to delete.
- There are no volumes that are marked to be moved to the location.
- DFSMSrmm does not list any volumes for the location by using the RMM SEARCHVOLUME subcommand with the LOCATION operand.
- There are no vital record specifications that use the location you want to delete.

Switching Volumes to Installation Defined Storage Locations

You can switch volumes from built-in storage locations to the installation defined storage locations without moving the volumes.

Make sure you have defined the LOCDEF command in parmlib to identify the target installation defined storage location and its attributes as shown in Figure 53.

```
LOCDEF LOCATION(REMOTE) TYPE(STORAGE) MEDIANAME(CART) -
   MANAGEMENTTYPE(BINS)
```

Figure 53. Identifying an Installation Defined Storage Location

Issue the RMM CHANGEVOLUME subcommand with the BIN operand to assign bin numbers for the volumes you are switching from the built-in storage location to the installation defined storage location. For example, the volume MIKE01 resides in the built-in storage location REMOTE in bin number 000010. To switch the volume to the installation defined storage location REMOTE, issue:

```
RMM CHANGEVOLUME MIKE01 LOCATION(REMOTE) BIN(000010) CONFIRMMOVE
```

The volume is allocated to the bin number in the installation defined storage location and the original bin number in built-in location REMOTE is empty. To move all the volumes you can use the data in the report extract file to generate the RMM CHANGEVOLUME subcommands.

DFSMSrmm provides sample JCL as EDGJCVB in SAMPLIB. You can use this sample to switch the storage location name to any installation defined storage location name while keeping the assigned shelf location number the same. Substitute the LOCATION(REMOTE) with LOCATION(*storage_locname*) you choose.

Converting from Built-In Storage Locations

If you already have volumes stored in the DFSMSrmm built-in storage locations, you need to decide if you migrate the volumes to installation defined storage locations or allow the built-in storage locations to fall out of use when volumes are moved out and not replaced.

Once you decide to move to installation defined storage locations, whether or not you decide to continue to use the same names (LOCAL, DISTANT, and REMOTE), you must redefine the vital record specification definitions so that DFSMSrmm knows you are using installation defined storage locations.

The built-in storage locations LOCAL, DISTANT and REMOTE can be defined on LOCDEF parameters. This gives you the ability to use the existing built-in names rather than having to change the storage location names in use. When you use the DFSMSrmm built-in storage locations in a LOCDEF parmlib command, the DFSMSrmm built-in storage locations are treated like any other installation defined storage location.

You cannot use the LOCDEF command to change the PRIORITY of LOCAL, DISTANT or REMOTE and have them otherwise continue to work as before. Once you have defined the built-in names using LOCDEF:

- You can no longer use previously assigned bin numbers. You must define new bin numbers using the media names that are specified in the LOCDEF command.
- You must redefine the vital record specifications which reference these locations so that volumes can be scheduled to move to the installation defined location.

If the built-in storage location names are defined using LOCDEF without the PRIORITY operand, the default installation defined location priority is used.

Going Back to Built-In Storage Locations

If you converted built-in storage locations to installation defined storage locations as described in "Converting from Built-In Storage Locations" on page 125, you can go back to using the storage locations as built-in storage locations.

1. Remove the LOCDEF commands that use the built-in names.
2. Restart or refresh the DFSMSrmm procedure to use the updated parmlib member.
3. Define the bin numbers for the built-in storage locations using the RMM ADDBIN subcommand.
4. If LOCAL, DISTANT, or REMOTE were used as installation defined storage location names, any vital record specification using the names must be deleted and then redefined.
5. Run inventory management vital record processing to produce a Vital Records Retention report.
6. Run inventory management storage location management processing to assign destinations and bin numbers. Request a report extract file for EDGRPTD.
7. Run EDGRPTD to produce the movement and inventory reports for use to pull and ship the volumes to the correct locations.
8. Once volumes have been moved, use the RMM CHANGEVOLUME subcommand to confirm volume movement.
9. Delete the empty bins used for the installation-defined storage location.

Chapter 8. Using the Parmlib Member EDGRMMxx

This section describes the options that you can specify in the parmlib member EDGRMMxx.

Specify the options using the following format:

```
command operand1(value1) operand2(value2) -
        operand3(value3)
```

You can include comments in the parmlib member by enclosing your comments within /* */ as shown in the following example. Comments can precede and follow the parameters as well as appear within the parameters:

```
/*Your command syntax example*/
command operand1(value1) operand2(value2) /* comment */ -
        operand3(value3) /*end of command*/
```

The parmlib member EDGRMMxx uses the following commands:

- LOCDEF that is described in “Defining Storage Locations: LOCDEF.”
- MNTMSG that is described in “Defining Mount and Fetch Messages: MNTMSG” on page 132.
- OPTION that is described in “Defining System Options: OPTION” on page 134.
- REJECT that is described in “Defining Tapes Not Available on Systems: REJECT” on page 157.
- SECCLS that is described in “Defining Security Classes: SECCLS” on page 159.
- VLPOOL that is described in “Defining Pools: VLPOOL” on page 162.

Use the OPTION command to set defaults for DFSMSrmm on a system. Use the VLPOOL command to override the system-wide defaults for particular tape pools.

Do not specify duplicate operands. If you do, DFSMSrmm uses the last value you specified.

Defining Storage Locations: LOCDEF

Use the LOCDEF command to define installation defined storage locations to DFSMSrmm. You can also use LOCDEF to set the priority for shelf locations and system-managed tape libraries. The LOCDEF command in Figure 54 defines the location MIKESLOC, which accepts media with the media name SQUARE, and which is shelf-managed.

```
     /* LOCDEF - Add a location definition                  */
LOCDEF LOCATION(MIKESLOC) MEDIANAME(SQUARE) TYPE(STORAGE)  -
   MANAGEMENTTYPE(BINS)
```

Figure 54. Parmlib Member EDGRMMxx LOCDEF Command Example

LOCDEF Command Syntax

Figure 55 on page 128 shows the syntax of the LOCDEF Command for defining storage locations.

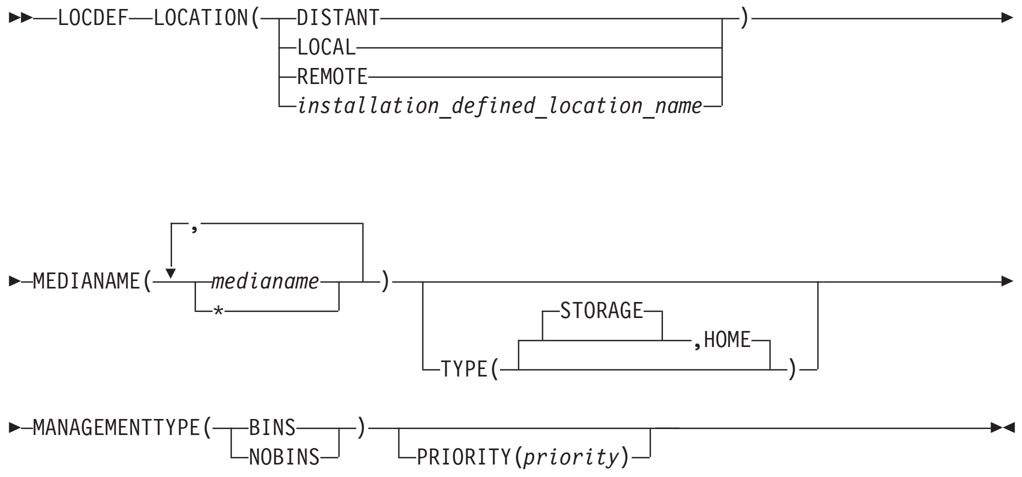


Figure 55. Parmlib Member EDGRMMxx LOCDEF Command for Defining Storage Locations

Figure 56 shows the syntax of the LOCDEF Command for defining shelf locations and system-managed libraries.

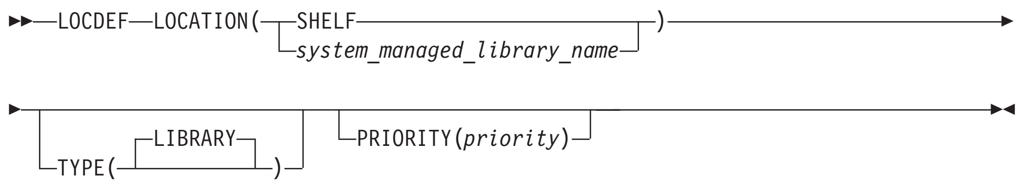


Figure 56. Parmlib Member EDGRMMxx LOCDEF Command for Defining Shelf and System-Managed Libraries

LOCDEF Command Operands

LOCATION(*installation_defined_location_name***|** *system_managed_library_name***|LOCAL|DISTANT|REMOTE|SHELF)**

Specifies the name of the location being defined. *installation_defined_location_name* can be any 1 to 8 character name or LOCAL, DISTANT, REMOTE, or SHELF. *system_managed_library_name* can be any 1 to 8 character name starting with a nonnumeric. The DFSMSrmm reserved location names HOME, CURRENT or ALL cannot be used as location names. When you use SHELF or a system-managed library, the only other operands that can be specified are PRIORITY and TYPE.

All LOCDEF location names must be unique. DFSMSrmm issues message EDG0225E if you define duplicate location names. You cannot have a storage location with the same name as a system-managed library. If you specify the same name, DFSMSrmm issues message EDG0233E. You cannot specify a distributed library name. If you specify a distributed library name, DFSMSrmm issues message EDG0235E.

MANAGEMENTTYPE(BINS|NOBINS)

Use this operand to identify the shelf-management technique you want for the location. This operand is required when defining a location of type STORAGE.

BINS DFSMSrmm shelf-manages the location by assigning bin numbers to volumes in the location.

NOBINS

Specifies that the location is not to be shelf-managed. Volumes sent to this location are not assigned bin numbers. However, they will still only be eligible to be sent to the location if their medianame or * appears in the LOCDEF media name list.

You can change the type of a location by specifying the LOCDEF command with the TYPE operand. The changed value applies to the vital record specifications you define after you make the change. To implement the change, you must change the location information in existing vital record specifications and run inventory management vital record processing. During vital record processing, DFSMSrmm changes the required location and location type for each volume. See “Changing Storage Locations” on page 124 for additional information.

Although vital record specifications must be updated and inventory management vital record processing must be run to implement the new LOCDEF MANAGEMENTTYPE, you can use the new LOCDEF MANAGEMENTTYPE when issuing DFSMSrmm CHANGEVOLUME subcommands with the LOCATION operand.

MEDIANAME(*medianame***|*)**

Specifies a list of the media names that are acceptable for the storage location. *medianame* can be any 1 to 8 character name you choose to describe a media name, type of media, a shape, or size. Examples of MEDIANAME include: CART, ROUND, SQUARE, 3490, 3590, TAPE, OPTICAL, CASSETTE, and so on. The media names you use on the LOCDEF commands should be the same as or a subset of the media names you use for your installation VLPOOL commands. See “Defining Pools: VLPOOL” on page 162.

When you specify MEDIANAME(*) using the parmlib LOCDEF command or the RMM ADDBIN or RMM ADDRACK subcommands, any volume with any media name can be sent to the location and DFSMSrmm does not segregate the volumes by shape. For example, if you specified a LOCDEF command with the media names:

```
LOCDEF LOCATION(MYLOC) MEDIANAME(3480,*)
```

then you could use the following media names on RMM ADDBIN subcommand requests:

```
RMM ADDBIN KG0002 LOCATION(MYLOC) MEDIANAME(3480)
```

and

```
RMM ADDBIN KG0100 LOCATION(MYLOC) MEDIANAME(*)
```

The example shown in Figure 57 is not valid because the MEDIANAME(3420) was not listed on the LOCDEF parameters for the storage location.

```
RMM ADDBIN KG0005 LOCATION(MYLOC) MEDIANAME(3420)
```

Figure 57. Using a Medianame Not Defined in the LOCDEF Command

PRIORITY(1-9999)

Defines the priority of this location relative to other locations. Lower numbers have higher priorities. PRIORITY is used to determine where to move a volume when a volume is assigned multiple destinations based on matching to two or more vital record specification definitions. PRIORITY is used to select a location in case of a move conflict. Move conflicts include:

- Multiple data sets with different required locations on one volume.
- Multiple volumes with different required locations in one volume set and the DFSMSrmm EDGRMMxx parmlib option is set for movement by volume set.
- Multiple logical volumes with different required locations on one stacked volume.

You can override this value by specifying a PRIORITY on the vital record specification. If PRIORITY is omitted, priority is based on the location type:

Priority	**Location Name or Type**
2000	Installation defined STORAGE type, including LOCAL, DISTANT, and REMOTE if they are specified on LOCDEF.
4800	AUTO automated tape libraries
4900	MANUAL manual tape libraries
5000	SHELF location

For each location you do not specify in a LOCDEF, the default priority is determined from the following list:

Priority	**Location Name or Type**
100	REMOTE location
200	DISTANT location
300	LOCAL location
2000	installation-defined STORAGE type
4800	AUTO automated tape libraries
4900	MANUAL manual tape libraries
5000	SHELF location

TYPE(STORAGE|LIBRARY)
Use this operand to identify the type of location you are defining. The value can be either STORAGE or LIBRARY. This operand is optional. If MEDIANAME or MANAGEMENTTYPE are specified, only TYPE(STORAGE) is valid. If you do not specify the TYPE operand DFSMSrmm sets a default value based on whether you specify the MEDIANAME or MANAGEMENTTYPE operands.

STORAGE,HOME
Specifies that the location is a storage location, either shelf-managed or non-shelf-managed. You can identify a storage location as a home location by specifying TYPE(STORAGE,HOME). When you identify a storage location as a home location, you can manage volumes in a storage location like volumes that reside in a LIBRARY location. You can:

- Schedule release actions for volumes when they return to their storage home location
- Return volumes to scratch status while they reside in their storage home location

LIBRARY
Specifies that the location is either SHELF or a system-managed library. The only reason to define these locations on LOCDEF parameters is to change the default priority for the location. For a LIBRARY type location the only LOCDEF operands you can use are PRIORITY and LOCATION.

You can change the shelf-management technique used in a location by specifying the LOCDEF command with the TYPE operand. The changed value applies to the vital record specifications you define after you make the change. To implement the change, you must change the location information in existing vital record specifications and run inventory management vital record processing. During vital record processing, DFSMSrmm changes the required location and location type for each volume.

Although vital record specifications must be updated and inventory management vital record processing must be run to implement the new LOCDEF TYPE value, you can use the new LOCDEF TYPE when issuing DFSMSrmm CHANGEVOLUME subcommands with the LOCATION operand.

Points on Usage:

1. A LOCDEF parameter is required in the DFSMSrmm parmlib member for each installation defined storage location name and when you want to define a movement priority for the location.

   If you have multiple systems sharing a control data set, use the same LOCDEF commands on all sharing systems. However, you only need to define storage locations to the systems where you will use any RMM ADDVRS, ADDBIN or CHANGEVOLUME subcommands, and where you plan to run inventory management. During storage location management processing, DFSMSrmm ensures that the storage locations used during vital record processing are defined by LOCDEF commands. If you specify a system-managed library name on a LOCDEF command, DFSMSrmm validates that it is a library which is defined on the current system. If your system-managed libraries are not defined on all systems, DFSMSrmm assumes a TYPE(LIBRARY) location is manual tape library if it is not defined as a system-managed library. During inventory management vital record selection processing, if no priority is obtained from a vital record specification, DFSMSrmm obtains the priority from the LOCDEF commands. If no LOCDEF command is specified and DFSMSrmm knows that the location is a system-managed library, DFSMSrmm uses the default priority for the location type. Otherwise DFSMSrmm uses a priority of 9999.
2. If a LOCDEF parameter is altered or removed, existing bin numbers which are now in a location which is not defined via LOCDEF, or which have a media name which is no longer listed under the LOCDEF MEDIANAME operand, are handled correctly. Bin numbers are freed up when a volume is moved from the storage location or can be removed using the RMM DELETEBIN subcommand. You cannot add more bin numbers nor assign the bin numbers to new volumes going to the location.
3. Changes to LOCDEF PRIORITY or MEDIANAME operands take effect during the first run of vital record processing after DFSMSrmm has been stopped and restarted, or the MODIFY command used to update the changed LOCDEF parameters in parmlib.
4. Changes to LOCDEF MANAGEMENTTYPE or TYPE operands only take effect after the vital record specifications that refer to the value have been redefined, or when using the LOCATION operand on the RMM ADDVOLUME or CHANGEVOLUME subcommands.
5. See Chapter 7, “Managing Storage Locations,” on page 117 for more information on LOCDEF and storage locations.

Defining Mount and Fetch Messages: MNTMSG

Use the MNTMSG command to tailor mount and fetch messages so they display the volume serial number, rack number, and pooling decision. The pooling decision can be a pool prefix, pool name, or storage group name. The operator can use this information to pull and mount volumes.

MNTMSG is an optional parmlib command. Specify a MNTMSG command for any message you want DFSMSrmm to update. When you specify the MNTMSG command, you must include all the MNTMSG operands.

DFSMSrmm inserts the required information within the message text or at the end of the message based on the value you use with the RACK operand. For nonspecific mount requests, if you have defined a pool name with the VLPOOL command, DFSMSrmm provides a pool name or storage group name rather than a pool prefix in the message. DFSMSrmm updates mount messages with a pool name only when DFSMSrmm updates messages at the end of the message text.

Use the VOLUME operand to define the position of the volume serial number in a message. Figure 58 shows examples of the parmlib member EDGRMMxx MNTMSG command specified with the VOLUME operand defined for 4-digit device numbers. If your definitions specify a volume serial number offset 1 less than the value shown in the examples, your definitions are from MVS supported 3-digit device numbers and should be updated to match the position of the volume serial number when 4-digit device numbers are used.

```
     /* MNTMSG - Add RACK= or POOL= at end of WTOs           */
MNTMSG  MSGID(IEF233A)      ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID(IEF233D)      ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID('IEF234E K')  ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID('IEF234E R')  ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID('IEF234E D')  ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID('IEF455D')    ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID(IEC501A)      ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID('IEC502E K')  ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID('IEC502E D')  ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID('IEC502E R ') ID(1) VOLUME(16) RACK(999)
MNTMSG  MSGID('IEC502E RD') ID(1) VOLUME(17) RACK(999)
MNTMSG  MSGID('IEC502E RK') ID(1) VOLUME(17) RACK(999)
MNTMSG  MSGID(IAT5110) ID(1) VOLUME(44) RACK(999)
MNTMSG  MSGID(IAT5210) ID(1) VOLUME(50) RACK(999)
MNTMSG  MSGID(IAT5410) ID(1) VOLUME(20) RACK(999)
```

Figure 58. Parmlib Member EDGRMMxx MNTMSG Command Examples for 4-digit Devices

JES3 Considerations: When you use the JES3 IATUX71 exit, use the RACK operand to determine if you want the rack number to replace the volume serial number in the message or appended to the end of the message. If the RACK operand value indicates a position within the length of the message, DFSMSrmm replaces the volume serial number in the message with the required information. If the RACK operand value exceeds the message length, DFSMSrmm appends the information to the end of the message.

When the JES3 IATUX71 exit is not used, use the TYPE=MCS keyword on at least one CONSOLE statement in the JES3 parmlib. This activates JES3 MCS console support so JES3 issues messages as WTO messages rather than directly to JES3 consoles.

MNTMSG Command Syntax

Figure 59 shows the syntax of the MNTMSG command:

```
►►──MNTMSG──MSGID(nnnnnnnnnnnn)──ID(nnn)──RACK(─┬─nnn─┬─)──VOLUME(nnn)──────────►◄
                                                └─999─┘
```

Figure 59. Parmlib Member EDGRMMxx MNTMSG Command Syntax

DFSMSrmm supplies a sample set of mount messages for your use. You can define additional MNTMSG commands to include other IBM operator messages and messages produced by your installation.

For example, if you define:

```
MNTMSG MSGID(IEF233A) ID(1) VOLUME(16) RACK(999)
```

your mount message displays the following:

```
IEF233A M 0480,999003,,J1400001,S0300 - RACK = T14103
```

If you specify the following:

```
MNTMSG MSGID(IEF233A) ID(1) VOLUME(16) RACK(16)
```

your mount message displays the following:

```
IEF233A M 0480,T14103,,J1400001,S0300
```

If you have a non-specific mount request where you have defined scratch pool AB* for that system, and you specify:

```
MNTMSG MSGID(IEF233A) ID(1) VOLUME(16) RACK(999)
```

your mount message displays the following:

```
IEF233A M 0480,PRIVAT,,J1400001,S0300 - POOL = AB****
```

If you specify:

```
MNTMSG MSGID(IEF233A) ID(1) VOLUME(16) RACK(16)
```

your mount message displays the following:

```
IEF233A M 0480,AB****,,J1400001,S0300
```

MNTMSG Command Operands

ID(*nnn*)
Specifies the starting position of the message identifier. Specify a value between 1 and 128.

Default: None. You must specify ID when you define a MNTMSG command in parmlib.

MSGID(*nnnnnnnnnnnn*)
Specifies message text 1 to 12 characters long. Normally the text is the message number, but it can include additional characters and blanks. You must enclose the MSGID value in quotes if you use blanks or special characters.

Default: None. You must specify MSGID when you define a MNTMSG command in parmlib.

RACK(*nnn*|999)
Specifies the position to insert the rack number, pool prefix, pool

name, or storage group in the message. Specify a value between 1 and 128 to insert the value within the message text.

Use 999 if you want to add the value to the end of the message.

DFSMSrmm adds either RACK=rack_number or POOL=pool_value to the end of the message if there is enough space for the additional information. If you specify RACK(999), DFSMSrmm adds '- POOL=pool_value' if a pool name or storage group is substituted for a non-specific volume mount. If there is not enough space to add the information, DFSMSrmm overlays the end of the message text with the 6 to 8 character rack number or pool value preceded by a -. For a rack number, DFSMSrmm writes *nnnnnn*. For a pool prefix, DFSMSrmm writes *nnnnn**. For a pool name or storage group, DFSMSrmm writes *nnnnnnnn*.

- If you specify RACK(999), DFSMSrmm adds - RACK=*nnnnnn* if a rack number is added:
  ```
  IEF233A M 0480,999003,,J1400001,S0300 - RACK=T14103
  ```
- If you specify RACK(999), DFSMSrmm adds - POOL=*ppp** if a pool prefix is substituted for a non-specific volume mount:
  ```
  IEF233A M 0480,PRIVAT,,J1400001,S0300 - POOL=AB****
  ```
- If you specify RACK(999), DFSMSrmm adds - POOL=pool_value if a pool name is substituted for a non-specific volume mount.
  ```
  IEF233A M 0480,PRIVAT,,J1400001,S0300 - POOL=SCRTCH00
  ```

If there is not enough space to write the rack number or pool prefix, DFSMSrmm overwrites the end of the message text.

To display the pool prefix in the message, you must specify the SYSID operand on the VLPOOL command as described in "Defining Pools: VLPOOL" on page 162.

Default: None. You must specify RACK when you define a MNTMSG command in parmlib.

VOLUME(*nnn*) Specifies the position of the volume serial number in the message. Specify a value between 1 and 128.

Default: None. You must specify VOLUME when you define a MNTMSG command in parmlib.

Defining System Options: OPTION

Use the OPTION command to define the installation options for DFSMSrmm. Figure 60 on page 135 shows an example of the OPTION command and the operands that you can code in the parmlib member EDGRMM*xx*.

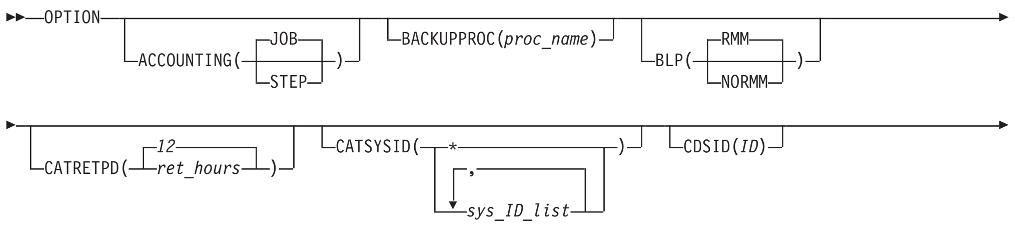
```
OPTION  OPMODE(P)                  /* protect mode        */   -
        ACCOUNTING(J)              /* Account information  */   -
        BACKUPPROC(RMMBKUP)        /* backup procedure     */   -
        BLP(RMM)                   /* bypass label process */   -
        CATRETPD(12)               /* catalog retention    */   -
        CATSYSID(SYSTEM1,SYSTEM2,SYSTEM3,SYSTEM4,SYSTEM5,       +
          SYSTEM6,SYSTEM7,SYSTEM8) /* CATALOG SYS          */   -
        CDSID(MVS2)                /* control data set ID  */   -
        COMMANDAUTH(OWNER)         /* security check       */   -
        DATEFORM(E)                /* European dates       */   -
        DISPDDNAME(DISPDD)         /* DD card name         */   -
        DISPMSGID(EDG4054I)        /* WTO message number   */   -
        DSNAME(RMM.CONTROL.DSET)   /* control data set     */   -
        IPLDATE(NO)                /* ipl date             */   -
        JOURNALFULL(75)            /* journal percentage   */   -
                                   /* threshold            */   -
        JRNLNAME(RMM.JOURNAL.DSET) /* journal              */   -
        LINECOUNT(60)              /* lines per page       */   -
        LOCALTASKS(10)             /* number of tasks      */   -
        MASTEROVERWRITE(LAST)      /* overwrite default    */   -
        MAXHOLD(100)               /* number of records    */   -
        MAXRETPD(NOLIMIT)          /* maximum retention    */   -
        MEDIANAME(3480)            /* media name           */   -
        MOVEBY(VOLUME)             /* movement processing  */   -
        MSG(M)                     /* message mixed case   */   -
        NOTIFY(N)                  /* no notification      */   -
        PDA(ON)                    /* PDA trace enabled    */   -
        PDABLKCT(255)              /* PDA block count      */   -
        PDABLKSZ(12)               /* PDA blocksize        */   -
        PDALOG(OFF)                /* Disable trace logging*/   -
        PREACS(NO)                 /* Preacs option        */   -
        RETAINBY(VOLUME)           /* retention processing */   -
        RETPD(5)                   /* default retention    */   -
        REUSEBIN(CONFIRMMOVE)      /* reuse bin            */   -
        SCRATCHPROC(RMMSCR)        /* scratch procedure    */   -
        SMFAUD(248)                /* SMF records          */   -
        SMFSEC(249)                /* SMF records          */   -
        SMSACS(NO)                 /* SMSACS option        */   -
        SMSTAPE(UPDATE(EXITS,SCRATCH,COMMAND),PURGE(YES))       -
                                   /* TCDB update and purge*/   -
        SYSID(DG4)                 /* system name          */   -
        TPRACF(AUTOMATIC)          /* automatic            */   -
        TVEXTPURGE(RELEASE)        /* EDGTVEXT action      */   -
        UNCATALOG(Y)               /* catalog option       */   -
        VRSCHANGE(VERIFY)          /* VRS change information*/  -
        VRSEL(NEW)                 /* VRS processing       */   -
        VRSJOBNAME(2)              /* retention by job name*/   -
        VRSMIN(1,FAIL)             /* VRS minimum          */
```

Figure 60. Parmlib Member EDGRMMxx OPTION Command Examples

OPTION Command Syntax

Figure 61 shows the syntax of the OPTION command:

Figure 61. Parmlib Member EDGRMMxx OPTION Command Syntax



```
►──┬─────────────────────────────────────────────────┬──┬────────────────────────────────────┬──────────►
   └─CLIENT(SERVERNAME(Servername) PORT(PortNumber))─┘  │           ┌─,──────────┐           │
                                                        │           ▼  ┌─OWNER─┐ │           │
                                                        └─COMMANDAUTH(─┴─DSN───┴─┴───────)───┘

►──┬───────────────────────┬──┬─────────────────────────┬──┬─────────────────────────────────┬──────────►
   │           ┌─JULIAN───┐ │  └─DISPDDNAME(DD_name)─┘  │            ┌─EDG4054I───┐        │
   └─DATEFORM(─┼──────────┼─)┘                          └─DISPMSGID(─┴─message_id─┴─)──────┘
               ├─EUROPEAN─┤
               ├─AMERICAN─┤
               └─ISO──────┘

►──┬───────────────────────────┬──┬─────────────────────┬──┬─────────────────────────┬──────────►
   │         ┌─MASTER DD─┐     │  │          ┌─NO──┐    │  │              ┌─75─┐     │
   └─DSNAME(─┴─name──────┴─)───┘  └─IPLDATE(─┴─YES─┴─)──┘  └─JOURNALFULL(─┴─nn─┴─)───┘

►──┬───────────────────────────────┬──┬────────────────────────┬──┬─────────────────────────────┬──────────►
   │           ┌─JOURNAL DD─┐      │  │            ┌─54──┐     │  │             ┌─10─────┐      │
   └─JRNLNAME(─┴─name───────┴─)────┘  └─LINECOUNT(─┴─nnn─┴─)───┘  └─LOCALTASKS(─┴─number─┴─)────┘

►──┬──────────────────────────────┬──┬──────────────────────┬──┬─────────────────────────────┬──────────►
   │                  ┌─LAST──┐   │  │          ┌─100─┐     │  │            ┌─NOLIMIT─┐      │
   └─MASTEROVERWRITE(─┼───────┼─)─┘  └─MAXHOLD(─┴─nnn─┴─)───┘  └─MAXRETPD(─┴─nnnn────┴─)────┘
                      ├─MATCH─┤
                      ├─USER──┤
                      └─ADD───┘

►──┬───────────────────────────────┬──┬──────────────────────────┬──┬─────────────────────┬──────────►
   │            ┌─3480────────┐    │  │         ┌─VOLUME─┐       │  │      ┌─MIXED─┐      │
   └─MEDIANAME(─┴─medianame───┴─)──┘  └─MOVEBY(─┴─SET────┴─)─────┘  └─MSG(─┴─UPPER─┴─)────┘

►──┬────────────────────┬──┬──────────────────────────┬──┬──────────────────┬──┬───────────────────────┬──────────►
   │         ┌─NO──┐    │  │         ┌─PROTECT─┐      │  │      ┌─ON───┐    │  │             ┌─255─┐   │
   └─NOTIFY(─┴─YES─┴─)──┘  └─OPMODE(─┼─────────┼─)────┘  └─PDA(─┼──────┼─)──┘  └─PDABLKCT(─┴─nnn─┴─)─┘
                                     ├─MANUAL──┤                ├─NONE─┤
                                     ├─RECORD──┤                └─OFF──┘
                                     └─WARNING─┘

►──┬──────────────────────────┬──┬───────────────────────┬──┬──────────────────────┬──┬──────────────────────────┬──────────►
   │            ┌─blksz─┐     │  │          ┌─ON──┐      │  │         ┌─NO──┐      │  │            ┌─VOLUME─┐    │
   └─PDABLKSZ(─┴─nn────┴─)───┘  └─PDALOG(─┴─OFF─┴─)────┘  └─PREACS(─┴─YES─┴─)────┘  └─RETAINBY(─┴─SET────┴─)──┘

►──┬──────────────────────┬──┬────────────────────────────────┬──┬──────────────────────────────┬──────────►
   │        ┌─5────┐      │  │           ┌─CONFIRMMOVE─┐      │  │              ┌─EDGXPROC──┐   │
   └─RETPD(─┴─nnnn─┴─)────┘  └─REUSEBIN(─┴─STARTMOVE───┴─)────┘  └─SCRATCHPROC(─┴─proc_name─┴─)─┘

►──┬────────────────────────────────────────────────────────┬──┬──────────────────────────────────────┬──────────►
   │                                       ┌─10─────┐       │  │          ┌─No SMF audit records─┐    │
   └─SERVER(PORT(port_number) SERVERTASKS(─┴─number─┴─))────┘  └─SMFAUD(─┴─nnn──────────────────┴─)──┘

►──┬──────────────────────────────────────────┬──┬──────────────────────┬──┬────────────────────────────────┬──────────►
   │         ┌─No SMF security records─┐      │  │         ┌─NO──┐      │  │        ┌─SMF System name─┐     │
   └─SMFSEC(─┴─nnn─────────────────────┴─)────┘  └─SMSACS(─┴─YES─┴─)────┘  └─SYSID(─┴─System name─────┴─)───┘

►──┬─────────────────────────────────────────────────────────────────────────────────────────────┬──────────►
   └─SMSTAPE(─┬───────────────────────────────────────────────────────────────────────────────┬─)─┘
              │                                                                  ┌─ASIS─┐        │
              └─UPDATE──(─┬─────────┬─┬───────────┬─┬───────────┬─)──PURGE──(─┼──────┼─)──────┘
                          └─EXITS───┘ └─SCRATCH───┘ └─COMMAND───┘                ├─NO───┤
                                                                                 └─YES──┘

►──┬─────────────────────────────┬──┬────────────────────────────┬──┬──────────────────────────┬──────────►
   │         ┌─NONE───────┐      │  │             ┌─RELEASE─┐    │  │            ┌─YES─────┐   │
   └─TPRACF(─┼────────────┼─)────┘  └─TVEXTPURGE(─┼─────────┼─)──┘  └─UNCATALOG(─┼─────────┼─)─┘
             ├─PREDEFINED─┤                       ├─NONE────┤                    ├─NO──────┤
             └─AUTOMATIC──┘                       └─EXPIRE──┘                    └─SCRATCH─┘

►──┬───────────────────────────┬──┬───────────────────────┬──┬──────────────────────────────────────────┬──────────►
   │            ┌─VERIFY─┐     │  │             ┌─2─┐     │  │         ┌─1──────────┐    ┌─FAIL─┐        │
   └─VRSCHANGE(─┴─INFO───┴─)───┘  └─VRSJOBNAME(─┴─1─┴─)───┘  └─VRSMIN(─┴─countrange─┴─,──┼──────┼─)──────┘
                                                                                         ├─WARN─┤
                                                                                         └─INFO─┘
```

```
►─┬──────────────────────┬─────────────────────────────────────────────────────►◄
  │          ┌─OLD─┐     │
  └─VRSEL(───┤     ├──)──┘
             └─NEW─┘
```

OPTION Command Operands

ACCOUNTING(JOB|STEP)
Specifies whether DFSMSrmm records JOB or STEP accounting information along with volume information.

JOB DFSMSrmm records the accounting information from the JOB statement of the JCL.

STEP DFSMSrmm records the accounting information from the EXEC statement of the JCL. If you specify STEP and there is no accounting information in the EXEC statement, DFSMSrmm records the JOB statement accounting information.

Default: ACCOUNTING(JOB).

BACKUPPROC
Specifies the name of the procedure that you want to be started automatically when the journal percentage full threshold is reached.

Specify a valid alphanumeric procedure name from 1 to 8 characters. If no name is specified, then no automatic start command is issued. See “Steps for Automating Control Data Set Backup and Journal Clearing” on page 314 for more information.

Default: None. DFSMSrmm ignores BACKUPPROC on the client system.

BLP(RMM|NORMM)
Specifies how DFSMSrmm controls bypass label processing (BLP).

Authorization to perform BLP is still dependent on the ICHBLP resource in the RACF FACILITY class, and the ability to use BLP can still be controlled via JES.

Note: DFSMSrmm allows BLP processing to continue if a volume is mounted with a VOL1 header label that matches the volume serial number specified in JCL or if the volume has no label. When a labeled volume is mounted and the volume serial number does not match the requested volume, DFSMSrmm prevents processing when either volume is defined to DFSMSrmm. To circumvent this DFSMSrmm processing, you can request that DFSMSrmm ignores the volume serial number as described in “Using EDGUX100 to Ignore Duplicate or Undefined Volume Serial Numbers” on page 225.

RMM
You can use BLP for input from and output to volumes in user status, and for input from volumes in master status.

NORMM
You can use BLP under the normal system controls and DFSMSrmm records the activities you perform on tapes. You can also use BLP for input from and output to volumes in user and master status, and for output to scratch tapes. BLP can be used for reading and writing of master and user status tapes and for output to scratch tapes. BLP read of scratch tapes is not supported.

For scratch tapes written using BLP, DFSMSrmm changes the volume to master status and sets the initialize release action so that the tape is correctly labeled on return to scratch. DFSMSrmm also overrides the logical volume serial number generated by OPEN for BLP output to scratch tapes, so that the correct volume serial number is used for cataloging of data sets.

DFSMSrmm does not allow no label (NL) tapes to be mounted in response to a scratch request. However, you can use BLP to create NL tapes during scratch processing.

Default: BLP(RMM)

CATRETPD(*ret_hours***)**

Specifies the number of hours from creation that a data set should be retained if it has not been cataloged and matches a vital record specification with the WHILECATALOG operand.

DFSMSrmm retains the data set for the catalog retention period if the data set has never been cataloged. DFSMSrmm does not retain the data set if DFSMSrmm detected that the data set was cataloged and then uncataloged during the catalog retention period.

ret_hours is 0-9999 hours. For example, CATRETPD(24) keeps data sets for 24 hours. Set *ret_hours* to 0 to request that DFSMSrmm does not perform this processing.

Default: CATRETPD(12).

CATSYSID(*|*sys_ID_list***)**

Specify CATSYSID to enable DFSMSrmm catalog synchronization. Use the CATSYSID operand to identify the user catalogs you want tracked when you run the DFSMSrmm EDGHSKP utility with the CATSYNCH operand to exploit catalog status tracking. All the systems you identify must have the code that supports catalog synchronization. See“Running DFSMSrmm Catalog Synchronization” on page 305 for more information.

CATSYSID(*) means that all catalogs are fully shared. You must specify an * to specify that catalogs are fully shared so that any data set can be processed by DFSMSrmm on any DFSMSrmm subsystem.

CATSYSID(sys_ID_list) provides a list of DFSMSrmm system IDs that share tape data set user catalogs with this DFSMSrmm subsystem. You can identify up to 16 system IDs for DFSMSrmm subsystems. You must specify a list of system IDs when user catalogs are not shared and must ensure they are synchronized with the DFSMSrmm control data set before you run inventory management vital record processing. Be sure to include IDs for systems that you no longer use but for which you are still retaining tape data sets.

Default: None.

CDSID(*ID***)**

Specifies the identifier of the control data set that must be used on this system. Specify a value one to eight alphanumeric characters long.

When you start DFSMSrmm, the CDSID ID is compared to the ID in the control data set control record. If the IDs match, DFSMSrmm startup continues. If the control data set does not have an ID, DFSMSrmm creates the ID in the control record from the CDSID. If the IDs do not match, DFSMSrmm startup fails and DFSMSrmm issues a message to the operator to select another parmlib member.

If you do not specify a value for CDSID, you cannot use a control data set that already has an ID in its control record. See “Creating or Updating the Control Data Set Control Record” on page 341 for information about how the DFSMSrmm EDGUTIL utility sets the control data set ID.

Default: None.

CLIENT(SERVERNAME(ServerName) PORT(PortNumber))
Specifies the type of system you want to set up. CLIENT is mutually exclusive with SERVER. If neither client nor server are specified, DFSMSrmm starts as a standard system

SERVERNAME(*servername***)**
The *servername* can either be an IP address, a fully qualified domain name, or a server host name. DFSMSrmm uses the domain name system (DNS) to resolve a domain name or a host name into an IP address. SERVERNAME is a required operand when you specify CLIENT. *servername* can be 63 alphanumeric characters, period (.), and hyphen (-). The host name can be a maximum of 63 characters. The host name must contain one or more tokens separated by a period. Each token must be larger than one character. The first character in each token must start with a letter. The remaining characters in each token can be a letter, number, or hyphen. For example, CLIENT(SERVERNAME(RMMPLX1.MAINZ.IBM.COM) PORT(1950)) tells DFSMSrmm to start as a client without direct DASD access and to share the tape inventory in access by the RMM server with the host name RMMPLX1 using network IP protocol port 1950.

PORT(*PortNumber***)**
Use this operand to specify the port number to be used for IP communication. The PORT operand is required. Specify a value from 1024 to 65535. Port numbers 1 to 1023 are reserved. Also, the client port number and server port number must match for the systems to communicate.

Default: None.

COMMANDAUTH(OWNER DSN)
Specifies the type of authorization that DFSMSrmm is to check. Specify OWNER when you expect the owners of volume information and data set information to be able to update their own data sets and volumes using RMM TSO subcommands. Specify DSN when you expect changes to volume and data set information to be authorized using the RACF DATASET class and TAPEVOL class.

You can set up authorization so that DFSMSrmm checks for authorization by owner first, and then checks for authorization using the DATASET class and TAPEVOL class. Set up this type of checking by specifying both the OWNER operand and the DSN operand separated by a comma. When the RACF name-hiding function is enabled in the system, this overrides the DFSMSrmm COMMANDAUTH processing. DFSMSrmm processing, when the name-hiding function is enabled, is the same as COMMANDAUTH(DSN).

Default: COMMANDAUTH(OWNER)

DATEFORM(AMERICAN|EUROPEAN|ISO|JULIAN)
Specifies the date format for messages and reports. See "EXEC Parameters for EDGHSKP" on page 283 for information about setting different date formats for reports.

Value	Language	Format	Example
A	American	mm/dd/yyyy	12/15/1994
E	European	dd/mm/yyyy	15/12/1994
I	ISO	yyyy/mm/dd	1994/12/15
J	Julian	yyyy/ddd	1994/349

Default: DATEFORM(JULIAN)

DISPDDNAME(*DD_name*)
Specifies the name of the DD card which identifies the data set that contains disposition control statements that DFSMSrmm processes during CLOSE or EOV processing. The data set must be a sequential file and must be defined with LRECL 80. The data set can be a member of a partitioned data set. When you specify the DISPDDNAME operand, you are requesting that DFSMSrmm performs disposition processing during CLOSE or EOV processing.

If you code DISPDDNAME, you must provide the name of the DD card that identifies the data set containing the disposition control statements optionally included in your JCL.

| For information about DFSMSrmm disposition processing, see Chapter 19,
| "Setting Up DFSMSrmm Disposition Processing," on page 389.

Default: None.

DISPMSGID(*message_id*)
Specifies the message number that DFSMSrmm uses for write–to–operator messages specified in the disposition control file.

EDG4054I	The message text provides the device number, volume serial number, volume sequence, the location where the volume is to move, and any message text you defined in the disposition control file.
message_id	You can define any alphanumeric value of up to eight characters.

| For information about DFSMSrmm disposition processing, see Chapter 19,
| "Setting Up DFSMSrmm Disposition Processing," on page 389.

Default: Message EDG4054I.

DSNAME(*name*)
Specifies the name of the DFSMSrmm control data set. Specify a name up to 44 characters long.

If you do not specify DSNAME, you must specify the data set name in the MASTER DD statement in the DFSMSrmm started procedure. If you specify a name both for DSNAME and MASTER DD, DFSMSrmm ignores the MASTER DD statement.

| DFSMSrmm ignores DSNAME on the client system.

IPLDATE(NO|YES)
Specifies whether IPL date checking is required. Specify *YES* to request IPL date checking, or *NO* to bypass it.

If you specify YES, DFSMSrmm issues a write-to-operator message during startup only if the date of the last run of expiration processing did not occur within two days of the current date. The message prompts the operator to enter the current date and day of the week. Initialization does not proceed until DFSMSrmm receives a valid reply. If you specify *NO* for this operand value, DFSMSrmm does not issue a message.

The IPLDATE operand helps prevent you from running expiration processing with an incorrect system date that can result in expiration of unexpired volumes.

Under normal conditions, the operator message requesting the current date is only issued once per IPL of MVS. If you restart DFSMSrmm, it does not recheck the date.

Additionally, if you run expiration processing daily, a message is not issued. As a result, it is possible under normal conditions to run for a long time and never have the operator prompted to confirm the system date.

Default: IPLDATE(NO)

JOURNALFULL(*nn*)
Specify JOURNALFULL to define a percentage full threshold for the journal data set. When DFSMSrmm detects that the journal has reached this threshold, DFSMSrmm issues message EDG2107E. DFSMSrmm also issues message EDG2107E at DFSMSrmm startup if the journal has already reached the threshold specified. DFSMSrmm issues message EDG2108E as a reminder until the backup completes and the journal is reset. If you specify a backup procedure name on the BACKUPPROC operand, the procedure is started automatically. If you specify a value of 0, DFSMSrmm issues no warnings on that system. You can specify different threshold values for sharing systems. See “Steps for Automating Control Data Set Backup and Journal Clearing” on page 314 for additional information.

Specify a value in the range 0-99.

Default: 75.

| DFSMSrmm ignores JOURNALFULL on the client system.

JRNLNAME(*name*)
Specifies the name of the journal. Specify a name up to 44 characters long.

If you do not specify JRNLNAME in EDGRMMxx, you can specify a name in the JOURNAL DD statement in the DFSMSrmm started procedure. If you do not specify a journal name in either EDGRMMxx or the started procedure, DFSMSrmm does not provide journaling. If you specify a name in both, DFSMSrmm uses the JRNLNAME value and ignores the JOURNAL DD statement.

| DFSMSrmm ignores JRNLNAME on the client system.

LINECOUNT(*nnn*)
Specifies the default number of lines per page for reports, including heading and trailer lines. Specify a value between 10 and 999.

You can override this value when producing individual reports by specifying a parameter to the report program or report utility as described in the *z/OS DFSMSrmm Reporting* document.

Default: LINECOUNT(*54*)

| **LOCALTASKS(***number***)**
| Use this operand to set the number of tasks available on each system for processing locally initiated requests. You can optionally specify a value for local tasks on each and every instance of the DFSMSrmm subsystem; client system, server system, or standard system. On a client system, LOCALTASKS is also the maximum number of tasks that can make a socket connection to the server. Specify a value from 1 to 999.

| Recommendation: Specify or accept the default value of 10 local tasks on all systems. Most of these tasks are rarely used by DFSMSrmm.

| The number of local and server tasks you can use and still successfully start DFSMSrmm is limited by the size of the private region above and below 16MB. To start with more tasks, you will require a larger REGION size.

| Default: LOCALTASKS(*10*)

MASTEROVERWRITE(ADD|LAST|MATCH|USER)
Specify to control how DFSMSrmm allows the overwriting of a volume. You can use one of the following values:

ADD
Specify this value so new data can be created and no existing data can be destroyed. No existing file on a volume can be re-created, but the last file can have new data added to it. When adding data to the last file, DFSMSrmm checks that the data set name used must match the existing data set name. Select this option when you want the last file on the volume to be extended or a new file added to the volume.

| **Note:** DFSMSrmm enforces the MASTEROVERWRITE(ADD) option on a WORM tape that is in master status. This is done to ensure that you see a message from DFSMSrmm rather than one of a number of symptoms as a result of the tape drive preventing overwrites.

LAST
Specify this value to ensure that when an existing file on a master volume is being written to that only the last file on the volume can be used. The data set name used must match the existing data set name. Select this option when you want the last file on the volume to be used for output.

MATCH
Specify this value to ensure that when an existing file on a master volume is being used for output that exactly the same data set name must be used. Select this option when you want any existing file on the volume to be re-created regardless of whether it is the last file on the volume as long as the same data set name is used.

When you use an existing tape file for output all the files which are higher in sequence are destroyed.

USER
Specify this value to allow any existing file on a master

volume to be used for output regardless of the data set names being used and its relative file position on the volume. Select this option when you want validation of master volumes to be just the same as for user status volumes.

When you use an existing tape file for output all the files which are higher in sequence are destroyed.

Default: LAST.

MAXHOLD(*nnn*)
Specifies the maximum number of activities DFSMSrmm performs before the reserve is released and reacquired. For example, if you specify MAXHOLD(100) and you issue the RMM ADDRACK command with COUNT(1000), DFSMSrmm adds 100 racks before the reserve is released and reacquired. Specify MAXHOLD to minimize the impact that long operations, such as RMM TSO ADD subcommands with large COUNT values, or large searches, have to other users.

Even if the DASD where the control data set resides is not shared, consider specifying MAXHOLD because it controls how often users on the same system can gain access to the data set when a long task is running. MAXHOLD also influences the amount of virtual storage that DFSMSrmm uses. Increasing the value might improve the performance of individual DFSMSrmm functions, but other users are locked out from the control data set for a longer time. In a shared environment, the device is also reserved for a longer time, possibly impacting users of other data on the volume.

You do not normally need to alter this value. If your system is storage constrained, you might lower the value from 100 to reduce the amount of virtual storage that DFSMSrmm needs, and therefore its demand on real storage.

In a shared DASD environment in which you are using global resource serialization to convert the DFSMSrmm control data set reserve, you could increase the value to reduce the frequency of the release/reserve, which reduces the traffic on the global resource serialization ring. However, this action causes more virtual storage to be used and increases the time between releases of the control data set. The result is that other users wait longer before gaining access to the control data set.

Specify a value between 10 and 500. Under normal conditions, use the default value.

Default: MAXHOLD(*100*)

MAXRETPD(NOLIMIT|*nnnn*)
Specifies the maximum retention period that a user can request for data sets on volumes. Specify NOLIMIT or a value between 0 and 9999 days. When a value between 0 and 9999 days is specified, the value is added to the current date to determine the maximum allowed expiration date. Specify NOLIMIT to use the dates 99365 or 99366 which mean to never expire. If the calculated date is 31 December 1999, the expiration date 1 January 2000 is used.

MAXRETPD is always used to determine the volume expiration date. The volume expiration date can be ignored when EXPIRYDATEIGNORE is specified on all vital record specifications. MAXRETPD is important if the vital record specification specifies UNTILEXPIRED and the decision is based on the volume expiration date. The preferred method is to put the retention requirements all into the vital record specifications and make MAXRETPD=RETPD. If users are allowed to specify expiration and retention period to override vital record specifications, select a MAXRETPD that covers the maximum retention period they would like to enforce. If this forces 99365 to be reduced, define vital record specifications for any data that should be permanently retained, like DFSMShsm tape data.

Use MAXRETPD to set limits on the values that can be specified for EXPDT and RETPD. If the retention period or expiration date specified in the JCL exceeds the MAXRETPD value, DFSMSrmm overrides it and uses the value in MAXRETPD to determine the expiration date. The volume label has the JCL-specified value because DFSMSrmm does not change tape labels or system control blocks. The control data set contains the JCL-specified value as well as the expiration date calculated from MAXRETPD. You can display the JCL-specified value for information only.

If the DFSMSrmm ISPF dialog or RMM TSO subcommands are used to specify a retention period or expiration date that exceeds the MAXRETPD value, DFSMSrmm fails the subcommand or panel request.

For more information about how to automatically handle expiration date-protected tapes, see “Defining Pools: VLPOOL” on page 162 for information on EXPDTCHECK.

Default: MAXRETPD(NOLIMIT)

MEDIANAME(3480|*medianame*)

Specify to set a default medianame value that DFSMSrmm uses when you do not specify a media name for a volume. The media name is used when you add volumes, define pools in your installation, define a default pool for your installation, and when the EDGINERS utility selects volumes for automatic processing.

Specify a one to eight character name. Here are examples of MEDIANAME that you might define: CART, ROUND, SQUARE, 3420, 3480, TAPE, OPTICAL, and CASSETTE. You can use any name for a media name because DFSMSrmm does not check that the media name you define is a device type that has been defined to MVS. Use MEDIANAME to identify different types of physical shelf space for different media or to distinguish different media characteristics such as Cartridge Tape and Enhanced Capacity Cartridge System Tape.

Prior to setting or changing the default media name, check for any VLPOOL commands that are using the media name that you plan to change. If you change the default media name that is used for a VLPOOL command for an existing pool of volumes, you must consider changing the media names for those existing volumes. Refer to “Changing Pool Definitions” on page 68 which describes how to use RMM CHANGEVOLUME *volser* MEDIANAME subcommand to change the volume media name to match the value in the VLPOOL command. Also check your jobs that run EDGINERS to ensure that MEDIANAME, if coded in the execution parameters, is still consistent with the values you are using.

Default: MEDIANAME(3480)

MOVEBY(SET|VOLUME)

Specify the MOVEBY operand to move volumes that are retained by DFSMSrmm vital record specifications as a set of volumes or as individual volumes.

SET Specify this value when you want volumes moved as a set.

DFSMSrmm moves all volumes in the same multi-volume set that are retained by vital record specifications. All the volumes are retained in the same location selected by the vital record specification priority or location priority for the volume.

VOLUME
Specify this value when you want volumes moved as individual volumes. DFSMSrmm moves the volumes without considering the location of the other volumes in the multi-volume set.

Default: VOLUME

MSG(MIXED|UPPER)
Specify to control the case for message text. You can use:

MIXED	Specify MIXED when you want messages to be displayed exactly as they appear in EDGMTAB. This includes mixed case message text.
UPPER	Specify UPPER to ensure that all messages are displayed in upper case only. Any mixed case messages in EDGMTAB are translated to upper case before they are displayed. This is the value used when DFSMSrmm is inactive and utilities need to issue messages. Default: MIXED.

NOTIFY(NO|YES)
Specifies whether DFSMSrmm should automatically notify volume owners when the volumes they own become eligible for release or when software product volumes are added. Specify YES for automatically notify owners, or NO for notification. You must run DFSMSrmm under the JES2 or JES3 subsystem to use the DFSMSrmm NOTIFY function.

Specify YES and DFSMSrmm notifies the owner using the TSO TRANSMIT command to send an electronic message. The owner information in the DFSMSrmm control data set must include a valid user ID and node name for the message to be sent, and the volume record must specify Notify Owner as a release action.

Specify NO for expired volumes with a release action of notify and the librarian performs the notification. The librarian issues the RMM CHANGEVOLUME subcommand with the CONFIRMRELEASE operand to indicate that the notification has been performed.

Default: NOTIFY(NO).

OPMODE(MANUAL|RECORD|WARNING|PROTECT)
Specifies the running mode of DFSMSrmm. The running mode affects how DFSMSrmm enforces the tape mount validation rules described in “How Does DFSMSrmm Validate Tape Mounts?” on page 18. Protect mode is the only running mode that provides complete validation of volumes and rejection of volumes that do not adhere to DFSMSrmm tape mount validation rules. The running mode also affects other actions and defaults used by DFSMSrmm.

MANUAL — Manual mode. When DFSMSrmm is running in manual mode:

– DFSMSrmm does not record volume usage or validate volumes.

- DFSMSrmm does not provide information to OAM during entry or eject activities.
- You can use RMM TSO subcommands, the DFSMSrmm ISPF dialog, and inventory management functions for all types of media. Updates to the TCDB are controlled by the SMSTAPE parmlib option and affect the following functions:
  - Volume status updates if you run inventory management processing.
  - Any TSO command change that affects the information in the TCDB, such as EJECT, status change, and media information.

RECORD — Record-only mode. When DFSMSrmm is running in record mode, DFSMSrmm performs the actions described for manual mode and also:

- DFSMSrmm records information about tape volumes used on the system, including details about volume owners and data set names.
- DFSMSrmm does not validate or reject volumes during OPEN processing.
- You can use RMM TSO subcommands, the DFSMSrmm ISPF dialog, and inventory management functions for all types of media. Updates to the TCDB are controlled by the SMSTAPE parmlib option and affect the following functions.
  - Volume status updates if you run inventory management processing. Any TSO command change that affects the information in the TCDB, such as EJECT, status change, and media information.
  - Providing information to OAM during entry or eject activities.
  - Updates to the DFSMSrmm control data set based on OAM activity.
  - Controlling the disposition of TCDB volume records at EJECT time.
- DFSMSrmm provides no information to OAM during entry or eject activities and does not update the volume status in the TCDB during inventory management processing or command processing.
- DFSMSrmm tracks OAM changes in the DFSMSrmm control data set.

WARNING — Warning mode. When DFSMSrmm is running in warning mode, DFSMSrmm performs the actions described for manual and record-only mode. DFSMSrmm issues warning messages when an action would have failed if DFSMSrmm was running in protect mode.

- DFSMSrmm records information about tape volumes used on the system, including details about volume owners and data set names.
- DFSMSrmm does not validate or reject volumes during OPEN processing.
- You can use RMM TSO subcommands, the DFSMSrmm ISPF dialog, and inventory management functions for all types of media. Updates to the TCDB are controlled by the SMSTAPE parmlib option and affect the following functions:
  - Volume status updates if you run inventory management processing.
  - Any TSO command change that affects the information in the TCDB, such as EJECT, status change, and media information.

- Providing information to OAM during entry or eject activities.
- Updates to the DFSMSrmm control data set based on OAM activity.
- Controlling the disposition of TCDB volume records at EJECT time.

PROTECT — Protect mode. In protect mode, DFSMSrmm is fully operational. In addition to performing the actions described for manual and record-only mode, DFSMSrmm also performs the following actions:

– Validates all magnetic tape requests and rejects magnetic tape volume mounts under certain conditions, discussed in “How Does DFSMSrmm Validate Tape Mounts?” on page 18.

– Sets the UNCATALOG operand default to YES. UNCATALOG(YES) specifies that DFSMSrmm should uncatalog data sets under conditions described in the UNCATALOG operand description. During OAM exit processing, DFSMSrmm validates all changes to the TCDB and fails those that are not allowed. During cartridge entry processing, DFSMSrmm information overrides information provided by OAM. DFSMSrmm always updates the TCDB. Use SMSTAPE(PURGE) to control the purging or keeping of records in the TCDB during eject processing.

Note: DFSMSrmm processes some parmlib options you have set even when DFSMSrmm does not validate or reject volumes. For example, if you set the TPRACF or UNCATALOG operands described in this section, DFSMSrmm honors these operands when running in any mode. For example if you specify UNCATALOG(YES) and OPMODE(RECORD), DFSMSrmm uncatalogs data sets even though DFSMSrmm does not validate or reject volumes. When you specify OPMODE(WARNING) or OPMODE(PROTECT), DFSMSrmm also honors the setting of the VLPOOL EXPDTCHECK options. To obtain the results you desire, you should review the values you select for these additional options.

Table 18 and Table 19 provide information about the options which are affected by the OPMODE value.

Table 18. How OPMODE Honors the Settings of Various Options

Option	Manual	Record Only	Warning	Protect
EXPDTCHECK	N	N	Y	Y
UNCATALOG	Y	Y	Y	Y
TPRACF	Y	Y	Y	Y
SMSTAPE(UPDATE)	Y	Y	Y	N
SMSTAPE(PURGE)	N	Y	Y	Y

Table 19. How OPMODE Value Affects System-Managed Tape Library Support

Main Area of Activity	Manual	Record Only	Warning	Protect
Command Processing	N	N	N	Y
Expiration Processing	N	N	N	Y
Support for CBRUXxxx Exits	Y	N	N	Y
Purging TCDB Records During Eject Processing	Y	N	N	Y

Default: OPMODE(PROTECT).

PDA(ON|NONE|OFF)
Specify to enable or disable the PDA trace facility.

NONE Specify to prevent DFSMSrmm from enabling the PDA facility. If NONE is specified at startup, DFSMSrmm does not obtain storage for the trace buffer. If NONE is specified when DFSMSrmm is refreshed, it is equivalent to specifying PDA(OFF) and PDALOG(OFF).

OFF Specify to disable the trace facility after DFSMSrmm initialization. When the trace facility is disabled, trace data is not accumulated.

ON Specify to enable the trace facility.

Default: ON.

PDABLKCT(*nnn*)
Specifies the number of blocks or buffers that make up the in-storage trace wrap table. Each block or buffer is the size specified by the PDABLKSZ operand. *nnn* is a minimum of 3 and a maximum of 255.

Default: 255.

PDABLKSZ(*blksz*)
Specifies the DASD blocksize for the DFSMSrmm PDA trace facility data sets, EDGPDOX and EDGPDOY. *nn* is the number of kilobytes in each DASD block and is a minimum of 1 and a maximum of 31.

The default is 27 for 3390 DASD devices, 22 for 3380 DASD devices, and 12 for all other DASD devices.

PDALOG(ON|OFF)
Specify to enable or disable output to the PDA trace data sets.

OFF Specify to disable output to the PDA trace data sets.

ON Specify to enable output to the PDA trace data sets.

Default: ON.

PREACS(NO|YES)
Specify this operand to control whether DFSMSrmm-supplied and EDGUX100 installation exit-supplied values are input to SMS Pre-ACS processing.

NO Specify NO to avoid DFSMSrmm Pre–ACS processing using the DFSMSrmm EDGUX100 installation exit.

YES Specify YES to enable DFSMSrmm Pre–ACS processing using the DFSMSrmm EDGUX100 installation exit.

Default: NO.

RETAINBY(SET|VOLUME)
Use the RETAINBY option to specify whether DFSMSrmm retains multi-volume sets as a set or as individual volumes.

SET When you retain by set, if any volume in a set is retained by a vital record specification, all volumes in the set are retained as vital records. DFSMSrmm uses highest retention date of all volumes in the set as the retention date for all volumes retained as vital records in a set. If no volume in a set is retained by a vital record specification, DFSMSrmm performs expiration processing by set.

DFSMSrmm does not expire volumes in a set if at least one volume in a set is still not ready to expire because it has not reached its expiration date and you have not specified that you want the expiration date ignored.

VOLUME
When you retain by volume, DFSMSrmm retains a volume based on vital record specifications and on the volume expiration date. DFSMSrmm does not consider other volumes in the set.

DFSMSrmm sets an indicator 'retained by set' in the volume information when a volume is vital record specification retained or not expired only because it is a member of a set.

The location where volumes are retained is determined by the MOVEBY option.

Default: RETAINBY(VOLUME).

RETPD(*nnnn*)
Specifies the default retention period for all new data sets on volumes. Specify a value between 0 and 9999 days. The specified value is added to the current date to determine the expiration date. Select a default retention for parmlib RETPD that is a small value to ensure that all tape data created outside the service levels is released as soon as possible. The MAXRETPD value you specify in the parmlib limits the calculated expiration date.

DFSMSrmm sets a default retention period as follows:

- If you specify RETPD or EXPDT, the value is used as the volume's new expiration date. If you do not specify RETPD or EXPDT, DFSMSrmm uses the EXPDT or RETPD allocation attribute of a data class, if all the following are true:
  - The Storage Management Subsystem is active
  - The data set is associated with a data class, either explicitly by the DATACLAS keyword on the JCL or implicitly by an automatic class selection routine
  - The data class has an EXPDT or RETPD allocation attribute
- If you do not specify RETPD or EXPDT, DFSMSrmm uses the default retention period set in EDGRMMxx.

Whenever a new data set is written to tape, DFSMSrmm checks whether the volume's expiration date should be updated, based on whether the new data set has a longer expiration date than the volume on which it is written. DFSMSrmm gets the expiration date for a data set from the job file control block (JFCB) at open time. If there is a date in the JFCB, DFSMSrmm compares this date to the current expiration date for the volume. If the date in the JFCB allows the volume to be retained longer, DFSMSrmm uses that date to update the volume's expiration date.

If there is no expiration date in the JFCB, DFSMSrmm uses the EDGRMMxx RETPD value to calculate the new expiration date. If the RETPD value allows the volume to be retained longer, DFSMSrmm uses that date to update the volume's expiration date.

You can set the date in the JFCB in several ways, including:

- RETPD and EXPDT keywords in the JCL

- Data class when the Storage Management Subsystem is active and the volume is system-managed
- Management class when the Storage Management Subsystem is active and the volume is system-managed
- A user program, using the RDJFCB macro and the OPEN TYPE=J after modifying the JFCB
- Installation exits in use on your particular system

Use vital record specifications to define more specific default retention periods for users by using a data set name prefix. For example, specify:

```
RMM ADDVRS DSNAME('RICK.**') DAYS COUNT(30)
```

to keep all data sets for ID RICK for 30 days. For more information about using vital record specifications, see *z/OS DFSMSrmm Guide and Reference*.

Default: RETPD(*5*)

REUSEBIN(CONFIRMMOVE|STARTMOVE)
Use the REUSEBIN operand to control how DFSMSrmm reuses bins when a volume is moving from a bin.

CONFIRMMOVE
When a volume moves out of a bin, DFSMSrmm does not reuse this bin until the volume move has been confirmed.

STARTMOVE
A bin can be reused as soon as a volume starts moving out of a bin. Extended bin support must be enabled before you can use this operand. See "Enabling Extended Bin Support" on page 347 to enable extended bin support.

Default: REUSEBIN(CONFIRMMOVE).

SCRATCHPROC(*proc_name*)
Specifies the name of the procedure DFSMSrmm starts to replenish scratch volumes in an automated tape library. Specify a procedure name one to eight characters long.

You must run DFSMSrmm with a scratch procedure. You can modify or replace the DFSMSrmm-supplied sample, EDGXPROC, to support your location procedures. You can use the scratch procedure to take any action you would like. For example, you can code the procedure to trigger the required inventory management expiration processing job, to run inventory management, or to take no action.

When an automated tape library detects a low-on-scratch condition, OAM issues a write-to-operator message (CBR3660A). DFSMSrmm intercepts the message and starts the SCRATCHPROC procedure. This procedure runs DFSMSrmm expiration processing to replenish the automated tape library's scratch volumes.

Default: SCRATCHPROC(EDGXPROC)

See "Replenishing Scratch Volumes in a System-Managed Library" on page 381 for information about EDGXPROC the DFSMSrmm default procedure.

| **SERVER(PORT(PortNumber) SERVERTASKS(number))**
| Specifies the type of system you want to set up. SERVER is mutually

exclusive with CLIENT. Neither SERVER nor CLIENT must be specified when DFSMSrmm is used as a standard system.

PORT(*PortNumber***)**
Use this operand to specify the port number to be used for IP communication. The PORT operand is required. Specify a value from 1024 to 65535. Port numbers 1 to 1023 are reserved. The port number must be the same for the client system and the server system to establish a network connection.

Default: None.

SERVERTASKS(*number***)**
Use this operand to specify how many DFSMSrmm tasks should be available on the server to handle socket connections from client systems. DFSMSrmm uses this number to determine how many tasks are to be started for processing all client requests on this server. Specify a value from 1 to 999.

Recommendation: Specify or accept the default value of 10 server tasks on each server system. Depending on the number of CLIENT systems, this value should be increased to allow 3 tasks per CLIENT. Most of the local tasks are rarely used by DFSMSrmm. You do not need more server tasks than the sum of local tasks across your CLIENT systems.

The number of local and server tasks you can use and still successfully start DFSMSrmm is limited by the size of the private region above and below 16MB. To start with more tasks, you will require a larger REGION size.

Default: 10.

SMFAUD(*nnn***)**
Specifies the SMF record number to be used for audit records. Specify a number between 128 and 255 that is different from the value for SMFSEC. The value must conform to standard SMF conventions.

If you do not specify a number, DFSMSrmm does not produce audit records.

Default: No audit records

DFSMSrmm ignores SMFAUD on the client system.

SMFSEC(*nnn***)**
Specifies the SMF record number to be used for security records. Specify a number between 128 and 255 that is different from the value for SMFAUD. The value must conform to standard SMF conventions.

If you do not specify a number, DFSMSrmm does not produce security records.

Default: No security records

SMSACS(NO|YES)
Specify this operand to control whether DFSMSrmm calls SMS ACS processing to enable use of storage group and management class values with DFSMSrmm.

NO Specify NO to prevent DFSMSrmm from calling the SMS ACS processing to obtain management class and storage group names. DFSMSrmm system-based scratch pooling, and scratch pooling and VRS management values based on the EDGUX100 installation exit are used.

YES Specify YES to enable DFSMSrmm calls to the SMS ACS processing to obtain management class and storage group names. If values are returned by the SMS ACS routines the values are used instead of the DFSMSrmm and EDGUX100 decisions.

Default: NO.

| **SMSTAPE(UPDATE PURGE)**

Use SMSTAPE to specify how DFSMSrmm updates the TCDB and controls system-managed tape processing.

UPDATE

Use UPDATE to select the system-managed tape functions DFSMSrmm provides. The UPDATE operand has 3 subparameters: EXITS, SCRATCH, and COMMAND. You can specify one or more of the subparameters. When DFSMSrmm is running in PROTECT mode, DFSMSrmm ignores the UPDATE operand and performs processing as if you specified EXITS, SCRATCH, and COMMAND. When DFSMSrmm is running in WARNING or RECORD mode, DFSMSrmm does not update TCDB information unless you request the update. You can specify one or more of the values. When you specify a value, DFSMSrmm performs the updates to the TCDB.

EXITS Specify EXITS when you want DFSMSrmm volume status information to override the OAM volume status during entry processing, and you want to use the DFSMSrmm VNL exit.

SCRATCH

Specify SCRATCH when you want DFSMSrmm to update the volume status in the TCDB during expiration processing when volumes are returned to scratch status.

COMMAND

Specify COMMAND when you want to use the RMM TSO subcommands or the DFSMSrmm API to update the TCDB. This controls change of status, TDSI and owner information, eject processing and manual cartridge entry processing.

| Default: None if you are running in MANUAL, RECORD, or PROTECT
| mode, but DFSMSrmm ignores the update options and forces them to
| UPDATE(EXITS,SCRATCH,COMMAND) when DFSMSrmm is running
| in PROTECT MODE.

PURGE

Use PURGE to control how DFSMSrmm affects the TCDB volume records during EJECT processing. The default is PURGE(ASIS) in all operating modes except MANUAL mode. In manual mode, DFSMSrmm provides no support for eject processing.

ASIS Specify ASIS when you do not want DFSMSrmm to determine the TCDB volume record dispositions at eject time. Specifying ASIS allows the eject requestor or Library defaults to control the TCDB volume record disposition

NO Specify NO to request that DFSMSrmm prevent TCDB records being deleted.

YES Specify YES when you want DFSMSrmm to force the TCDB volume records to be purged at eject time.

Default: ASIS.

SYSID(*system_name*)
Specifies the name of the system on which DFSMSrmm is running. Specify a unique system name one-to-eight characters long for each system.

If you are running multiple MVS systems and sharing the control data set and journal, specify a unique SYSID for each system.

If you have unshared catalogs, you can specify a list of system IDs using the CATSYSID operand. Select the SYSID values from the list of IDs which include current IDs and previously used IDs.

Default: DFSMSrmm uses the system's SMF identification.

TPRACF(NO|PREDEFINED|AUTOMATIC)
Specifies the type of RACF tape support that DFSMSrmm provides. Use this operand when you want DFSMSrmm to maintain the security profiles that protect tape volumes. You can define RACF tape support for pools of volumes within your installation by using the VLPOOL RACF command described in "Defining Pools: VLPOOL" on page 162. RACF tape support you define can be overridden by RACF tape support you define with VLPOOL.

The TPRACF(NO) option is assumed for any volume serial number containing special characters. To protect tape volumes that use special characters in the volume serial number, use RACF generic TAPEVOL profiles which are outside of DFSMSrmm control.

DFSMSrmm honors all other TPRACF options for volume serial numbers that are alphanumeric including national characters. You can also use RACF generic DATASET profiles to protect data created on tape volume. DFSMSrmm honors the TPRACF setting when running in all modes.

If you set TPRACF(PREDEFINED) or TPRACF(AUTOMATIC), DFSMSrmm ensures that all nonscratch tapes are protected by a discrete RACF TAPEVOL profile by checking that a RACF profile exists whenever a data set is written on a tape. If a profile does not exist, DFSMSrmm creates one. Therefore you do not need to use RACF installation exits to set the JCL PROTECT=YES option or specify PROTECT=YES in your JCL. You can use generic data set profiles for all tape data sets without changes to JCL or installation procedures, if you used the VLPOOL command with the RACF(Y) operand, because DFSMSrmm creates a TVTOC when you use the RACF TAPEDSN option.

For both TPRACF(PREDEFINED) and TPRACF(AUTOMATIC), DFSMSrmm ensures TAPEVOL profiles are always deleted during recycling of scratch volumes, if they exist. When you use a volume for output and there is no RACF protection for the volume when you close the data set, DFSMSrmm creates a profile to protect the volume. The owner of the volume is given RACF access to the profile. If TAPEVOL and TAPEDSN are active, DFSMSrmm creates a TVTOC and adds the first file data set entry.

For more information on RACF processing and security options, see Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171.

- When you specify TPRACF(NO), DFSMSrmm does not create RACF tape profiles for any volumes in your installation.

  If you also specify one of these VLPOOL values:

  - VLPOOL RACF(N)

    DFSMSrmm does not create tape profiles or TVTOCS for volumes in the pool that you are defining with the VLPOOL command.
  - VLPOOL RACF(Y)

    DFSMSrmm does not create tape profiles or TVTOCS for volumes in the pool that you are defining with the VLPOOL command.
- When you specify TPRACF(PREDEFINED) or TPRACF(P), DFSMSrmm creates predefined RACF tape profiles for any volumes in your installation.

  If you also specify one of these VLPOOL values:

  - VLPOOL RACF(N)

    DFSMSrmm does not create tape profiles or TVTOCS for volumes in the pool that you are defining with the VLPOOL command.
  - VLPOOL RACF(Y)

    DFSMSrmm creates RACF tape profiles for volumes in the pool that you are defining with the VLPOOL command. The RACF tape profiles are created using RACF options that you have specified and the RACF tape profiles that DFSMSrmm creates when RMM ADDVOLUME, CHANGEVOLUME, or DELETEVOLUME subcommands are issued.

    DFSMSrmm deletes TAPEVOL profiles during recycling of scratch tapes if the profiles exist.

    DFSMSrmm creates a profile when a volume that is used for output does not have a RACF tape profile defined.

    The owner of the volume is given RACF access to the profile.

    If TAPEVOL and TAPEDSN are active, DFSMSrmm creates a TVTOC and adds the first file data set entry.
- When you specify TPRACF(AUTOMATIC) or TPRACF(A), DFSMSrmm creates tape profiles for any volume in your installation.

  If you also specify one of these VLPOOL values:

  - VLPOOL RACF(N)

    DFSMSrmm does not create tape profiles or TVTOCS for volumes in the pool that you are defining with the VLPOOL command.
  - VLPOOL RACF(Y)

    DFSMSrmm creates RACF tape profiles for volumes in the pool. Table 29 on page 186 shows what DFSMSrmm processing takes place when you specify RACF(Y) for a pool.

  The processing is the same as TPRACF(PREDEFINED) with the following exceptions:

  - Scratch tape volumes do not have TAPEVOL profiles created.
  - IF TAPEVOL and TAPEDSN are active, RACF automatically creates TAPEVOL and data set profiles for the ADSP and PROTECT=YES users.

– TPRACF(AUTOMATIC) also supports the environment where an installation uses RACF installation exits to create TAPEVOL profiles.

Default: TPRACF(*NO*)

TVEXTPURGE(RELEASE|EXPIRE|NONE)

Specifies how DFSMSrmm processes volumes to be purged by callers of EDGTVEXT or EDGDFHSM.

- NONE — DFSMSrmm take no action for volumes to be purged.
- RELEASE — DFSMSrmm releases volume to be purged according to the release actions set for the volume. You must run expiration processing to return a volume to scratch status.
- EXPIRE — Use the EXPIRE option to set the volume expiration date to the current date for volumes to be purged. Use this operand in combination with vital record specifications that use the UNTILEXPIRED retention type. You can then optionally extend retention using the extra days retention type.

Default: RELEASE

UNCATALOG(YES|NO|SCRATCH)

Specifies the type of catalog support to provide:

N — Do not use DFSMSrmm catalog processing. DFSMSrmm does not uncatalog data sets under any circumstances.
Y — always uncatalog data sets. Use DFSMSrmm catalog processing. DFSMSrmm uncatalogs data sets when:

– A volume is returned to scratch status, DFSMSrmm uncatalogs all the data sets on the volume.

– The RMM DELETEVOLUME FORCE subcommand is issued for a volume, DFSMSrmm uncatalogs all the data sets on the volume.

– The RMM CHANGEVOLUME DSNAME subcommand is issued for a volume, DFSMSrmm uncatalogs all the data sets on the volume. If the data set name specified on the RMM CHANGEVOLUME subcommand command matches the data set name on the volume, then DFSMSrmm only uncatalogs subsequent data sets.

– The RMM DELETEDATASET subcommand is issued for a data set, DFSMSrmm uncatalogs the data set. Also, DFSMSrmm uncatalogs all data sets recorded on the same volume with higher data set sequence numbers.

– A tape data set is overwritten, DFSMSrmm uncatalogs the data set. Also, all data sets recorded on the same volume with higher data set sequence numbers are uncataloged.

S — Only uncatalog data sets when the volume on which they reside is returned to scratch status.

Default: The default is UNCATALOG(N) unless DFSMSrmm is running in protect mode. DFSMSrmm uses the UNCATALOG default of Y when DFSMSrmm is running in protect mode.

If you set UNCATALOG(S) or UNCATALOG(Y), DFSMSrmm uncatalogs data sets even when DFSMSrmm is running in manual mode, record mode, or warning mode.

You should use the UNCATALOG(N) option during early implementation of DFSMSrmm on your system, when DFSMSrmm is running at the same time as your existing management software.

If you use the DFSMSrmm EDGUX100 exit to request suppression of data set recording for a volume, you should ensure that any data sets that are cataloged, but not recorded by DFSMSrmm are uncataloged by some other mechanism. To leave nonexistent data sets cataloged could lead to later processing problems.

If you use the UNCATALOG(N) option to prevent DFSMSrmm from uncataloging tape data sets, you should ensure that data sets are uncataloged by some other mechanism. To leave nonexistent data sets cataloged could lead to later processing problems.

Ensure Integrated catalog facility catalogs are shared if catalog control is required or if you specify the UNCATALOG(Y) or (S) operand.

VRSCHANGE(INFO|VERIFY)
Use VRSCHANGE to specify the action that DFSMSrmm should take during inventory management if you have made any changes to vital record specifications using RMM ADDVRS or RMM DELETEVRS subcommands.

If You Specify	Then
INFO	No additional processing or actions are required when vital record specification changes occur.
VERIFY	Any changes made to vital record specifications must be verified by running EDGHSKP vital record processing using the VERIFY parameter. DFSMSrmm must issue return code zero before EDGHSKP vital record processing can be performed to update the control data set.

Default: VERIFY.

VRSJOBNAME(1|2)
Use VRSJOBNAME to select how DFSMSrmm uses a job name in a vital record specification to match a data set to a movement and retention policy. DFSMSrmm records the data set name and job name for new tape files you create. You can define the name of the job that created a data set by using the RMM ADDDATASET subcommand or the CHANGEDATASET subcommand for existing data sets. You can specify a job name and data set name in a vital record specification so that data sets that match the data set name and job name are moved and retained by the policy defined in the vital record specification.

If You Specify	DFSMSrmm Uses	And if There Is No Match
VRSJOBNAME(1)	The data set and job name to match a data set to a vital record specification. Job name is the primary value used to match the data set to a vital record specification.	A match by data set name only is acceptable.
VRSJOBNAME(2)	The data set name to match a data set to a vital record specification. If a data set matches multiple vital record specifications with the same data set name, then DFSMSrmm uses a job name to further qualify the data set name.	DFSMSrmm does not apply a policies to the data set.

Default: VRSJOBNAME(2).

VRSMIN(*count***,***action***)**
Use VRSMIN to specify a minimum number of vital record specifications

required by inventory management vital record processing, and the action to be taken by DFSMSrmm when the minimum number of vital record specifications are not defined. DFSMSrmm counts the vital record specifications used by vital record processing. DFSMSrmm does not count vital record specifications that are deleted during vital record processing.

count can be 0 to 2,147,483,647. Specify a count value of 0 to disable this checking.

The default count value is 1. Specify *action* to control the action DFSMSrmm takes when the *count* is not reached. *Action* can be FAIL, INFO, or WARN.

If You Specify	DFSMSrmm
FAIL	Issues message EDG2229I to the MESSAGE file and stops inventory management processing. Processing ends with return code 8.
INFO	Issues message EDG2229I to the MESSAGE file and processing continues.
WARN	Issues message EDG2229I to the MESSAGE file and sets a minimum return code of 4. Processing continues.

Default: FAIL.

VRSEL(OLD|NEW)

Use VRSEL to control how DFSMSrmm inventory management vital record processing uses retention and movement information that are defined in vital record specifications.

OLD Specify OLD if you want DFSMSrmm to:
- Ignore retention information that is defined in NAME vital record specifications
- Ignore release options that are defined in vital record specifications
- Treat the ANDVRS in the RMM ADDVRS command as if it is a NEXTVRS

NEW Specify NEW if you want DFSMSrmm to process retention information in NAME vital record specifications, release options defined in vital record specifications, and vital record specifications chained using the RMM ADDVRS ANDVRS operand.

Default: OLD.

Defining Tapes Not Available on Systems: REJECT

Use the REJECT command to prevent a range of tapes from being used on a particular system. For example, you can reject production tapes on a development system.

This command is useful under the following conditions:
- Using a shared RACF data set and the security profiles for a volume allow the volume to be used on all systems.
- To partition tape libraries as described in “Partitioning System-Managed Tape Libraries” on page 109.
- Preventing a stacked volume from being used outside a VTS.

The REJECT command helps you limit tape usage at the system level. Figure 62 shows an example of the REJECT command and the operands you can code in the parmlib member EDGRMMxx.

```
REJECT OUTPUT(CC12*)
REJECT ANYUSE(VM1*),OUTPUT(VM*)
REJECT ANYUSE(DD0*)
REJECT ANYUSE(*)
```

Figure 62. Parmlib Member EDGRMMxx REJECT Command Examples

REJECT Command Syntax

Figure 63 shows the syntax of the REJECT command:

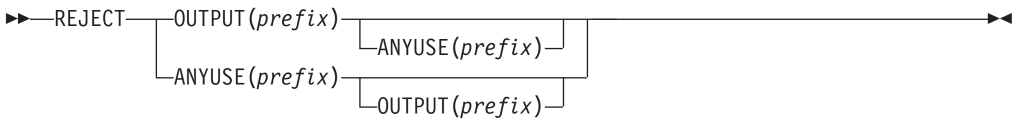


Figure 63. Parmlib Member EDGRMMxx REJECT Command Syntax

You can specify multiple REJECT commands to define different ranges of tapes. However, use only one ANYUSE operand and one OUTPUT operand for each REJECT command, as shown in Figure 62.

DFSMSrmm allows all volume mounts. DFSMSrmm checks volume validity only when a data set on the volume is opened. When a data set on a volume is opened, DFSMSrmm checks the volume's shelf location against the prefixes specified in the REJECT command. If a volume has no library shelf location, DFSMSrmm checks the volume serial number against the prefixes specified in the REJECT command. DFSMSrmm recognizes the most specific prefix match. If you specify both ANYUSE and OUTPUT for the same prefix, ANYUSE overrides OUTPUT. Normally, REJECT prefixes match those that are used on VLPOOL commands, but it is not mandatory.

At OPEN time, DFSMSrmm can reject a range of tapes if you define the range of volumes to DFSMSrmm. DFSMSrmm cannot reject a range of tapes that are not defined to it.

You can use the REJECT ANYUSE command with an automated tape library to prevent volumes from being automatically defined in the TCDB when they are entered in an automated tape library. If the volume matches the REJECT ANYUSE command, the volume is not defined to the TCDB. Another system sharing the automated tape library can take the volume to define it in its TCDB only. You can also prevent a volume from being defined in the TCDB by specifying other than the USE(MVS) attribute when defining the volume to DFSMSrmm.

If you use REJECT OUTPUT commands, the volume is defined in the TCDB and can be entered into an automated tape library.

Use caution when using the REJECT command with an automated tape library as it is possible to continue needless rejecting of volumes.

REJECT Command Operands

ANYUSE(*prefix*)
Specifies the shelf locations of volumes not available for read or write

processing. *prefix* is a generic shelf location that consists of one-to-five alphanumeric, national, or special characters that are followed by an asterisk (*). Specify a single asterisk if, at OPEN time, you want to reject all volumes not defined to DFSMSrmm. For example, ANYUSE(*) prevents the use of foreign tapes, unless you use the DFSMSrmm installation exit EDGUX100 to request that DFSMSrmm ignore such tapes. You can also use the REJECT ANYUSE command to partition system-managed tape libraries. See Chapter 4, Partitioning System-Managed Tape Libraries for more information.

Default: None.

OUTPUT(*prefix***)**
Specifies the shelf locations of volumes not available for write processing. *prefix* is a generic shelf location that consists of one-to-five alphanumeric, national, or special characters that are followed by an asterisk (*). Specify a single asterisk if, at OPEN time, you want to prohibit writing to all volumes that are not defined to DFSMSrmm.

Default: None.

Defining Security Classes: SECCLS

Use the SECCLS command to define security classes for data sets and volumes. These security classes appear in reports and in output for the RMM TSO subcommands. DFSMSrmm records these security classes in the DFSMSrmm control data set only; it does not make RACF aware of them. There is no connection between these definitions and any similar definitions or function provided in RACF, but you can use similar values for overall consistency.

DFSMSrmm determines the security classification of a tape volume with multiple data sets by the highest classification found for a single data set.

Data sets that do not match any of the masks specified in SECCLS definitions are assigned no security classification. DFSMSrmm uses a number 0 to indicate no security classification. Whenever you create a tape data set, DFSMSrmm uses the SECCLS masks to classify data sets and volumes. Figure 64 on page 160 shows an example of the SECCLS command and the operands you can code in the parmlib member EDGRMMxx.

If you remove a security class that is assigned to a volume, DFSMSrmm issues an error message when a data set on the volume is opened. DFSMSrmm also treats the volume as having the lowest defined security level. The default security class is defined by a mask of '**'.

```
SECCLS  NUMBER(01)                                          -
        NAME(UNCLASS)                                       -
        DESCRIPTION('UNCLASSIFIED')                         -
        MASK('**')                                          -
        SMF(N)                                              -
        MESSAGE(N)                                          -
        ERASE(N)
SECCLS  NUMBER(02)                                          -
        NAME(U)                                             -
        DESCRIPTION('UNCLASS')                              -
        MASK('STSGAM.**')                                   -
        SMF(N)                                              -
        MESSAGE(N)                                          -
        ERASE(N)
SECCLS  NUMBER(10)                                          -
        NAME(IUO)                                           -
        DESCRIPTION('INTERNAL USE ONLY')                    -
        SMF(N)                                              -
        MESSAGE(N)                                          -
        ERASE(N)                                            -
        MASK('SYS1.IUO.**')
SECCLS  NUMBER(30)                                          -
        NAME(CC)                                            -
        DESCRIPTION('CONFIDENTIAL')                         -
        MASK('PAYROLL.**')                                  -
        SMF(Y)                                              -
        MESSAGE(N)                                          -
        ERASE(Y)
SECCLS  NUMBER(100)                                         -
        NAME(IC)                                            -
        DESCRIPTION('CONFIDENTIAL')                         -
        MASK(+
       '**.IC.**',+
        '**.VERTR.**',+
        '**.CONFI.**'+
             )                                              -
        SMF(Y)                                              -
        MESSAGE(N)                                          -
        ERASE(Y)
```

Figure 64. Parmlib Member EDGRMMxx SECCLS Command Examples

SECCLS Command Syntax

Figure 65 shows the syntax of the SECCLS command:

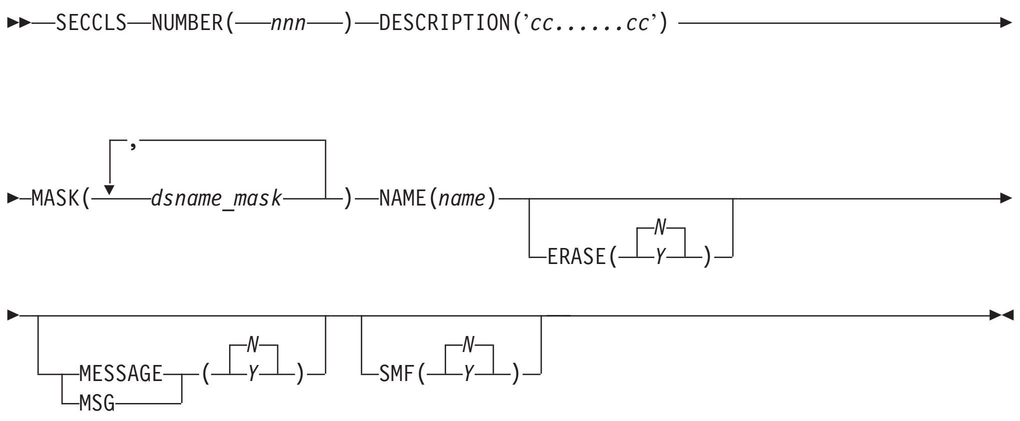


Figure 65. Parmlib Member EDGRMMxx SECCLS Command Syntax

To make the default security class the highest level, specify the following operands with the SECCLS command:

```
SECCLS NUMBER(255) MASK('**')
```

If you do not specify any SECCLS statements, DFSMSrmm defaults to a classification of blanks.

SECCLS Command Operands

DESCRIPTION(*'cc......cc'***)**
Describes the security class. DFSMSrmm uses the description you provide as the security class name. Specify a description between 1 and 32 characters.

Default: None. You must specify this operand.

ERASE(N|Y)
Specifies whether volumes must be erased on release. *Y* indicates volumes must be erased on released, and *N* indicates that they do not have to be.

You can use the DFSMSrmm utility EDGINERS to automatically erase and reinitialize tapes, or you can use your own method to erase them.

Default: ERASE(N)

MASK(,*'dsname_mask'***)**
Specifies a data set name mask used for assigning a security classification. You can specify multiple data set name masks separated by a comma. In addition to standard data set naming conventions, you can use the following masking characters to create the data set name mask:

***** A single * represents a single qualifier of any number of characters.

a single * when used within a qualifier represents zero or more characters.

more than one single * can be used within a qualifier as long as a character precedes or follows the *.

.** represents zero or more qualifiers. At the end of the mask, it indicates to ignore any remaining characters.

** indicates to select all data sets.

% (percent_sign)
A place holder for a single character.

For example, you can specify MASK('USERID.**.CONF.**').

Default: You are required to specify this operand.

MESSAGE|MSG(N|Y)
Specifies whether to issue operator confirmation messages when a data set on a volume in this security class is opened. Specify *Y* to issue operator confirmation messages for this class, or *N* to suppress the messages. The message number is EDG4008A.

You can use this opportunity to perform any local operation procedures, such as adding a security sticker to a volume.

Default: MSG(*N*)

NAME(*name***)**
Specifies a name for the security class. The name is used for reports and communication with users. Specify a value between 1 and 8 characters.

Default: None. You are required to specify this operand.

NUMBER(*nnn***)**
Specifies a security classification number between 1 and 255.

The security classification numbers indicate relative levels of importance. Higher numbers mean higher levels of security. Your installation can assign more specific meanings to the numbers.

Default: None. You are required to specify this operand.

SMF(N|Y)
Specifies whether to write an SMF security record each time a tape data set in this class is opened. *Y* means to write an SMF record, and *N* means not to write it. See *z/OS DFSMSrmm Reporting* for information on using EDGAUD to produce a report for these SMF records.

Default: SMF(*N*)

Defining Pools: VLPOOL

| Use the VLPOOL command to define pools and to set release actions for all
| volumes in the pool. See Chapter 4, “Organizing the Removable Media Library,” on
| page 63 for information about pooling. When you add a new volume to the library,
| DFSMSrmm assigns it a shelf location from the specified pool. Figure 66 on page
| 163 shows an example of the VLPOOL command and the operands you can code
| in the parmlib member EDGRMM*xx*.

```
VLPOOL PREFIX(Y2*) TYPE(R) DESCRIPTION('CUSTOMER REEL TAPES') -
        MEDIANAME(3420) RACF(Y) EXPDTCHECK(Y)
VLPOOL PREFIX(Y3*) TYPE(R) DESCRIPTION('CUSTOMER 3480 CARTRIDGES') -
        MEDIANAME(3480) RACF(Y) EXPDTCHECK(Y)
VLPOOL PREFIX(P0*) TYPE(R) DESCRIPTION('SOFTWARE PRODUCT REELS') -
        MEDIANAME(3420) RACF(Y) EXPDTCHECK(Y)
VLPOOL PREFIX(P2*) TYPE(R) DESCRIPTION('SOFTWARE CARTRIDGES') -
        MEDIANAME(3480) RACF(Y) EXPDTCHECK(Y)
VLPOOL PREFIX(C*) TYPE(R) DESCRIPTION('MVS AND VM BACKUP TAPES') -
        MEDIANAME(3480) RACF(Y) EXPDTCHECK(Y)
VLPOOL PREFIX(X8*) TYPE(R) DESCRIPTION('IN TAPES') -
        MEDIANAME(3480) RACF(Y) EXPDTCHECK(Y)
VLPOOL PREFIX(M*) TYPE(S) DESCRIPTION('SCRATCH POOL') -
        MEDIANAME(3480) RACF(Y) EXPDTCHECK(Y) SYSID(DG4)
VLPOOL PREFIX(*) TYPE(R) DESCRIPTION('EVERYTHING ELSE') -
        MEDIANAME(3480) RACF(Y) EXPDTCHECK(Y)
VLPOOL PREFIX(*) NAME(DEFAULT) TYPE(S) SYSID(SYS1) -
        DESCRIPTION('Default pool')
VLPOOL PREFIX(S*) NAME(SPECIAL) TYPE(R) -
        DESCRIPTION('EDGUX100 set pool')
```

Figure 66. Parmlib Member EDGRMMxx VLPOOL Command Examples

VLPOOL Command Syntax

Figure 67 shows the syntax of the VLPOOL command:

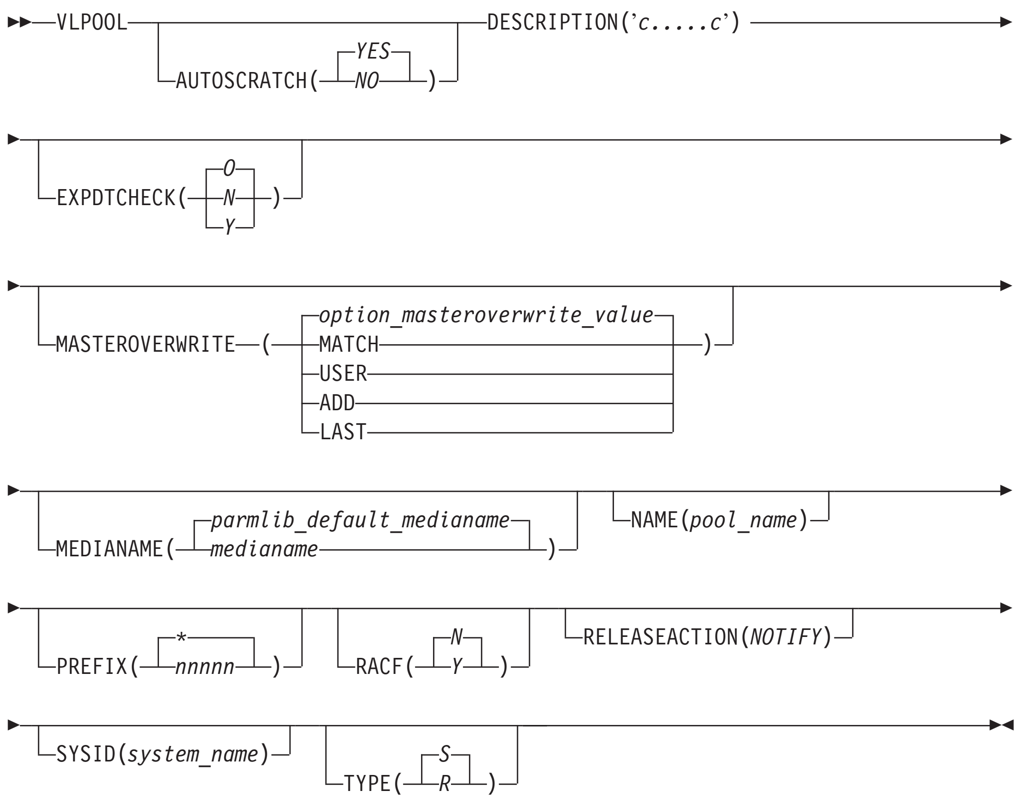


Figure 67. Parmlib Member EDGRMMxx VLPOOL Command Syntax

Each VLPOOL command defines a pool. Each generic shelf location defined by PREFIX must be unique. If you do not specify any VLPOOL commands, or do not specify a VLPOOL command for POOL(*), DFSMSrmm automatically creates a default pool definition that includes all the shelf locations and volumes in them. The defaults for this pool are:

```
VLPOOL RACF(N) TYPE(S) EXPDTCHECK(O) MEDIANAME(parmlib_default_medianame) -
  DESCRIPTION('DEFAULT POOL') PREFIX(*)
```

VLPOOL Command Operands

AUTOSCRATCH(YES|NO)

Use this operand when you want DFSMSrmm to automatically perform release actions for all the volumes in a pool. The default is AUTOSCRATCH(YES). If you need to take any special actions outside of DFSMSrmm, perhaps on another system, specify AUTOSCRATCH(NO).

Use this operand to override the scratch processing release action for volumes in this pool. By default, AUTOSCRATCH(YES) return to scratch is performed automatically when expiration processing is running on a system that has access to the catalogs, TCDB, and library for the volume. If you need to take any special actions not performed by DFSMSrmm, perhaps on another system, specify AUTOSCRATCH(NO).

When you specify AUTOSCRATCH(YES) and do not manually confirm the scratch release action, DFSMSrmm performs return to scratch processing, including UNCATALOG parmlib option, TPRACF parmlib option, and SMSTAPE(UPDATE(SCRATCH)).

When you specify AUTOSCRATCH(NO) or manually confirm the scratch release action, DFSMSrmm does not return the volume to scratch status until you have manually confirmed the volume release action and run expiration processing.

DFSMSrmm checks the AUTOSCRATCH setting and the scratch release action for the volume when the volume is returning to scratch. If the scratch release action is set and the volume is in a pool with AUTOSCRATCH(NO), DFSMSrmm leaves the volume in pending release status. You must perform whatever actions or cleanup activity you require and confirm that the scratch release action has been performed using the subcommand: RMM CV volser CRLSE(SCRATCH). Run expiration processing on any DFSMSrmm system to return the volume to scratch.

Default: AUTOSCRATCH(*YES*)

DESCRIPTION(*'c.....c'*)

Describes the pool. Specify a description between 1 and 32 characters. You must enclose the value in quotes if you use blanks or special characters.

Default: None. You are required to specify this operand.

EXPDTCHECK(Y|N|O)

Specifies to automate the processing of unexpired tape data sets at the pool level.

N indicates that DFSMSrmm is not to check or validate the expiration date of data sets on a tape. It allows the tape to be overwritten after the system has checked the validity of the tape. DFSMSrmm does not overwrite or replace the expiration date on the tape. This situation happens when a volume is released early, or when someone changes its expiration date or retention period by using the RMM TSO subcommands.

Recommendation: Specify N when running a DFSMSrmm-managed scratch tape pool to ensure that tapes are reused only when the information on them is no longer required. DFSMSrmm can easily automate reuse of tapes if they are still expiration protected, for example, when they are reused as scratch tapes.

Y indicates that DFSMSrmm ensures that all unexpired tape data sets are never overwritten. It fails specific mount requests and requests a remount if the tape request is nonspecific. If an expiration date or retention period was coded in the JCL when the date was originally written to the volume, the tape label is expiration date protected, and DFSMSrmm records this as the original expiration date for the volume. Setting EXPDTCHECK(*Y*) ensures that the original expiration dates specified in the JCL are honored, even if the tape has been released early, or its expiration date or retention period has been changed using the RMM TSO subcommands.

When you specify Y and a tape is expiration protected, reinitialize it before using it again. You can determine what volumes you need to reinitialize by looking at the original expiration date recorded in the DFSMSrmm control data set. To find the date, see the extract data set or use the RMM LISTVOLUME subcommand.

O indicates that DFSMSrmm takes no action but allows the operator or automated software such as NetView to reply as necessary to any write-to-operator messages (IEC507D).

DFSMSrmm ignores the EXPDTCHECK operand and requires an operator reply to the write-to-operator with reply messages when:

- DFSMSrmm is running in record-only mode. DFSMSrmm does not have enough information to base a decision on. Your installation's procedures help the operator to decide how to reply to messages.
- DFSMSrmm is running in warning mode and the volume is rejected for any reason. DFSMSrmm decides that the volume should not be used, but allows its use because in warning mode it does not reject volumes. Your operator or current operating procedures determine whether to reuse the volume and override the expiration protection.

Note: When you use the EDGUX100 exit to tell DFSMSrmm to ignore a volume, DFSMSrmm does not reply to the IEC507D message because the volume is not DFSMSrmm-managed.

Default: EXPDTCHECK(*O*)

MASTEROVERWRITE(ADD|LAST|MATCH|USER))

Specify the VLPOOL MASTEROVERWRITE operand to control how DFSMSrmm allows the overwriting of a volume. You can specify if you want to allow data to be appended to the end of a volume or overwritten, or to allow new files to be added to a volume. The MASTEROVERWRITE value applies to all volumes that reside in a pool. If you do not specify a MASTEROVERWRITE value, DFSMSrmm uses the MASTEROVERWRITE value that you set using the EDGRMM*xx* parmlib member OPTION MASTEROVERWRITE operand, as described in "Defining System Options: OPTION" on page 134.

ADD
Specify this value so new data can be created and no existing data can be destroyed. No existing file on a volume can be recreated, but the last file can have new data added to it. When adding data to the last file, DFSMSrmm checks that the data set name used must match the existing data set name. Select this option when you want the last file on the volume to be extended or a new file added to the volume.

Note: DFSMSrmm enforces the MASTEROVERWRITE(ADD) option on a WORM tape that is in master status. This is done to ensure that you see a message from DFSMSrmm rather than one of a number of symptoms as a result of the tape drive preventing overwrites.

LAST
Specify this value to ensure that when an existing file on a master volume is being written to that only the last file on the volume can be used. The data set name used must match the existing data set name. Select this option when you want the last file on the volume to be used for output.

MATCH
Specify this value to ensure that when an existing file on a master volume is being used for output that exactly the same data set name must be used. Select this option when you want any existing file on the volume to be recreated regardless of whether it is the last file on the volume as long as the same data set name is used.

When you use an existing tape file for output all the files which are higher in sequence are destroyed.

USER
Specify this value to allow any existing file on a master volume to be used for output regardless of the data set names being used and its relative file position on the volume. Select this option when you want validation of master volumes to be just the same as for user status volumes.

When you use an existing tape file for output all the files which are higher in sequence are destroyed.

Default: EDGRMM*xx* parmlib member OPTION MASTEROVERWRITE operand.

MEDIANAME(*media_name***)**
Specifies a media name for all volumes in this pool. Specify a one-to-eight character name. If you do not specify a MEDIANAME, DFSMSrmm uses the medianame that you set using the EDGRMM*xx* parmlib member OPTION MEDIANAME operand, as described in “Defining System Options: OPTION” on page 134.

You can specify any name because DFSMSrmm does not check whether the device type has been defined to MVS. Some examples of MEDIANAME that you might define include: CART, ROUND, SQUARE, 3420, 3480, TAPE, OPTICAL, CASSETTE. DFSMSrmm accepts media names that have not been generated on the MVS system that is running DFSMSrmm. Use of MEDIANAMEs that describe size or shape can give you more flexibility in the media that can reside in a pool. Use MEDIANAME to identify different types of physical shelf space for different media, or to distinguish different media characteristics such as Cartridge System Tape and Enhanced Capacity Cartridge System Tape. See “Defining Storage Locations: LOCDEF” on page 127 for information about using media names when defining storage locations.

If you change a media name for a VLPOOL command for an existing pool of volumes, or add a new VLPOOL command that has a different media name than existing volumes which are covered by that VLPOOL command, you must consider changing the media names for those existing volumes. Refer to “Changing Pool Definitions” on page 68 which describes how to use RMM CHANGEVOLUME *volser* MEDIANAME to make the volume media name match the value in the VLPOOL command.

Default: MEDIANAME(*parmlib_default_medianame*)

NAME(*pool_name***)**
Specifies a pool name that DFSMSrmm uses to update operator messages and tape drive displays for nonspecific tape mount requests. *pool_name* can be a 1 to 8 character name. The first character must be an alphanumeric or national character. Blank, comma, or semicolon cannot be used. Specify a BTLS category name to use DFSMSrmm pooling support for volumes managed by BTLS. *pool_name* can be a tape storage group name defined in your active SMS configuration. NAME is optional and is used on tape drive displays. To use the pool name in messages, you must specify RACK(999) on the MNTMSG command so that the message is updated at the end of the message text.

If you specify a NAME value that is a valid storage group name, DFSMSrmm uses the NAME value as the default storage group name for all the volumes added into this pool range. By naming a specific storage group with the RMM ADDVOLUME subcommand, you can select a pool for a volume at the individual volume level.

You can use the same NAME value on multiple VLPOOL commands enabling you to group multiple prefix ranges into a single logical pool without the need to use the RMM ADDVOLUME subcommand to specify storage group names. You can use RMM ADDVOLUME or RMM CHANGEVOLUME subcommand with the STORAGEGROUP operand to add or remove individual volumes in a logical pool.

Default: None.

PREFIX(*nnnnn****)**
Specifies a generic shelf location that is used in operator messages, RMM TSO subcommands, and the DFSMSrmm ISPF dialog. A pool prefix is one to five alphanumeric, national, or special characters followed by an asterisk. Use a single asterisk to specify the default volume pool that contains all volumes not specifically included in another pool.

If a volume's shelf location falls into a number of possible pools, DFSMSrmm chooses the most specific pool. For example, you defined AB* and ABC* as pools. If volume ABC123 is from shelf location ABC123, it belongs in pool ABC*, not AB*. DFSMSrmm prevents duplicate pool prefixes.

DFSMSrmm uses the prefix value to assign a newly defined volume to a scratch pool. For all volumes, DFSMSrmm uses the:

- Storage group value specified with the RMM ADDVOLUME subcommand issued when the volume is defined to DFSMSrmm.
- PARMLIB VLPOOL NAME operand value to assign the storage group when NAME specifies a valid tape storage group.
- Pool prefix to assign a pool when no NAME operand value or storage group value is specified.

Default: PREFIX(***)

RACF(Y|N)
Specifies the type of RACF tape support that DFSMSrmm should provide for volumes in the pool you are defining. When you are defining RACF tape support for volumes in a pool, you must look at the RACF tape support you have defined for your installation with the OPTION TPRACF command described in "Defining System Options: OPTION" on page 134. Specify *Y* if you want DFSMSrmm to create RACF tape profiles for the volumes in the pool.

Ensure that OPTION TPRACF(*AUTOMATIC or PREDEFINED*) is specified. If OPTION TPRACF(*NO*) is specified, DFSMSrmm will not create the RACF tape profiles.

Only specify *Y* for tape pools because TAPEVOL profiles are not required for optical volumes.

Specify *N* if you do not want DFSMSrmm to create RACF tape profiles for the volumes in the pool. If you specify *N*, DFSMSrmm plays no part in creating or deleting RACF tape profiles, regardless of the system-wide option TPRACF.

Default: RACF(*N*)

RELEASEACTION(*NOTIFY***)**
Use this operand with the NOTIFY value to automatically set the NOTIFY release action for all volumes in this pool. If you have an e-mail address for the owner of the volume, DFSMSrmm sends the owner notification that the volume is pending release. By default, DFSMSrmm does not set the NOTIFY release action. The VLPOOL NOTIFY value is checked at the time the volume is set pending release. If NOTIFY is set for the volume, DFSMSrmm sets the NOTIFY release action.

Recommendation: You could use this option to build in a delay or a checkpoint in scratch processing to ensure that volumes are ready to return to scratch. If parmlib OPTION NOTIFY(NO) is set , or you do not have an e-mail address for the owner of the volume, you must notify the user and later confirm the notify action to DFSMSrmm.

Default: None.

SYSID(*system_name***)**
Associates the scratch pool you are defining with a particular system. Specify a value one to eight characters long. DFSMSrmm matches the value with the SYSID operand of the OPTION command.

DFSMSrmm enforces a match on SYSID when validating nonspecific tape mounts. Only scratch volumes from a pool associated with a specific system can satisfy nonspecific mount requests for that system. DFSMSrmm rejects volumes from other pools on that system.

When you specify a SYSID, and you have also activated MNTMSG, all nonspecific mount messages are updated to include the pool prefix to identify the pool to be used to the operator.

You can use the EDGUX100 installation exit to assign multiple scratch pools to be used for nonspecific tape volume requests for each system.

If there is no scratch pool defined for the current system, and the EDGUX100 exit does not select a specific scratch pool, DFSMSrmm does not update the mount message. DFSMSrmm selects the first pool in the collating sequence with the correct SYSID to update mount messages. However, the operator can mount a volume from any eligible pool. DFSMSrmm ignores the SYSID value when you use ACS routines to select a storage group pool for a nonspecific tape request.

Default: The pool is not associated with any particular system.

TYPE(S|R)
Specifies the type of pool. In DFSMSrmm, there are two categories of pools: rack and scratch. Specify *R* for a rack pool, and *S* for a scratch pool.

A *rack pool* is shelf space that can be assigned to hold any volumes that are generally read-only and that enter and leave your installation on an ad hoc basis. These volumes are typically software product volumes and customer volumes and do not adhere to your installation's naming conventions. Although you can add scratch volumes to rack pools, you cannot normally use these volumes to satisfy nonspecific mount requests unless EDGUX100 selects a pool or a storage group scratch pool is selected.

A *scratch pool* is shelf space assigned to hold volumes for use with DFSMSrmm system based scratch pooling. The volumes assigned to this shelf space can be used to satisfy scratch requests as long as the volumes are in scratch status. Once the volume has been written to, it becomes a volume with MASTER status until the data is no longer required by the installation. The volume remains in the same DFSMSrmm scratch pool in that it occupies the same shelf space regardless of status.

Default: TYPE(*S*)

Chapter 9. Authorizing DFSMSrmm Users and Ensuring Security

To protect DFSMSrmm functions, you need to use an external security product, such as RACF. Invocations to the product are made by DFSMSrmm through the System Authorization Facility (SAF). If RACF is not installed, you must provide equivalent function via the SAF interface described in “Using the SAF Interface” on page 191.

This topic explains how you can define RACF profiles to protect DFSMSrmm functions. Define RACF profiles and authorize DFSMSrmm users to various levels of access, either CONTROL, READ, ALTER, or UPDATE. These access levels vary depending on the security requirements of your installation.

In addition, “Controlling RACF Tape Profile Processing” on page 184 describes how to use the DFSMSrmm parmlib OPTION TPRACF command and VLPOOL command and your installation’s RACF options to control the actions that DFSMSrmm takes on RACF tape profiles.

The following sections in this topic also explain how to ensure security:
- “Creating Audit Trails” on page 183
- “Using Security Classification Processing” on page 184
- “Preventing the Use of IEHINITT” on page 184

Protecting DFSMSrmm Resources with RACF Profiles

The DFSMSrmm resources you protect with RACF profiles in the FACILITY class each have an entity name prefixed with STGADMIN.EDG. Table 20 lists the DFSMSrmm resources.

For optimal security, define RACF profiles to control access to DFSMSrmm functions that are protected by the DFSMSrmm resource. If there is a RACF profile, specific or generic, that matches a DFSMSrmm resource, the resource is treated as if it has been defined. DFSMSrmm provides control for some resources, as described in “Setting the Level of Access for the DFSMSrmm Resources” on page 173, even when you do not protect DFSMSrmm resources with RACF or another equivalent security product.

| The TSO commands and utilities are authorized by DFSMSrmm on the system on
| which you run them.

Table 20. Resources You Protect with RACF Profiles

Define the Resource	To Control the
STGADMIN.EDG.FORCE	Changing of information recorded by DFSMSrmm during O/C/EOV processing. Adding or deleting data sets on volumes.
STGADMIN.EDG.HOUSEKEEP	Use of DFSMSrmm inventory management functions.
STGADMIN.EDG.HOUSEKEEP.RPTEXT	Use of DFSMSrmm inventory management extract function

Table 20. Resources You Protect with RACF Profiles (continued)

Define the Resource	To Control the
STGADMIN.EDG.IGNORE.TAPE.*volser*	Use of duplicate volume serial numbers and to allow a volume to be ignored. **Recommendation:** Do not assign an access level to the STGADMIN.EDG.IGNORE.TAPE.*volser* resource to any specific user group. When a tape volume that must be ignored by DFSMSrmm is identified, grant the user or user group the needed access level. Once the volume is no longer needed, delete the resource.
STGADMIN.EDG.IGNORE.TAPE.RMM.*volser*	Use of duplicate volume serial numbers and to allow a volume to be ignored. **Recommendation:** Specify UACC(NONE) to the STGADMIN.EDG.IGNORE.TAPE.RMM.*volser* resource. Grant a user or user group the needed access level only when access is needed. When the volume is no longer needed, delete the resource.
STGADMIN.EDG.IGNORE.TAPE.NORMM.*volser*	Use of volume serial numbers that are not defined to DFSMSrmm to allow a volume to be ignored.
STGADMIN.EDG.LABEL.*volser*	Creation of standard tape labels. The variable *volser* can be specified as a specific volume serial number or a generic volume serial number. For example, A12345 is a specific volume serial number and *AB** is a generic volume serial number. If you use generic profiles you can use these functions in a subset of your volumes. If the volume serial numbers and rack numbers match, you can control relabeling at the pool level. For example you could have a pool using rack number prefix AB*. If you want to create an AL tape and your installation has an SL scratch pool, you need ALTER access to STGADMIN.EDG.LABEL.*volser*. The *volser* can be specified as the pool prefix of the scratch pool. If you want to switch to an AL tape from either an SL or NL tape that has already been assigned to you, UPDATE access to STGADMIN.EDG.LABEL.*volser* is required.
STGADMIN.EDG.LISTCONTROL	Use of the RMM LISTCONTROL subcommand to display DFSMSrmm control data set control record information and EDGRMMxx parmlib settings.
STGADMIN.EDG.MASTER	Access to information in the DFSMSrmm control data set. Assign the control data set a universal access of NONE so that DFSMSrmm grants access to various functions through STGADMIN.EDG.MASTER.
STGADMIN.EDG.NOLABEL.*volser*	Creation of tapes without labels.
STGADMIN.EDG.OPERATOR	Use of the initialize and erase functions.

Table 20. Resources You Protect with RACF Profiles (continued)

Define the Resource	To Control the
STGADMIN.EDG.OWNER.*userid*	Access to owned resources. DFSMSrmm checks this entity only if the command issuer is not the owner of the resource and does not have CONTROL access to STGADMIN.EDG.MASTER. Use of the RMM CHANGEVOLUME subcommand to update information based on the owner. Using STGADMIN.EDG.OWNER.*userid*, individual owners can permit other users to access owned volumes. An owner can be a group or department as well as an individual. Define owner resources only for those owners who will allow their volumes to be managed by another user.
STGADMIN.EDG.RELEASE	Use of the RMM DELETEVOLUME RELEASE subcommand to process any release actions specified for a volume.
STGADMIN.EDG.RESET.SSI	Use of the RESET facility for removing DFSMSrmm from the system. You can use the facility without defining this resource when you have no security product installed.
STGADMIN.EDG.VRS	Use of the RMM LISTVRS and SEARCHVRS subcommands to obtain information about vital record specifications. Use of the RMM ADDVRS and DELETEVRS subcommands to define or remove vital record specifications.
STGADMIN.EDG.INERS.WRONGLABEL	Processing for volumes mounted with the wrong label.

Creating Profiles

To protect an DFSMSrmm resource, you can create a RACF profile as shown in Figure 68.

```
RDEFINE FACILITY STGADMIN.EDG.OWNER.HSMATH0 UAC(ALTER)
```

Figure 68. Creating a RACF Profile

For the most complete records, create profiles with auditing options that are set to record attempted activities. For example, to ensure that RACF logs all successful and failed attempts to change vital record specifications, create profiles as shown in Figure 69.

```
RDEFINE FACILITY STGADMIN.EDG.VRS UACC(NONE)
   RALT FACILITY STGADMIN.EDG.VRS GLOBALAUDIT(ALL(UPDATE))
```

Figure 69. Creating a RACF Profile with Audit Options

Setting the Level of Access for the DFSMSrmm Resources

When you define the DFSMSrmm resources, you need to authorize levels of access to these resources. DFSMSrmm checks the resource and the level of access to ensure that users are authorized to request certain tasks. For example, if you attempt to change an owned volume, DFSMSrmm checks to ensure that you have at least UPDATE access to resource STGADMIN.EDG.MASTER.

Table 21 shows the access that is required to perform DFSMSrmm functions.

Table 21. Authorized Functions

When You Define	With Access	Then
STGADMIN.EDG.FORCE	Entity not defined	Information previously recorded by DFSMSrmm cannot be changed.
	UPDATE	Information previously recorded by DFSMSrmm can be changed based on access to STGADMIN.EDG.MASTER.
STGADMIN.EDG.HOUSEKEEP	Entity not defined	Same as READ access
	READ	Any of the inventory management facilities can be invoked.
STGADMIN.EDG.HOUSEKEEP.RPTEXT	Entity not defined	Same as READ access
	READ	RPTEXT inventory management function can be invoked.
STGADMIN.EDG.IGNORE.TAPE.*volser*	Entity not defined	Volumes cannot be ignored using the DFSMSrmm installation exit.
	READ	Volumes that are to be ignored by DFSMSrmm for input requests are allowed to be opened.
	UPDATE	Volumes that are to be ignored by DFSMSrmm for output requests are allowed to be opened.
STGADMIN.EDG.IGNORE.RMM.TAPE.*volser*	Entity not defined	Access is based on the STGADMIN.EDG.IGNORE.TAPE.*volser* setting. Use of the STGADMIN.EDG.IGNORE.RMM.TAPE.*volser* profile allows volumes that are defined to DFSMSrmm to be ignored.
	READ	Volumes that are defined to DFSMSrmm can be ignored by DFSMSrmm for input requests are allowed to be opened.
	UPDATE	Volumes that are defined to DFSMSrmm can be ignored by DFSMSrmm for output requests are allowed to be opened.
STGADMIN.EDG.IGNORE.NORMM.TAPE.*volser*	Entity not defined	Access is based on the STGADMIN.EDG.IGNORE.TAPE.*volser* setting. Use of the STGADMIN.EDG.IGNORE.NORMM.TAPE.*volser* profile allows volumes that are not defined to DFSMSrmm to be ignored.
	READ	Volumes not defined to DFSMSrmm that are to be ignored for input requests are allowed to be opened.
	UPDATE	Volumes not defined to DFSMSrmm that are to be ignored for output requests are allowed to be opened.

Table 21. Authorized Functions (continued)

When You Define	With Access	Then
STGADMIN.EDG.LABEL.*volser*	Entity not defined	A volume must be in user status to switch from NL or to change to label types at OPEN time.
	UPDATE	Standard labels can be created on a non-scratch volume for an AL or SL output request.
	ALTER	Standard labels can be created on a scratch volume during a nonspecific volume request for AL or SL.
STGADMIN.EDG.LISTCONTROL	Entity not defined	Functions are based on STGADMIN.EDG.MASTER access.
	CONTROL	You are allowed to use the RMM LISTCONTROL subcommand.
STGADMIN.EDG.MASTER	Entity not defined	Same as CONTROL access.
	READ	The following functions can be performed: • List all control data set information except vital record specifications and control information • Search for all control data set information except vital record specifications • Update your own owner ID details • Request a scratch volume for yourself • Release an owned volume when the STGADMIN.EDG.RELEASE resource is not protected
	UPDATE	Same as READ access, plus: Some non-restricted fields can be updated for owned volumes and data sets based on user ID. See *z/OS DFSMSrmm Guide and Reference*, RMM CHANGEVOLUME subcommand information, for a list of the non-restricted fields. See “Using RACF Options for Authorizing RMM TSO Subcommands” on page 191 for information about changing information using DFSMSrmm command authorization by data set name.
	CONTROL	Same as UPDATE access, plus: You can • Define, change, and delete any control data set entries except vital record specifications • List control information when the STGADMIN.EDG.LISTCONTROL resource is not protected
STGADMIN.EDG.NOLABEL.*volser*	Entity not defined	A volume must be in user status to switch to NL at OPEN time.
	UPDATE	You are allowed to destroy labels on a non-scratch volume for a no label output request.
	ALTER	You are allowed to destroy labels on a scratch volume during a nonspecific volume request for no labels.

Table 21. Authorized Functions (continued)

When You Define	With Access	Then
STGADMIN.EDG.OPERATOR	Entity not defined	Same as UPDATE access.
	NONE	No authority is granted to use EDGINERS to initialize and erase volumes.
	READ	No authority is granted to use EDGINERS to initialize and erase volumes.
	UPDATE	EDGINERS can be used to initialize and erase volumes.
STGADMIN.EDG.OWNER.*userid*	Entity not defined	No authority is granted to update volume information except based on STGADMIN.EDG.MASTER access.
	NONE	Based on STGADMIN.EDG.MASTER access.
	UPDATE	Some non-restricted fields for volumes and data sets owned by *userid* can be updated. Also a volume owned by *userid* can be released. See *z/OS DFSMSrmm Guide and Reference*, RMM CHANGEVOLUME subcommand information, for a list of the non-restricted fields. See “Using RACF Options for Authorizing RMM TSO Subcommands” on page 191 for information about changing information using DFSMSrmm command authorization by data set name.
STGADMIN.EDG.RELEASE	Entity not defined	Based on STGADMIN.EDG.MASTER access.
	READ	You are allowed to use the RMM DELETEVOLUME RELEASE subcommand to release an owned volume.
STGADMIN.EDG.RESET.SSI	Entity not defined	You cannot use the EDGRESET utility to remove DFSMSrmm from the system. If you have no security product installed, you can use EDGRESET to remove DFSMSrmm from the system.
	ALTER	You are allowed to withdraw DFSMSrmm from the system during error recovery or problem during implementation.
STGADMIN.EDG.VRS	Entity not defined	Same as CONTROL access.
	READ	You are allowed to list and search for all vital record specifications.
	CONTROL	Same as READ access, plus: You can define and delete vital record specifications.
STGADMIN.EDG.INERS.WRONGLABEL	Entity not defined	Use of the EDGINERS EXEC statement PARM IGNORE and RMMPROMPT parameters is denied.
	UPDATE	You are allowed to use the EDGINERS EXEC statement PARM RMMPROMPT parameter.
	CONTROL	You are allowed to use the EDGINERS EXEC statement PARM IGNORE parameter.

Authorizing Resources

Table 22 provides suggestions for authorizing different types of users. Implementing these suggestions depends on your organization's structure, administrative procedures, and security requirements.

Table 22. Suggested Resource Access

Resource	General User	Storage Administrator	System Programmer	Librarian	Inventory Management Functions	Operator
STGADMIN.EDG.FORCE						
	-	U	-	U	-	-
STGADMIN.EDG.HOUSEKEEP						
	-	-	-	-	R	-
STGADMIN.EDG.HOUSEKEEP.RPTEXT						
	-	R	R	R	R	R
STGADMIN.EDG.IGNORE.TAPE.*volser*						
	-	-	-	-	-	-
STGADMIN.EDG.IGNORE.TAPE.RMM.*volser*						
	-	-	-	-	-	-
STGADMIN.EDG.IGNORE.TAPE.NORMM.*volser*						
	-	-	-	-	-	-
STGADMIN.EDG.LABEL.*volser*						
	U	A	A	A	A	A
STGADMIN.EDG.LISTCONTROL						
	C	C	C	C	-	C
STGADMIN.EDG.MASTER						
	R	C	C	C	-	C
STGADMIN.EDG.NOLABEL.*volser*						
	U	A	A	A	A	A
STGADMIN.EDG.OPERATOR						
	-	-	U	U	-	U
STGADMIN.EDG.OWNER.*userid*						
	-	U	-	-	-	-
STGADMIN.EDG.RELEASE						
	R	-	-	-	-	-
STGADMIN.EDG.RESET.SSI						
	-	-	-	-	-	A
STGADMIN.EDG.VRS						
	R	C	C	C	-	-
STGADMIN.EDG.INERS.WRONGLABEL						
	-	-	-	U	-	-

Access: R—Read, U—Update, C—Control, A—Alter

The following sections describe the functions available to DFSMSrmm users that are based on the recommendations in Table 22 on page 177.

General User Functions

Table 23 describes general user functions.

Table 23. General User Functions

With	The General User Can
UPDATE access to STGADMIN.EDG.LABEL.*volser*	Create standard labels on a non-scratch volume for an AL or SL output request.
CONTROL access to STGADMIN.EDG.LISTCONTROL	Display control information and installation options and rules.
READ access to STGADMIN.EDG.MASTER	Display all control data set details except vital record specifications and control information. Search for data sets, software products, shelf locations, and volumes. Update their owner ID details. Request a scratch volume.
UPDATE access to STGADMIN.EDG.NOLABEL.*volser*	Erase labels on a non-scratch volume for a no label output request.
READ access to STGADMIN.EDG.RELEASE	Release volumes they own.
READ access to STGADMIN.EDG.VRS	Search for any vital record specification and display the details that DFSMSrmm records.

Storage Administrator Functions

Table 24 describes storage administrator functions.

Table 24. Storage Administrator Functions

With	The Storage Administrator Can
ALTER access to STGADMIN.EDG.LABEL.*volser*	Create standard labels on a scratch volume during a non-specific volume request for an AL or SL volume.
CONTROL access to STGADMIN.EDG.LISTCONTROL	Display control information and installation options and rules.
CONTROL access to STGADMIN.EDG.MASTER	Display all control data set details. Request scratch volumes. Release volumes. Update volume, data set, owner, and software product details. Add and delete information about volumes, data sets, shelf locations, owner IDs, and software products.
ALTER access to STGADMIN.EDG.NOLABEL.*volser*	Erase labels on a scratch volume during a non-specific volume request for a no label volume.
UPDATE access to STGADMIN.EDG.OWNER.*userid*.	Update details that DFSMSrmm records for volumes that *userid* owns and release volumes that *userid* owns. The storage administrator might use this level of authorization rather than CONTROL access to STGADMIN.EDG.MASTER for administrators responsible for specific production applications.
CONTROL access to STGADMIN.EDG.VRS	Create and delete vital record specifications.
READ access to STGADMIN.EDG.HOUSEKEEP.RPTEXT	Can obtain information from the control data set that can be used as input for creating reports.

System Programmer Functions

Table 25 describes system programmer functions.

Table 25. System Programmer Functions

With	The System Programmer Can
ALTER access to STGADMIN.EDG.LABEL.*volser*	Create standard labels on a scratch volume during a non-specific volume request for an AL or SL volume.
CONTROL access to STGADMIN.EDG.LISTCONTROL	Display control information and installation options and rules
CONTROL access to STGADMIN.EDG.MASTER	Display all control data set details. Request scratch volumes. Release volumes. Update volume, data set, owner, and software product details. Add and delete information about volumes, data sets, shelf locations, owner IDs, and software products.
ALTER access to STGADMIN.EDG.NOLABEL.*volser*	Erase labels on a scratch volume during a non-specific volume request for a no label volume.
UPDATE access to STGADMIN.EDG.OPERATOR	Can initialize and erase volumes.
CONTROL access to STGADMIN.EDG.VRS	Create, display, and delete vital record specifications.
READ access to STGADMIN.EDG.HOUSEKEEP.RPTEXT	Can obtain information from the control data set that can be used as input for creating reports.

Librarian Functions

Table 26 describes librarian functions.

Table 26. Librarian Functions

With	The Librarian Can
ALTER access to STGADMIN.EDG.LABEL.*volser*	Create standard labels on a scratch volume during a non-specific volume request for an AL or SL volume.
CONTROL access to STGADMIN.EDG.LISTCONTROL	Display control information and installation options and rules.
CONTROL access to STGADMIN.EDG.MASTER	Display all control data set details. Request scratch volumes. Release volumes. Update volume, data set, owner, and software product details. Add and delete information about volumes, data sets, shelf locations, owner IDs, and software products.
ALTER access to STGADMIN.EDG.NOLABEL.*volser*	Erase labels on a scratch volume during a non-specific volume request for a no label volume.
UPDATE access to STGADMIN.EDG.OPERATOR	Initialize and erase volumes.
CONTROL access to STGADMIN.EDG.VRS	Search, display, add, and delete vital record specifications.
READ access to STGADMIN.EDG.HOUSEKEEP.RPTEXT	Can obtain information from the control data set that can be used as input for creating reports.

Inventory Management Functions

Table 27 describes inventory management functions.

Table 27. Inventory Management Functions

With	The Person Performing Inventory Management Can
ALTER access to STGADMIN.EDG.LABEL.*volser*	Create standard labels on a scratch volume during a non-specific volume request for an AL or SL volume.
READ access to STGADMIN.EDG.HOUSEKEEP	Invoke the DFSMSrmm inventory management facilities to perform the following functions: • Create an extract of the control data set. • Select volumes to be retained and moved as vital records or for disaster recovery. • Perform expiration processing. • Perform storage location management processing. • Back up the control data set.
ALTER access to STGADMIN.EDG.NOLABEL.*volser*	Erase labels on a scratch volume during a non-specific volume request for a no label volume.
READ access to STGADMIN.EDG.HOUSEKEEP.RPTEXT	Can obtain information from the control data set that can be used as input for creating reports.

Operator Functions

Table 28 on page 182 describes operator functions.

Table 28. Operator Functions

With	The Operator Can
ALTER access to STGADMIN.EDG.LABEL.*volser*	Create standard labels on a scratch volume during a non-specific volume request for an AL or SL volume.
CONTROL access to STGADMIN.EDG.LISTCONTROL	Display control information and installation options and rules
CONTROL access to STGADMIN.EDG.MASTER	Display all control data set details. Request scratch volumes. Release volumes. Update volume, data set, owner, and software product details. Add and delete information about volumes, data sets, shelf locations, owner IDs, and software products.
ALTER access to STGADMIN.EDG.NOLABEL.*volser*	Erase labels on a scratch volume during a non-specific volume request for a no label volume.
UPDATE access to STGADMIN.EDG.OPERATOR	Initialize and erase volumes.
ALTER access to STGADMIN.EDG.RESET.SSI	Remove DFSMSrmm from the system.
READ access to STGADMIN.EDG.HOUSEKEEP.RPTEXT	Can obtain information from the control data set that can be used as input for creating reports.

Normally, the operator performs these functions by using prepared procedures or batch jobs. In these cases, the user ID performing the procedure requires the relevant access; not the operator's user ID. For example, the EDGRESET function is invoked by starting the DFSMSrmm procedure with OPT=RESET. The user ID that the DFSMSrmm procedure uses needs the access to STGADMIN.EDG.RESET.SSI.

Using the Tape Relabeling Resources

DFSMSrmm supports three basic label types: NL, SL, and AL. You can switch between any of these basic types by specifying the new label type in your JCL when creating the first file on a volume.

Using EDGINERS to relabel tape volumes requires a separate mount of the volume and must be performed prior to the use of the volume.

Changing volume labels at OPEN time when creating the first file does not require EDGINERS or the INIT release action. Tape volume labels can be created or destroyed at any time regardless of volume status when the user has the required access to a security resource. Scratch volumes can only be used for nonspecific output requests.

To enable this processing, use the STGADMIN.EDG.LABEL.*volser* and STGADMIN.EDG.NOLABEL.*volser* profiles in the FACILITY class. These profiles are described in Chapter 9, "Authorizing DFSMSrmm Users and Ensuring Security," on page 171. If you do not create the appropriate security profiles in FACILITY class, the volume must be in user status.

When DFSMSrmm allows the tape labels to be created or destroyed, it automatically replies to the operator WTOR to provide the required information. In addition, for any volume with the return to scratch release action, which has a NL label type at release time, DFSMSrmm sets the INIT release action to ensure that only standard label tapes are returned to the scratch pool.

Your installation can bypass this processing for any volume by using the DFSMSrmm EDGUX100 installation exit to request that DFSMSrmm ignore the volume. EDGUX100 is called before any tape label conflicts are resolved. See "Using the DFSMSrmm EDGUX100 Installation Exit" on page 221.

Creating Audit Trails

There are several ways you can create audit information with DFSMSrmm. You can use the following sources:

- Control data set information
- SMF audit records
- RACF audit information

See *z/OS DFSMSrmm Reporting* for information about creating reports that use DFSMSrmm control data set information and SMF audit records as input.

Control Data Set Information

Use control data set information as input to inventory reports and movement reports to keep track of volumes in your removable media library.

You can also obtain control data set information by using RMM TSO subcommands and the DFSMSrmm ISPF dialog. Here is some of the information available:

- User ID, system, date, and time stamp of the last update before the audit record was created.
- Create date, time, user ID, and system.
- Storage location data and bin numbers.
- Original expiration date and the current expiration date. The original expiration date is the expiration date coded in the JCL when the data was originally written to the volume.
- Current location of a volume.
- Release actions for a volume.

See *z/OS DFSMSrmm Guide and Reference* for information on using the RMM TSO subcommands.

SMF Audit Information

Using the DFSMSrmm audit facility, you can record in SMF records all the updates your users make to the control data set. To specify that you want auditing, use the SMFAUD installation parameter in the EDGRMMxx parmlib member. You can collect records through your installation's normal SMF processing and use them as input to the DFSMSrmm program EDGAUD. An SMF audit record can capture any change to the DFSMSrmm inventory.

DFSMSrmm provides several mapping macros for DFSMSrmm SMF audit and security records. See *z/OS DFSMSrmm Reporting* for the layout of these macros. See "Updating SMFPRMxx (Optional)" on page 28 for information on how identify records to be collected. See “Defining Security Classes: SECCLS” on page 159 for information on defining which SMF record numbers to use.

RACF Audit Information

Using standard RACF facilities, you can keep an audit trail of any command and authorization violations that occur in RMM TSO subcommands or utilities. Use RACF AUDIT and GLOBALAUDIT options to get RACF to create this information.

Using Security Classification Processing

Certain classifications of data might require special processing, or operator action, or both. You can control these actions by using DFSMSrmm security classifications, defined with the SECCLS command in the EDGRMMxx parmlib member. Use this command to specify that DFSMSrmm:

- Issues messages when certain data sets are opened so the operator can take any required installation action before confirming the use of the volume
- Creates a special SMF record for each access to secure volumes
- Automatically flags these volumes to be erased when they are released, before they can be reused

Preventing the Use of IEHINITT

You no longer need to use IEHINITT because the DFSMSrmm utility EDGINERS labels tapes and validates mounted volumes. Use EDGINERS to process new volumes, volumes where old and new labels are known to DFSMSrmm, or old volumes that have been degaussed. EDGINERS defines volumes to the DFSMSrmm control data set that have not been defined previously. For existing volumes, EDGINERS updates the volume record to show that volumes have been initialized. See Chapter 16, “Initializing and Erasing Tape Volumes,” on page 351 for information about using EDGINERS.

To protect IEHINITT from being used or to limit usage to specific individuals, add RACF program resource profiles and create limited access lists as shown in Figure 70:

```
RDEFINE PROGRAM IEHINITT UACC(NONE)
PE IEHINITT CLASS(PROGRAM) RESET(STANDARD)
```

Figure 70. Limiting the Use of IEHINITT

If program control is not active, use the RACF command SETROPTS WHEN(PROGRAM) to activate it. If program control is already active, specify the RACF command SETROPTS WHEN(PROGRAM) REFRESH to activate use of the new profile.

Controlling RACF Tape Profile Processing

You can control the actions that DFSMSrmm takes on RACF tape profiles through the DFSMSrmm OPTION TPRACF command and your installation’s RACF options. For more information on OPTION TPRACF, see “Defining System Options: OPTION” on page 134.

If you are running DFSMSrmm with DFSMShsm, see “Securing Tapes When Running DFSMShsm and DFSMSrmm” on page 269. See *z/OS Security Server RACF System Programmer’s Guide* for information about RACF tape security.

RACF provides the following tape protection options:
- No protection
- TAPEVOL class
- TAPEDSN option
- TAPEVOL and TAPEDSN

To protect volumes, you can specify one of the following actions:

No protection
If neither TAPEVOL nor TAPEDSN is active then DFSMSrmm takes no action, regardless of the setting of the TPRACF option.

TAPEVOL
TPRACF(N)—no action.

TPRACF(P)—a TAPEVOL profile is created or deleted only as a result of issuing a DFSMSrmm ADD, CHANGE, or DELETE subcommand for a private volume from a RACF-protected pool. This includes scratch tapes that are assigned by the user or librarian by using the RMM GETVOLUME subcommand.

If you require tape volume protection for volumes that are used for nonspecific tape requests, you can either:

1. Specify the JCL PROTECT=YES option, and DFSMSdfp processing protects the volume.

   Or
2. Leave DFSMSrmm to protect the volume when a data set on the volume is closed, by creating a TAPEVOL profile.

DFSMSrmm automatically deletes any TAPEVOL profile for recycled scratch tapes when they return to scratch. If you use nonspecific mounts and scratch pools, no TAPEVOL profile exists while the volume is in scratch status.

TPRACF(A)—The same as TPRACF(P).

TAPEDSN
TPRACF(N)—no action.

TPRACF(P) and TPRACF(A)—The processing is the same for both options. DFSMSrmm does not cause any RACF profiles to be created at any time. During recycling or releasing of tapes, DFSMSrmm checks for discrete RACF data set profiles for data sets known to be on the volume, and automatically deletes them.

TAPEVOL and TAPEDSN
TPRACF(N)—no action.

TPRACF(A)—TAPEVOL profiles are created for identified, RACF- controlled, pools of non-scratch tapes when you use the RMM ADD, CHANGE and DELETE subcommands. No TVTOC is created in these circumstances.

DFSMSrmm assumes that correct DFSMSdfp and RACF processing, when a data set on a volume is opened, result in both TAPEVOL and discrete DATASET profiles being created. DFSMSrmm automatically deletes any such profiles for recycled scratch tapes when they return to scratch.

If a volume is unprotected when a data set on the volume is closed, DFSMSrmm automatically protects the volume with a TAPEVOL profile. If it is the first file of an IBM standard label tape being processed, DFSMSrmm creates a TVTOC containing an entry for the first file. This enables the installation to use RACF generic data set profiles to control access to the tape data sets, even when the JCL does not include the PROTECT=YES option.

TPRACF(P) processing is the same as TPRACF(A) except that all scratch tapes, whether defined by RMM TSO subcommand or returned to scratch by expiration processing, are protected by predefined RACF TAPEVOL profiles. The TAPEVOL profile includes an empty TVTOC so that RACF considers the volume to be scratch.

Table 29 shows DFSMSrmm processing when RACF is active and OPTION TPRACF is set for volume pools identified as DFSMSrmm and RACF-managed. The processing is dependent on the combinations of RACF TAPEVOL class and TAPEDSN option, and the TPRACF value. The DFSMSrmm EDGRMMxx parmlib VLPOOL RACF(Y) is in effect.

Table 29. RACF Processing Performed by DFSMSrmm

Command or Function	TAPEVOL	TAPEDSN	TAPEVOL and TAPEDSN
ADDVOLUME MASTER	• Create TAPEVOL profile • Add access list built using the owner, user, and access information	No processing	As for TAPEVOL
ADDVOLUME USER	• Create TAPEVOL profile • Add access list built using the owner, user, and access information	No processing	As for TAPEVOL
ADDVOLUME MASTER PREVVOL	Add to tape volume set	No processing	As for TAPEVOL
ADDVOLUME USER PREVVOL	Add to tape volume set	No processing	As for TAPEVOL
ADDVOLUME SCRATCH	No processing	No processing	If TPRACF(P) • Create TAPEVOL profile • Add a TVTOC if SL or AL.
DELETEVOLUME FORCE / REMOVE	Delete TAPEVOL profile	If data set records, delete tape data set profiles	• Delete TAPEVOL profile • If data set records, delete tape data set profiles
DELETEVOLUME RELEASE	No processing	No processing	No processing
CHANGEVOLUME RACK Change of pool—RACF(Y) to RACF(N)	Delete TAPEVOL profile	If data set records, delete tape data set profiles	• Delete TAPEVOL profile • If data set records, delete tape data set profiles
CHANGEVOLUME RACK Change of pool—RACF(N) to RACF(Y)	• Create TAPEVOL profile • Add access list built using the owner, user, and access information	No processing	As for TAPEVOL
CHANGEVOLUME OWNER OWNERACC PROT USERS	Replace access list	No processing	As for TAPEVOL

Table 29. RACF Processing Performed by DFSMSrmm (continued)

Command or Function	TAPEVOL	TAPEDSN	TAPEVOL and TAPEDSN
CHANGEVOLUME USER / MASTER (not from SCRATCH)	No processing	No processing	No processing
CHANGEVOLUME USER / MASTER (from SCRATCH) GETVOLUME	• Delete TAPEVOL profile • Create TAPEVOL profile • Add access list	If data set records, delete tape data set profiles	As for TAPEVOL
CHANGEVOLUME PREVVOL	• Delete TAPEVOL profile • Add volume to tape volume set	No processing	As for TAPEVOL
SCRATCH mount processing	At close end-of-volume create a TAPEVOL profile if one does not exist and add OWNER as accessor	No processing	At close/end-of-volume create a TAPEVOL profile if one does not exist and create a TVTOC containing first file data set and add OWNER as accessor
MASTER / USER mount processing	No processing	No processing	No processing
Release processing When volume returns to scratch	Delete TAPEVOL profile	If data set records, delete tape data set profiles	• Delete TAPEVOL profile For TPRACF(P) • Create TAPEVOL profile • Add a TVTOC if SL or AL.
DELETEDATASET	No processing	Delete tape data set profile	As for TAPEDSN
DELETEOWNER NEWOWNER	As for CHANGEVOLUME OWNER	As for CHANGEVOLUME OWNER	As for CHANGEVOLUME OWNER

Recommendations for Using RACF Tape Profile Processing

For optimal security, make TAPEVOL and TAPEDSN active with either TPRACF(P) or TPRACF(A) to:

- Obtain full protection at the volume and the data set level
- Avoid the need to predefine volumes to individual users
- Avoid the need to use ADSP or PROTECT=YES which eliminates exit code that is not standard
- Use generic tape data set profiles

Although we discourage this, it is possible to have no tape protection, or to use only TAPEVOL profiles. If you have no tape security, you cannot control the creation and use of data on tape. If you use only TAPEVOL profiles to protect tape volumes, you must maintain the access lists in the TAPEVOL profiles to allow access to data. Consider the use of TAPEDSN so that access to tape data is covered by your normal, existing, generic data set profiles.

There is a potential security exposure with scratch volumes that have no RACF profile, but with DFSMSrmm active in protect mode, you can prevent reading of scratch tapes.

You can use TAPEDSN on its own to provide data set level security. RACF cannot, however, guarantee full data set name integrity (only the last 17 characters of the data set name that are recorded in the tape label). Run DFSMSrmm in protect mode to ensure that full 44-character data set names are validated. With TAPEDSN only, you lack control of access to tape volumes at the volume level. If you do not use protect mode, your system security could be circumvented and tape data could be accessed.

Rejecting Volumes on Specific Systems in a System Complex

In a system complex where all systems share the RACF data set, the RACF profile provides the same protection for all systems. If you want to prevent a volume from being used on one system, you must prevent its use on all systems. However, you can prevent volume usage on individual systems by using the DFSMSrmm REJECT command in the EDGRMMxx parmlib member. You can reject volumes that are defined to DFSMSrmm based on your chosen pool prefix using the REJECT command in parmlib.

To protect several volumes across all systems in the complex, you can use generic RACF profiles. **Example:** Create a profile that prevents any tape starting with a volume serial number prefix AB from being used on all systems in a system complex.

```
RDEFINE TAPEVOL AB* UACC(NONE)
```

To prevent a pool of volumes from being used on one system in a multisystem complex, do not define generic RACF TAPEVOL profiles. Use the DFSMSrmm REJECT parmlib option as shown in Figure 71.

```
REJECT ANYUSE(AB*)
```

Figure 71. Using the REJECT Parmlib Option

Protect volumes by using a combination of generic TAPEVOL profiles that you create, the discrete TAPEVOL profiles that DFSMSrmm creates, and the DFSMSrmm REJECT commands that you define.

Maintaining the User Access List

For DFSMSrmm to maintain the access list, use the RMM TSO subcommands to change access lists, rather than RACF. You can use RACF commands to add users and owners if the times when DFSMSrmm updates, deletes, or creates the TAPEVOL profiles are well defined, and during the time a volume is not scratch. The RACF profile is updated only if the DFSMSrmm volume access list is updated.

If you set a DFSMSrmm option for TPRACF, other than N, DFSMSrmm ensures that your tapes are protected by RACF. DFSMSrmm ensures that all non-scratch tapes are protected by a discrete RACF TAPEVOL profile. DFSMSrmm checks that a RACF profile exists whenever a data set is written on a tape. If a profile does not exist, DFSMSrmm creates one. Therefore you do not need to use RACF installation exits to set the JCL PROTECT=YES option or specify PROTECT=YES in your JCL. Additionally, because DFSMSrmm creates a TVTOC when the RACF TAPEDSN option is used, you can use generic data set profiles for all tape data sets without changes to JCL or installation procedures.

Be careful about using RACF profiles to maintain a list of users who own and can access volumes. When you use the RMM ADDVOLUME and CHANGEVOLUME subcommands, you can maintain up to 12 users and owners for each volume. If the volume is in a RACF-controlled pool and RACF TAPEVOL class is active, the TAPEVOL profile access list is maintained with the list of users who can access the volume and the owner's user ID. If you change the RACF TAPEVOL profile access list using RACF commands, the DFSMSrmm control data set does not reflect the changes. The next time that DFSMSrmm updates the RACF TAPEVOL profile, it creates the access list from the volume information, but does not include any users or owners you added using RACF commands.

Using RACF With DFSMSrmm

Despite the varied tape security support that RACF provides, many installations use RACF exits to control access to tape volumes and to tape data sets. Although there are likely to be many different implementations, these are some of the commonly implemented functions:

- Use of the RACHECK exit or the RACDEF exit to test or set the PROTECT=YES option
- Use of the ability to model a TAPEVOL profile on another existing profile; either a model or data set profile
- Use of RACHECK post-processing exit to prevent use of tapes that are not protected by RACF (PROTECTALL for tape)

DFSMSrmm RACF Tape Security Support

The objective for DFSMSrmm RACF tape security support is to provide a complete interface with RACF for tapes so you can use any combination of valid RACF options, including use of any RACF installation exits you still have. You can tell DFSMSrmm not to provide any RACF support by specifying the DFSMSrmm EDGRMMxx parmlib OPTION TPRACF(N) command. You can set up the type of processing wanted by requesting automatic TPRACF(A) or predefined TPRACF(P) RACF profiles.You can also use the DFSMSrmm EDGRMMxx parmlib VLPOOL RACF command to provide control of RACF at the pool level.

DFSMSrmm Automatic Tape Security Support Processing

DFSMSrmm automatic tape security processing assumes that when a tape is mounted for output as a scratch tape it will not be RACF protected. During the open processing for the tape data set either DFSMSdfp or your installation exits will cause some RACF profiles to be created. DFSMSrmm predefined processing also ensures that, when TAPEVOL and TAPEDSN are active, RACF predefined TAPEVOL profiles with an empty TVTOC are created for scratch volumes. To enforce a DFSMSrmm standard, that all tapes are RACF protected when the data set is closed, DFSMSrmm checks that a RACF security profile exists for the volume. If a RACF security profile does not exist, DFSMSrmm creates one and places the owner of the tape in the access list for the TAPEVOL profile with ALTER authority. The process ensures that your current tape security mechanism should continue to work as long as it is based on TAPEVOL profiles.

Data Set Profile Processing Implications

If you only use data set profiles for tape data sets, you need to consider the implications which are described in this section.

The objective should be to get to standard RACF tape security without using installation exits. However, the introduction of the TAPEDSN option or use of TAPEDSN only can cause some complications. When TAPEDSN is in effect and a TAPEVOL profile with no TVTOC exists, only the owner can access the data set.

This happens because RACF does not use the data set profiles unless the TAPEVOL profile contains a TVTOC. When no security profile has been created by DFSMSdfp or RACF installation exits, DFSMSrmm creates a TAPEVOL profile with a TVTOC, a UACC(NONE) and the volume owner in the access list. The first file is added to the TVTOC using RACFIND=NO so that any generic data set profiles your installation has can be used with tape data sets. Once the first file entry is created RACF will maintain the TVTOC for any future tape activity on the same volume.

RACF Installation Exit Conversion

Some installations use only data set profiles to protect tape data. They rely on the tape management system to ensure that only valid scratch tapes are used for output. If they use TAPEVOL class without TAPEDSN option in effect, they have RACF exits that change the RACHECK for TAPEVOL class to a RACHECK for the data set being processed. Consequently, they prevent RACDEF in the TAPEVOL class. If your RACDEF exit is used to dummy out the DFSMSdfp RACDEFs that would normally create TAPEVOL profiles, they will also prevent DFSMSrmm from defining TAPEVOL profiles. If this applies to your installation you will have to make changes to your RACF exits before making use of the DFSMSrmm RACF support.

This section contains examples for converting RACF installation exits.

TAPEVOL class active. PROTECT=YES JCL option used

Currently the installation uses RACF exits to model the creation of the RACF TAPEVOL profile on an existing RACF data set profile. Access to tape data is effectively based on data set profiles.

When the DFSMSrmm TPRACF option is activated, volumes that are not covered by the PROTECT=YES option and the TAPEVOL profiles created using the data set modeling are protected by a TAPEVOL profile created by DFSMSrmm. Access to these data sets is based on the TAPEVOL profile which allows owner access only.

If the RACF exits modeling function is turned off, the volumes that were previously protected by a modeled profile are now only protected as described above for the DFSMSrmm created profiles. Most probably, users lose access to tape data.

The suggested approach is to activate the TAPEDSN RACF option. For those volumes where PROTECT=YES is specified, the TAPEVOL profiles have TVTOCs created and data access is via data set profiles which currently exist. The creation of the TAPEVOL profiles by DFSMSrmm is also affected, and DFSMSrmm ensures that a TVTOC is created. The protection for tape volumes protected by DFSMSrmm created profiles is also via data set profiles.

TAPEVOL class active. Exit requested PROTECT=YES option used

If your RACF installation exits are used to ’turn on’ the PROTECT=YES option, and those exits are removed, no DFSMSdfp requests are made to RACF to protect tape data, even if TAPEDSN is activated. All volumes get protected at data set CLOSE time when DFSMSrmm creates TAPEVOL profiles.

Whether or not you choose to use the RACF TAPEDSN option, the results in creating a security environment for your installation are the same before and after you remove your installation exits.

TAPEVOL active. RACF exits using DATASET not TAPEVOL profiles

Conversion to standard options to remove the exits involves deactivating the TAPEVOL class and activating the TAPEDSN option. Use the PROTECTALL option to ensure complete security for tape data. When the TAPEVOL class is inactive, DFSMSrmm does not create TAPEVOL profiles for volumes and cleans up any discrete data set profiles that exist when a tape returns to scratch status. You can rely on DFSMSrmm to validate volumes that are mounted and prevent overwrites, validate 44 character data set names and manage data set and volume expiration.

Using RACF Options for Authorizing RMM TSO Subcommands

Related Reading: See *z/OS DFSMSrmm Guide and Reference* for the authorization for specific RMM TSO subcommands.

DFSMSrmm allows you to set up authorization for the TSO subcommands that you use with DFSMSrmm. In addition to using STGADMIN.EDG resources in the FACILITY class, you can use ownership and RACF DATASET and TAPEVOL class resources to protect resources. You can use the DFSMSrmm parmlib OPTION COMMANDAUTH command to control how ownership and DATASET and TAPEVOL resources are used. You can also use the RACF name-hiding function that is provided by RACF. Use the RACF SETROPTS MLNAMES command to activate the name-hiding function. See “Defining System Options: OPTION” on page 134 for information about the DFSMSrmm parmlib OPTION COMMANDAUTH operand.

Set up the DFSMSrmm authorization and security to control access to the information in the DFSMSrmm control data set. You do not have to perform additional setup tasks for librarians and storage administrators because they typically have CONTROL access to STGADMIN.EDG.MASTER, which allows them to access all resources. To allow general users to access data sets and volumes that they do not own, set up authorization for the general user.

In general, when you use DATASET and TAPEVOL authorization instead of or as well as ownership you do not need to define additional resource profiles because the existing DATASET and TAPEVOL profiles are used. Using command authorization by data set name provides an additional check to determine who is authorized to access information about data sets and volumes in DFSMSrmm. DFSMSrmm checks OWNER profiles or DATASET or TAPEVOL profiles to determine ownership of the data and authorization to access the information.

Using the SAF Interface

DFSMSrmm does not provide any security functions itself but relies on installed security products to process requests. For example, DFSMSrmm relies on the installed security product to confirm that a user is authorized to perform a particular function using an RMM TSO subcommand.

DFSMSrmm uses the MVS SAF interface to perform authorization checks and other security processing, as described in “SAF Calls for Authorization Checking” on page 192. DFSMSrmm issues RACROUTE requests that RACF, or a functionally equivalent security product, can process.

"Protecting DFSMSrmm Resources with RACF Profiles" on page 171 describes security profile names and class name. If you have not installed a security product, you could write a SAF router exit to handle the calls that DFSMSrmm makes to the interface.

DFSMSrmm also uses the SAF interface to create, update, and delete tape-related security profiles and access lists as described in "SAF and RACF Calls for Creating, Updating and Deleting Security Profiles" on page 195. When you update the volume access list in the control data set, DFSMSrmm uses the RACF ICHEINTY macro to delete the entire access list and allows you to use the SAF interface to add the required access list. If your system does not support this function, do not use or update the DFSMSrmm volume access lists contained in the control data set.

SAF Calls for Authorization Checking

DFSMSrmm issues RACROUTE calls to determine if a user is authorized to perform a DFSMSrmm function. The calls are issued in the address space and under the task of the command or utility user. Most SAF calls in the FACILITY class are issued regardless of the state of RACF, the SAF interface, or the FACILITY class.

DFSMSrmm prevents RACF users with the OPERATIONS and PRIVILEGED attributes from gaining authorization to the DFSMSrmm resources in the FACILITY class. Any user attempting to use DFSMSrmm functions must be authorized through the resource access list or through universal access. For all authorization checks, except for EDGRESET, DFSMSrmm issues the RACROUTE request with an ACEE address that identifies an ACEE that has had these attributes removed.

Figure 72 shows the RACROUTE call that DFSMSrmm issues to create an ACEE for a user that is defined to RACF. DFSMSrmm issues this call in the address space of the command issuer or batch utility.

```
RACROUTE REQUEST=VERIFY,ENVIR=CREATE,RELEASE=1.9,
      USERID=ACEEUSER,GROUP=ACEEGRP,PASSCHK=NO,SUBPOOL=(3)
```

Figure 72. Creating an ACEE for a User Defined to RACF

Figure 73 shows the RACROUTE call that DFSMSrmm issues to create an ACEE for a user that is not defined to RACF. DFSMSrmm issues the call using a blank USERID value. DFSMSrmm issues this call in the address space of the command issuer or batch utility.

```
RACROUTE REQUEST=VERIFY,ENVIR=CREATE,RELEASE=1.9,
      USERID=ACEEUSER,PASSCHK=NO,SUBPOOL=(3)
```

Figure 73. Creating an ACEE for a User Not Defined to RACF

Examples: Checking Authorization for Issuers of RMM TSO Subcommands

Example 1: The following example shows authorization checking for issuers of RMM TSO subcommands and users of the utilities when a single RACROUTE is issued. The request is issued in the address space of the command issuer or batch utility.

```
RACROUTE REQUEST=AUTH,CLASS=class,ATTR=level,ENTITY=resource,
         LOG=ASIS
```

Example 2: The following example shows authorization checking for issuers of RMM TSO subcommands and users of the utilities when multiple RACROUTEs are issued. The request is issued in the address space of the command issuer or batch utility.

```
RACROUTE REQUEST=AUTH,CLASS=class,ATTR=level,ENTITY=resource,
       LOG=NOFAIL
```

The variables in the above examples have these meanings:

class **- FACILITY**
: This field names a 9 character variable. The first byte contains the length of the class name, which follows in the next 8 bytes.

level **- READ, UPDATE, CONTROL**
: Access level as described in Table 21 on page 174.

resource
: This field is 39 characters and contains the name of the resource to which access is being checked.

Examples: Checking for Authorization when Additional Security is in Use

Before you begin: See *z/OS Security Server RACF Security Administrator's Guide* for information about how to use the SETROPTS command.

Example 1: The following example shows how authorization checking for RMM TSO subcommands is set. The RACROUTE command is issued in the DATASET class. The request can be issued in the address space where the command is issued or in the DFSMSrmm subsystem address space. When you issue the command in the DFSMSrmm subsystem address space, a third-party RACROUTE is issued.

```
 RACROUTE REQUEST=AUTH,CLASS=DATASET,ATTR=level,ENTITY=resource,
            LOG=ASIS,DSNTYPE=T,FILESEQ=n
```

When a subcommand is issued against a data set name or a volume containing data sets, the RACROUTE is issued in the DATASET class with DSNTYPE=T. When there is no data set information for a MASTER or USER volume, the RACROUTE is issued in the TAPEVOL class. When DSNTYPE=T is coded, the authorization checking that is performed depends on the security product settings such as SETROPTS TAPEDSN, and whether the TAPEVOL class is active.

The variables in the above examples have these meanings:

level
: Is READ or UPDATE depending on the subcommand issued.

resource
: Is the data set name to be processed. When the subcommand is issued against a volume, the data set name is the name of the first file on the volume.

n Is the file sequence number of the data set on the volume.

Example 2: The following example shows how authorization checking for RMM TSO subcommands is set. The RACROUTE command is issued in the TAPEVOL class. The request can be issued in the address space where the command is issued or in the DFSMSrmm subsystem address space. When issued in the DFSMSrmm subsystem address space, a third-party RACROUTE is issued.

```
 RACROUTE REQUEST=AUTH,CLASS=TAPEVOL,ATTR=level,ENTITY=resource,
            LOG=ASIS
```

The variables in the above examples have these meanings:

level
: is READ or UPDATE depending on the subcommand issued.

resource
: Is the volume to be processed.

Example: Checking Authorization to Ignore Volumes

The following example shows the command that you can use to check whether the current user is authorized in order to have DFSMSrmm ignore a volume.

```
RACROUTE REQUEST=AUTH,ENTITY=resource, ACCESS=value,LOG=ASIS
```

The variables in the above examples have these meanings:

resource
: *resource* can be: STGADMIN.EDG.IGNORE.TAPE.*volser*, STGADMIN.EDG.IGNORE.TAPE.RMM.*volser*, or STGADMIN.EDG.IGNORE.TAPE.NORMM.*volser* as described in Table 21 on page 174.

value **- READ,UPDATE**
: One of these values is used, as described in Table 21 on page 174.

volser
: This field is the volume serial number of the current mounted volume or the requested volume. The default value is the mounted volume *volser*. The DFSMSrmm installation exit EDGUX100 can select whether the mounted or the requested volume serial number is used.

Example: Checking for Authorization to Create Label and No Label Volumes

The following example is the RACROUTE call for the STGADMIN.EDG.LABEL.*volser* and STGADMIN.EDG.NOLABEL.*volser* profiles.

```
RACROUTE REQUEST=AUTH,ENTITY=STGADMIN.EDG..volslabeler,
         ACCESS=value,LOG=ASIS
```

The variables in the above examples have these meanings:

label
: The label is either LABEL or NOLABEL.

value **- UPDATE,ALTER**
: One of these values is used, as described in Table 21 on page 174.

volser
: This field is the volume serial number of the current mounted volume.

Example: Checking Authorization to Remove DFSMSrmm from the System

When you do not have a security product installed, you can use the EDGRESET utility to remove DFSMSrmm from the system. To use the EDGRESET utility, you must use the SAF router exit.

The following example is used to check that EDGRESET utility can be used. This request is issued in the address space of the EDGRESET utility.

```
RACROUTE REQUEST=AUTH,CLASS=class,ATTR=ATTR,ENTITY=resource,
         RELEASE=1.8
```

The variables in the above examples have these meanings:

class - **FACILITY**
: This field names a 9 character variable. The first byte contains the length of the class name, which follows in the next 8 bytes.

resource - **STGADMIN.EDG.RESET.SSI**
: This field is 39 characters and contains the name of the resource to which access is being checked.

SAF and RACF Calls for Creating, Updating and Deleting Security Profiles

These are the SAF calls that DFSMSrmm issues to maintain the security protection of tape resources in your installation. They are not used to perform authorization checking. Authorization checking for access to tape data and tape volumes is still the responsibility of the OPEN macro and RACF. All these requests are issued from the DFSMSrmm started procedure address space.

When the requests are issued, the RACF ACEE control block for the DFSMSrmm started procedure has the privileged bit on (ACEEPRIV), so that DFSMSrmm requests are honored without authorization checking being performed.

Calls for TAPEVOL class are only issued if RACF is active and the TAPEVOL class is active. Calls for DATASET class are only issued if RACF is active and the TAPEDSN option is active.

For additional information on the following SAF calls, refer to Table 29 on page 186.

Example: Checking for DATASET Class Resource

```
RACROUTE REQUEST=AUTH,CLASS=class,ENTITY=(resource,PRIVATE),LOG=NONE,
         RELEASE=1.8,DSTYPE=T,FILESEQ=seq,VOLSER=vol
```

The variables in the above examples have these meanings:

class - **DATASET**
: This field names a 9 character variable. The first byte contains the length of the class name, which follows in the next 8 bytes.

resource
: This field is 44 characters and contains the name of the data set for which the protecting resource is requested to be returned.

seq
: The file sequence number for this data set.

vol
: The volume on which the data set resides.

Example: Checking for TAPEVOL Class Resource

```
RACROUTE REQUEST=AUTH,CLASS=class,ENTITY=(resource,PRIVATE),
              LOG=NONE,RELEASE=1.8
```

The variables in the above examples have these meanings:

class - **TAPEVOL**
: This field names a 9 character variable. The first byte contains the length of the class name, which follows in the next 8 bytes.

resource
: This field is 6 characters and contains the name of the volume for which the protecting resource is requested to be returned.

Example: Defining a TAPEVOL Class

```
RACROUTE REQUEST=DEFINE,CLASS=class,ENTITY=resource,TYPE=DEFINE,
         RELEASE=1.8,UACC=NONE
```

The variables in the above examples have these meanings:

class **- TAPEVOL**
: This field names a 9 character variable. The first byte contains the length of the class name, which follows in the next 8 bytes.

resource
: This field is 6 characters and contains the name of the volume for which the resource is being defined.

Example: Defining a TAPEVOL Class Resource When TAPEDSN Is Active

The following example shows how to add the first entry to the TVTOC of the discrete TAPEVOL profile just created if TAPEDSN option is active.

```
RACROUTE REQUEST=DEFINE,CLASS=class,ENTITY=resource,TYPE=DEFINE,
         RELEASE=1.8,UACC=NONE,RACFIND=NO,VOLSER=vol,
         DSTYPE=T,FILESEQ=1,TAPELBL=STD
```

The variables in the above examples have these meanings:

class **- DATASET**
: This field names a 9 character variable. The first byte contains the length of the class name, which follows in the next 8 bytes.

resource
: This field is 44 characters and contains the name of the data set for which the resource is being defined.

vol
: The volume on which the data set resides.

Example: Adding a Tape Volume

The following example is used to add a tape volume to an existing tape volume set.

```
RACROUTE REQUEST=DEFINE,CLASS=class,ENTITY=resource,TYPE=ADDVOL,
         RELEASE=1.8,VOLSER=vol
```

The variables in the above examples have these meanings:

class **- TAPEVOL**
: This field names a 9 character variable. The first byte contains the length of the class name, which follows in the next 8 bytes.

resource
: This field is 6 characters and contains the name of the volume for which the resource is being defined.

vol
: The previous volume in the tape volume set.

Example: Deleting a TAPEVOL Profile

The following example is used to delete a discrete TAPEVOL profile.

```
RACROUTE REQUEST=DEFINE,CLASS=class,ENTITY=resource,TYPE=DELETE,
         RELEASE=1.8
```

The variables in the above examples have these meanings:

class **- TAPEVOL**
: This field names a 9 character variable The first byte contains the length of the class name, which follows in the next 8 bytes.

resource
: This field is 6 characters and contains the name of the volume for which the resource is being deleted.

Example: Deleting a DATASET Profile

The following example is used to delete a discrete DATASET profile.

```
RACROUTE REQUEST=DEFINE,CLASS=class,ENTITY=resource,TYPE=DELETE,
         RELEASE=1.8,VOLSER=vol
```

The variables in the above examples have these meanings:

class **- DATASET**
: This field names a 9 character variable. The first byte contains the length of the class name, which follows in the next 8 bytes.

resource
: This field is 44 characters and contains the name of the data set for which the resource is being deleted.

vol
: The volume on which the data set resided.

Example: Checking for Authorization

Use the commands in Figure 74 to perform the following tasks:

- Create or update the volume access list
- Give the volume a TVTOC
- Rebuild a tape volume set when a volume is removed from within, rather than from the end of the set, when a TVTOC is used

```
RACROUTE REQUEST=EXTRACT,CLASS=class,ENTITY=resource,TYPE=EXTRACT,
         RELEASE=1.8,SUBPOOL=0,SEGMENT=BASE,FIELDS=fields
ICHEINTY ALTER,TYPE='GEN',CLASS=class,ACTIONS=D,ENTRY=resource,
         RELEASE=1.8,OPTIONS=FLDEF
ICHEACTN FIELD=ACLCNT,FLDATA='DEL',GROUP=YES,
         RELEASE=1.8
RACROUTE REQUEST=EXTRACT,CLASS=class,ENTITY=resource,TYPE=REPLACE,
         RELEASE=1.8,SEGDATA=data,SEGMENT=BASE,FIELDS=fields
```

Figure 74. Checking Authorization

The variables in the above examples have these meanings:

class **- TAPEVOL**
: This field is 8 characters and contains the class name.

resource
: This field is 6 characters and contains the name of the volume whose profile is being processed.

fields
: Defines which fields are being extracted and updated. The field names used include: TVTOCCNT,VOLCNT,ACLCNT,VOLSER,OWNER,UACC,ACL.

data
: *data* defines the fields that are being replaced. The field names used include: OWNER,UACC,ACL.

Chapter 10. Using DFSMSrmm Programming Interfaces

DFSMSrmm provides programming interfaces EDGLCSUX, EDGMSGEX, and EDG3X71 to use Object Access Method (OAM), DFSMSdfp, and JES3 installation exits. DFSMSrmm uses the installation exits in Table 30. Before implementing DFSMSrmm, ensure that there are no conflicts between how DFSMSrmm uses these exits and how your installation currently uses them.

DFSMSrmm provides programming interface EDGTVEXT for use from DFSMShsm or Tivoli Storage Manager. DFSMSrmm provides programming interfaces EDGTVEXT and EDGDFHSM for use by applications that want to release tape volumes.

Table 30. Installation Exits Used by DFSMSrmm

Exit	DFSMSrmm Programming interface	DFSMSrmm's Use of the Exit
CBRUXCUA	EDGLCSUX	DFSMSrmm uses OAM's change use attribute exit to manage the volumes in system-managed tape libraries.
CBRUXEJC	EDGLCSUX	DFSMSrmm uses OAM's cartridge eject exit to manage the volumes in system-managed tape libraries.
CBRUXENT	EDGLCSUX	DFSMSrmm uses OAM's cartridge entry exit to manage the volumes in system-managed tape libraries.
CBRUXVNL	EDGLCSUX	DFSMSrmm uses OAM's volume-not-in-library installation exit to process tape volumes that are not resident in system-managed tape libraries but that are needed for processing to continue.
IGXMSGEX	EDGMSGEX	DFSMSrmm uses the DFSMSdfp MSGDISP exit to update tape drive displays and control the use of cartridge loaders.
IATUX71	EDG3X71	DFSMSrmm uses the JES3 exit to determine the processing needed for JES3 messages and tape drive display processing.

When you are converting from another tape management product to DFSMSrmm, you might run DFSMSrmm in parallel with the other tape management system for some time before you complete your conversion. Some installation exits must be used by both systems. DFSMSrmm supplies exits that help you run the CBRUXCUA exit, the CBRUXEJC exit, and the IGXMSGEX exit from an existing tape management product and DFSMSrmm in parallel. When exits are used by both systems, the tape management system that is in control makes the tape management decisions and the second system records the activities. See "Setting Up Parallel Processing" on page 216 for additional information.

Releasing Tapes: EDGTVEXT

DFSMSrmm provides the programming interface EDGTVEXT that is used from DFSMShsm, OAM, or any other APF-authorized program that needs to obtain the same services as the DFSMShsm ARCTVEXT exit.

If you have a product with similar requirements for releasing tapes as DFSMShsm, you can use the EDGTVEXT program interface. You can also use the EDGDFHSM interface. The difference between EDGTVEXT and EDGDFHSM is that EDGTVEXT accepts the ARCTVEXT parameter list and the EDGDFHSM interface accepts only a single volume at a time in the parameter list.

Any caller of EDGTVEXT must be defined to RACF. If the caller is a started task, define the user ID with the STARTED class. Your application must also be authorized to release its own tape volumes.

Related Reading: Refer to Chapter 12, “Running DFSMSrmm with DFSMShsm,” on page 253 for information about setting up DFSMShsm with DFSMSrmm. You can use this information as an example for setting up other applications, including OAM, that manage tape. Refer to Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171 for information about the authorization support available with DFSMSrmm.

You must also consider how volumes are retained until the application calls the EDGTVEXT exit to release the volumes. You could retain tapes by defining vital record specifications like the ones shows in the following examples. The examples define policies to retain all the data until a volume is released by the application.

```
RMM ADDVRS DSN('**') JOBNAME(jobname) LOCATION(CURRENT) DAYS COUNT(99999)
RMM ADDVRS DSN('ABEND') JOBNAME(jobname) LOCATION(CURRENT) DAYS COUNT(99999)
RMM ADDVRS DSN('OPEN') JOBNAME(jobname) LOCATION(CURRENT) DAYS COUNT(99999)
```

You can use DFSMSrmm parmlib option TVEXTPURGE to control the processing that EDGTVEXT performs. You can release volumes or set the volume expiration date to the current date. Refer to TVEXTPURGE in “Defining System Options: OPTION” on page 134.

Invocation

Invoke EDGTVEXT from: LOAD and CALL macros or the LINK macro.

Input

The input is a parameter list that describes the volumes DFSMShsm is releasing and the actions required. The parameter list is identical to the one DFSMShsm passes to ARCTVEXT.

On entry, register 1 contains a pointer to the ARCTVEXT parameter list. See *z/OS DFSMS Installation Exits* for information on ARCTVEXT.

Output

EDGTVEXT issues messages when errors are encountered. EDGTVEXT sets the parameter list return code value to zero. EDGTVEXT does not always pass a zero return code in register 15 back to the caller. EDGTVEXT issues a non-zero return code in the register 15 return code from subsystem requests attempted by EDGTVEXT. You can obtain information about the return codes in *z/OS MVS Using the Subsystem Interface*.

Processing

EDGTVEXT ensures that DFSMSrmm is in use on your system before continuing with DFSMSrmm processing. If DFSMSrmm is not in use, EDGTVEXT sets return code zero and returns to the caller. If DFSMSrmm is in use or should be in use, EDGTVEXT calls EDGDFHSM to process the volumes passed to it in the ARCTVEXT parameter list.

Environment

EDGTVEXT must be link edited in an APF-authorized library. EDGTVEXT runs in AMODE(31) RMODE(ANY).

Managing DFSMShsm Tapes: EDGDFHSM

DFSMSrmm uses the EDGDFHSM programming interface to release DFSMShsm tape volumes. The interface between DFSMShsm and DFSMSrmm is automatically in place when you install DFSMS so you do not need to take any action to activate it or to call it.

Any caller of EDGDFHSM must be defined to RACF. If the caller is a started task, define the user ID with the STARTED class. Authorize your application to release its own tape volumes.

You must also consider how volumes are retained until the application calls the EDGDFHSM exit to release the volumes. You can use this program interface from programs other than DFSMShsm to release tape volumes. Refer to Chapter 12, “Running DFSMSrmm with DFSMShsm,” on page 253 for information about setting up DFSMShsm with DFSMSrmm. You can use this information as an example for setting other applications that manage tape. You could retain tapes by defining vital record specifications like the ones shows in the following examples.

Example: Define policies to retain all the data until a volume is released by the application.

```
RMM ADDVRS DSN('**') JOBNAME(jobname) LOCATION(CURRENT) DAYS COUNT(99999)
RMM ADDVRS DSN('ABEND') JOBNAME(jobname) LOCATION(CURRENT) DAYS COUNT(99999)
RMM ADDVRS DSN('OPEN') JOBNAME(jobname) LOCATION(CURRENT) DAYS COUNT(99999)
```

You can use DFSMSrmm parmlib OPTION TVEXTPURGE operand to control the processing that EDGDFHSM performs. You can release volumes or set the volume expiration date to the current date. Refer to TVEXTPURGE described in “Defining System Options: OPTION” on page 134.

Callers of EDGDFHSM can only request the processing of volumes when their RACF user ID is authorized as defined by the RACF FACILITY class profiles for DFSMSrmm.

Invocation

Invoke EDGDFHSM from: LOAD and CALL macros or the LINK macro.

Input

The input is a parameter list that describes the volume DFSMShsm is releasing and the actions required. The parameter list is similar to the one DFSMShsm passes to ARCTVEXT, except that DFSMSrmm’s parameter list handles only a single volume at a time.

On entry, register 1 contains a pointer to an 8-byte field. The first 6 bytes are the volume serial number to be processed and the last 2 bytes are flag bytes. The contents of the flag bytes are the flag bytes DFSMShsm passes to the ARCTVEXT exit. See *z/OS DFSMS Installation Exits* for information on ARCTVEXT.

On entry, register 13 contains the address of a standard 18 word save area and register 14 contains the return address.

Output

A non-zero return code is the register 15 return code from the subsystem request attempted by EDGDFHSM. You can obtain information about the return codes in *z/OS MVS Using the Subsystem Interface*.

Processing

EDGDFHSM issues messages when it encounters errors, but it does not always pass a non-zero return code back to the caller.

Environment

EDGDFHSM must be link edited in an APF-authorized library. It runs in AMODE(31) RMODE(ANY).

Managing System-Managed Tape Library Volumes: EDGLCSUX

DFSMSrmm uses the OAM installation exits CBRUXCUA, CBRUXEJC, CBRUXENT, and CBRUXVNL as described in Table 31 to manage the tape volumes defined in system-managed tape libraries. DFSMSrmm supplies Assembler source code for these installation exits, which is installed on your system as modules CBRUXCUA, CBRUXEJC, CBRUXENT, and CBRUXVNL. You can modify the source code to include your own function or merge with another product's supplied code.

DFSMSrmm uses EDGLCSUX to support the following functions:

- Tracking TCDB changes by updating the DFSMSrmm control data set.
- Updating the CBRUX*xxx* parameter list based on DFSMSrmm information.
- Partitioning a library.
- Handling volume-not-in-library conditions.
- Controlling the purging of TCDB volume entries at eject time.

You can use the DFSMSrmm EDGRMMxx OPTION SMSTAPE operand to control the support that DFSMSrmm provides. Refer to "Defining System Options: OPTION" on page 134 for information about the SMSTAPE operand. See Appendix B, "DFSMSrmm Mapping Macros," on page 409 for mapping macros you can use with the installation exits.

Table 31. OAM Installation Exits

Exit	Description
CBRUXCUA	The CBRUXCUA exit controls the return to scratch status and updates the DFSMSrmm control data set with information from the TCDB.
CBRUXEJC	The CBRUXEJC exit controls the ejection of a cartridge from a library and updates information in the DFSMSrmm control data set. You can run this exit in parallel with another copy of the exit. See "Setting Up Parallel Processing" on page 216 for more information.
CBRUXENT	The CBRUXENT exit controls the entry of a cartridge into a library, updates information in the DFSMSrmm control data set, and updates DFSMSrmm information in the TCDB. You can run this exit in parallel with another copy of the exit. See "Setting Up Parallel Processing" on page 216 for more information.
CBRUXVNL	The CBRUXVNL exit retrieves information from the DFSMSrmm control data set when a volume that is not in a tape library is needed for processing to continue.

The OAM exits that DFSMSrmm supplies call EDGLCSUX, the DFSMSrmm program that supports the OAM interface. EDGLCSUX is a callable program interface to DFSMSrmm that you can use only from the OAM installation exits.

If you do not have a license for DFSMSrmm, the DFSMSrmm OAM exits set a return code of 16, telling OAM to never call the exit again. For information on licensing, see “Enabling DFSMSrmm” on page 30. If you are licensed for DFSMSrmm, and you have performed some of the installation steps, but have not added EDGSSSI to the IEFSSNxx member of parmlib, or have not started the DFSMSrmm procedure, DFSMSrmm sets return code zero, allowing all OAM requests to be accepted. Once you have performed the installation steps, but do not start the DFSMSrmm subsystem, the OAM exits set a return code 8 and fail all requests.

Input

The input is a parameter list mapped by the macro EDGLCSUP, shown in “OAM Interface: EDGLCSUP” on page 409. The parameter list describes the OAM installation exit calling EDGLCSUX and provides the address of the information passed to the exit.

In the EDGLCSUP macro, the field LCSUP_LCSPL must contain the pointer to the OAM installation exit parameter list, which is the value in register 1 on entry to the OAM exit. The OAM installation exit parameter list macros are used to map this data. See *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries* for more information about CBRUXCPL, CBRUXEPL, CBRUXJPL, and CBRUXNPL.

Complete the LCSUP_FUNCTION field each time EDGLCSUX is called to indicate which OAM exit is calling it. Select from LCSUP_CUA, LCSUP_EJC, LCSUP_ENT, LCSUP_VNL, and LCSUP_ACTVNL.

The parameter list EDGLCSUP includes output fields for the OAM return code LCSUP_LCSRC and the DFSMSrmm reason code LCSUP_LCSRS. The fields LCSUP_LCSRC and LCSUP_LCSRS contain a reason code from DFSMSrmm and a return code to pass back to OAM. The LCSUP_LCSRC field contains the value that is recommended by DFSMSrmm to be returned for this OAM request. The supplied DFSMSrmm OAM installation exits pass back these values. For example, if the LCSUP_LCSRC value is 4, DFSMSrmm has updated the OAM parameter list during processing. You can also use the return and reason codes set in register 15 and register 0 after the call to EDGLCSUX to determine the processing you might want to perform.

On entry, register 13 contains the address of a standard 18 word save area and register 14 contains the return address.

Output

The DFSMSrmm control data set and OAM parameter list are updated depending on the type of processing performed.

Register 15 contains the return code which indicates what processing should be performed next. Register 0 contains the reason code. Use the return code and the reason code to determine the processing that DFSMSrmm has been able to perform. Table 32 on page 204 shows the return and reason codes that EDGLCSUX sets in register 15 and register 0.

Table 32. EDGLCSUX Return and Reason Codes Returned in Register 15 and Register 0

Return Code	Reason Code	Description
0	0	Processing successful. LCSUP_LCSRC contains the return code for OAM. LCSUP_LCSRS contains a reason code that provides information about DFSMSrmm processing. The reason codes are described by variables LCSUP_RS_xxxx listed in the EDGLCSUP macro.
0	1	Processing successful. LCSUP_LCSRC contains the return code for OAM.
0	2	Processing successful. DFSMSrmm is not licensed for use on this processor. LCSUP_LCSRC contains the return code 16 for OAM.
4	0	Processing not performed because the DFSMSrmm subsystem was not available. LCSUP_LCSRC contains the return code for OAM.
8	0	Processing not performed because there was a logic error during processing. LCSUP_LCSRC contains the return code for OAM.
12	Various reason codes set	Processing is unsuccessful. The reason codes are described by variables LCSUP_RS_*xxxx* listed in the EDGLCSUP macro. The supplied exits set RC 8 for OAM to fail requests.

Table 33 defines the return codes generated for each OAM function based on the reason codes set by DFSMSrmm in field LCSUP_LCSRS. The return code variables used in the table are defined in the OAM macros, CBRUXCPL, CBRUXEPL, CBRUXJPL, and CBRUXVNL. If the DFSMSrmm parmlib OPTION SMSTAPE(UPDATE(EXITS)) operand is not set or if DFSMSrmm is running in warning or record-only mode, the OAM return codes UXxFAIL are changed to UXxNOCHG.

Table 33. EDGLCSUX Return and Reason Codes Based on DFSMSrmm Reason Code Setting

Reason Code	Description	Related Message Number	Change Use Return Code	Volume Entry Return Code	Volume Eject Return Code	Volume- Not in- Library
0	DFSMSrmm accepted request	n/a	UXCNOCHG or UXCCHG[1]	UXENOCHG or UXECHG[1]	UXJNOCHG or UXJCHG[1]	UXNNORML or UXNCHG[1]
LCSUP_RS_DEB	Specified destination is not current library	EDG8192I	n/a	UXEFAIL	n/a	n/a
LCSUP_RS_DUPLV	Volume duplicates an existing logical volume	EDG8183I	n/a	UXEFAIL	n/a	n/a
LCSUP_RS_DUPPV	Volume duplicates an existing physical volume	EDG8182I	n/a	UXEFAIL	n/a	n/a

Table 33. EDGLCSUX Return and Reason Codes Based on DFSMSrmm Reason Code Setting (continued)

Reason Code	Description	Related Message Number	Change Use Return Code	Volume Entry Return Code	Volume Eject Return Code	Volume- Not in- Library
LCSUP_RS_DUPSV	The volume duplicates an existing stacked volume.	EDG8181I	UXCFAIL	UXEFAIL	UXJFAIL	n/a
LCSUP_RS_IRK	Volume rack number inconsistent	EDG8189I	UXCFAIL	UXEFAIL	UXJNOCHG	n/a
LCSUP_RS_IVU	User ID not valid for DFSMSrmm	EDG8195I	UXCFAIL	UXEFAIL	UXJNOCHG	n/a
LCSUP_RS_NMV	Volume not to be used with MVS	EDG8191I	UXCFAIL	UXEIGNOR	UXJFAIL	n/a
LCSUP_RS_NOTEXP	Imported volume is not exported	EDG8183I	n/a	UXEFAIL	n/a	n/a
LCSUP_RS_NRM	Volume not defined in a manual tape library	EDG8197I	UXCNOCHG	UXENOCHG	UXJNOCHG	UXNNORML
LCSUP_RS_PBD	Inconsistent parameter list	EDG8190I	UXCFAIL	UXEFAIL	UXJFAIL	UXNFAIL
LCSUP_RS_RIU	Rack to match volser not available	EDG8198I	UXCFAIL	UXEFAIL	UXJNOCHG	n/a
LCSUP_RS_RJP	Undefined volume rejected by reject prefix	EDG8193I	UXCFAIL	UXEIGNOR	UXJNOCHG	n/a
LCSUP_RS_RPX	Retention period exceeds installation maximum	EDG8196I	UXCFAIL	UXEFAIL	UXJNOCHG	n/a
LCSUP_RS_SCR	Private to scratch status not permitted	EDG8194I	UXCFAIL	UXEFAIL	UXJFAIL	n/a
LCSUP_RS_SMM	Entry status and DFSMSrmm status of volume mismatch	EDG8180I	n/a	UXENOCHG	n/a	n/a

Table 33. EDGLCSUX Return and Reason Codes Based on DFSMSrmm Reason Code Setting (continued)

Reason Code	Description	Related Message Number	Change Use Return Code	Volume Entry Return Code	Volume Eject Return Code	Volume- Not in- Library

Notes:

1. When the following conditions are true:
   - Status is private and no expiration date is supplied
   - Status is private and last read or write dates are lower than the DFSMSrmm equivalent dates
   - Status is scratch and owner information exists; the first 8 bytes are set to blanks
   - Volume entered into a system-managed tape library and DFSMSrmm has values for either storage group name, owner, or tape device selection information

Processing

For information about how DFSMSrmm works with system-managed tape libraries, see Chapter 5, “Running DFSMSrmm with System-Managed Tape Libraries,” on page 85. For information about how DFSMSrmm running mode affects system-managed tape library support, see “Defining System Options: OPTION” on page 134.

DFSMSrmm Processing for OAM Support

DFSMSrmm processing is dependent on the following conditions:

- The OPMODE processing mode specified in the DFSMSrmm EDGRMMxx parmlib member. When DFSMSrmm is running in manual mode, no DFSMSrmm processing is performed. When DFSMSrmm is running in warning mode, DFSMSrmm issues messages but does not fail any requests.
- The TCDB purge option specified in the DFSMSrmm EDGRMMxx parmlib member. Use the SMSTAPE(PURGE) option to control the processing that DFSMSrmm performs when a volume is ejected. DFSMSrmm can always keep the record, always purge the record, or accept the requestor’s decision. If the requestor did not make a decision, DFSMSrmm uses the ISMF default value. The ISMF default can be set at the library level. If the TCDB record is kept, DFSMSrmm adds the destination location and bin information to the OAM shelf location information. This avoids the WTOR to the operator prompting for shelf information. If the shelf location is still set to ’DEST=’ during entry processing, DFSMSrmm clears the field. Library partitioning driven by REJECT ANYUSE(*prefix*) and volume not for use on MVS, is performed by DFSMSrmm in all modes except manual mode.
- DFSMSrmm performing library partitioning driven by REJECT ANYUSE(*prefix*) and volume not for use on MVS when DFSMSrmm is running in all modes except manual mode.
- The DFSMSrmm EDGRMMxx parmlib OPTION SMSTAPE(UPDATE(EXITS)) operand. Information provided by OAM to the exits is used to enhance the information recorded by DFSMSrmm. You can decide whether the OAM information overrides the information already defined to DFSMSrmm. Use the SMSTAPE(UPDATE(EXITS)) option to use DFSMSrmm information to override other information and to activate the volume-not-library processing. When DFSMSrmm is running in protect mode, the DFSMSrmm information overrides the information from OAM and performs volume-not-library processing

See “Defining System Options: OPTION” on page 134 for information about the options.

Table 34 on page 207 describes DFSMSrmm processing for OAM support.

Table 34. Processing for the Change Use Attribute, Cartridge Entry and Cartridge Eject Parameter List

Variable	Field Name	Input	Output
Checkpoint volume indicator	UXCCHKPT UXECHKPT UXJCHKPT	Not used	Not used
Installation exit information	UXCEXITI UXEEXITI UXJEXITI	DFSMSrmm uses this for communication between different parts of DFSMSrmm.	Input only
Last entry or ejection date	UXCENTEJ UXEENTEJ UXJENTEJ	DFSMSrmm updates the control data set with this value if no date is recorded or if the input date is later than the existing date. DFSMSrmm uses this to set the movement tracking date.	Input only
Last mounted date	UXCMOUNT UXEMOUNT UXJMOUNT	DFSMSrmm records this date if no date exists or if this date is more current than the existing date.	DFSMSrmm returns a date in the parameter list if the date is less than the date already recorded by DFSMSrmm. DFSMSrmm sets return code UXyCHG to indicate that the parameter list has been changed.
Last written date	UXCWRITE UXEWRITE UXJWRITE	DFSMSrmm records this date if no date exists or if this date is later than the existing date.	If the date is less than the existing date, DFSMSrmm returns the last written date and sets return code UXyCHG to show that the parameter list has been changed.
Library console name	UXCLCON UXELCON UXJLCON	All error messages issued by DFSMSrmm to support the OAM functions are issued to the named console and to consoles using the correct routing codes.	Input only
Library description	UXCLDESC UXELDESC UXJLDESC	Not used	Input only
Library device type	UXCLDEV UXELDEV UXJLDEV	Not used	Input only

Table 34. Processing for the Change Use Attribute, Cartridge Entry and Cartridge Eject Parameter List (continued)

Variable	Field Name	Input	Output
Library logical type	UXCLTYP UXELTYP UXJLTYP	If the volume is being defined to DFSMSrmm resides in a system-managed tape library, the location type identifies if the volume resides in an automated tape library or manual tape library. For a volume that is already defined to DFSMSrmm, DFSMSrmm updates the location type with this value. When a volume residing in a system-managed tape library is not defined to DFSMSrmm, DFSMSrmm creates a volume record. The latest information is recorded by DFSMSrmm when volumes are ejected from the system-managed tape library. This ensures information will be available if the volume is ever entered into any other library controlled by the same control data set.	Input only
Library name	UXCLIB UXELIB UXJLIB	Location name • If the volume is being defined for the first time, DFSMSrmm records this value as the home location and location name for the volume. • If the volume is already defined to DFSMSrmm, DFSMSrmm will update the volume location name to this value if the existing location name is different. If the rack number for the volume is not the same as the volume serial number, then DFSMSrmm sets a return code and reason code and issues message EDG8189I. If the rack number is not available, DFSMSrmm sets a return code and reason code and issues message EDG8198I.	Input only

Table 34. Processing for the Change Use Attribute, Cartridge Entry and Cartridge Eject Parameter List (continued)

Variable	**Field Name**	**Input**	**Output**
New use attribute	UXCUSEA UXEUSEA UXJUSEA	The value can be: **S for scratch** For change use attribute and cartridge eject processing, if the volume currently is defined to DFSMSrmm as a master or user volume, then this request is rejected and DFSMSrmm does not change the use attribute. DFSMSrmm sets a reason code and return code and issues message EDG8194I. For cartridge entry processing, if the volume is currently defined as a master or user volume, then DFSMSrmm returns the volume status recorded in the control data set to the caller. **P for private** The volume information is changed or added to DFSMSrmm regardless of its current status.	This value is input only for change use attribute and cartridge eject processing. DFSMSrmm updates this information during cartridge entry processing if the volume is defined to DFSMSrmm and the status recorded in control data set is different.
Notification call	UXJNCALL	DFSMSrmm uses this field to avoid rejecting or failing an export of a logical volume.	Input only
Shelf location	UXCSHLOC UXESHLOC UXJSHLOC	Not used	Not used Cleared if the location name starts with DEST= Set to DEST=destination name, bin number, and medianame if TCDB record is kept.
Stacked volume	UXJSTKVS	DFSMSrmm uses this field as the ’in container’ value.	Input only
Storage group name	UXCGROUP UXEGROUP UXJGROUP	DFSMSrmm updates the DFSMSrmm control data set with the storage group name, except during cartridge entry processing when the storage group name is already set.	DFSMSrmm updates this value during cartridge entry processing if the storage group name is already set.

Table 34. Processing for the Change Use Attribute, Cartridge Entry and Cartridge Eject Parameter List (continued)

Variable	Field Name	Input	Output
Tape drive selection information	UXCTDSI UXETDSI UXJTDSI	DFSMSrmm uses this value to replace tape drive selection information during change use attribute and cartridge eject processing. Tape drive selection information corresponds to the DFSMSrmm recording format, media type, compaction, and special attributes.	DFSMSrmm uses this value to update tape drive selection information during cartridge entry processing. DFSMSrmm merges the known information from OAM with the information in the control data set to produce tape drive selection information.
Volume attribute	UXEVATTR	Volume type. DFSMSrmm uses the physical volume to process the volume as a real volume. DFSMSrmm uses the logical volume and imported logical volume attributes to identify the volume type as a logical volume. DFSMSrmm checks to see if logical volumes have been correctly imported or entered to control entry and import processing. For imported volumes, the volumes must not already be defined to DFSMSrmm or must be defined as an exported logical volumes. If the volumes are not correctly defined, DFSMSrmm issues message EDG8182I or EDG8184I and appropriate reason codes.	Input only
Volume expiration date	UXCEXPIR UXEEXPIR UXJEXPIR	DFSMSrmm records this value when a new private volume is added or when a volume is changed from scratch to master status. If the volume is already defined as a master volume, DFSMSrmm ignores this value. If the date exceeds the maximum retention period then DFSMSrmm fails the request and issues message EDG8196I and sets return code UXCFAIL.	If the input date is less than the existing date, DFSMSrmm returns the most recent date in the parameter list and sets return code UXyCHG to show that the parameter list has been changed. Note that this is an output field when changing the status to PRIVATE.

Table 34. Processing for the Change Use Attribute, Cartridge Entry and Cartridge Eject Parameter List (continued)

Variable	Field Name	Input	Output
Volume location code	UXCLOC UXELOC UXJLOC	Values can be: **S for SHELF** The volume was ejected from the system-managed tape library and is in transit. **L for LIBRARY** The volume is resident in the system-managed tape library.	Input only
Volume owner information	UXCOWNER UXEOWNER UXJOWNER	The first 8 characters are used to identify the DFSMSrmm owner. If the volume is changing from scratch to master status or is being added as master then DFSMSrmm checks the owner information. If this is a valid DFSMSrmm owner then DFSMSrmm adds or updates the volume owner information in the control data set. If the owner is not a valid owner name, DFSMSrmm sets a default which is the DFSMSrmm user ID or its step name if the DFSMSrmm ACEE cannot be located. If DFSMSrmm is running in protect mode during change use attribute processing, DFSMSrmm rejects the request and issues message EDG8195I and sets the CBRUXCUA return code UXCFAIL. If the volume is a master volume, DFSMSrmm examines the first 8 characters of the owner information. If the owner information is valid but is different from the current DFSMSrmm owner then DFSMSrmm uses the information to update the owner information for the volume. If the owner information is not valid, DFSMSrmm sets a return code and issues message EDG8195I. During cartridge entry processing, or when owner information does not exist, DFSMSrmm returns owner information in the parameter list.	For volumes to be added as scratch volumes, there should be no owner information. If the first 8 characters of owner information are not null or blank, DFSMSrmm set a null owner name and sets CBRUXCUA return code to UXCCHG. During cartridge entry processing, or if the input owner information was not provided, DFSMSrmm returns the first 8 bytes of owner information.

Table 34. Processing for the Change Use Attribute, Cartridge Entry and Cartridge Eject Parameter List (continued)

Variable	Field Name	Input	Output
Volume record creation date	UXCCREAT UXECREAT UXJCREAT	Not used	Input only
Volume serial	UXCVOLSR UXEVOLSR UXJVOLSR	This value is used to identify the volume to be processed.	Input only
Write protection status	UXCWPROT UXEWPROT UXJWPROT	Not used	Not used

Change Use Attribute Specific Processing

Table 35 describes change use attribute processing.

Table 35. Processing for the Change Use Attribute Parameter List

Variable	Field Name	Input	Output
Current use attribute	UXCCUSEA	Not used	Input only

Cartridge Entry Specific Processing

Table 36 describes cartridge entry processing.

Table 36. Processing for the Cartridge Entry Parameter List

Variable	Field Name	Input	Output
		There are no cartridge entry specific parameter list fields for which DFSMSrmm does processing.	

Cartridge Eject Specific Processing

Table 37 describes cartridge eject processing.

Table 37. Processing for the Cartridge Eject Parameter List

Variable	Field Name	Input	Output
Volume serial	UXJVDISP	This is a volume record disposition which can be: **K for KEEP** Specifies that the volume record should be kept **P for PURGE** Specifies that the volume record should be purged	DFSMSrmm processing depends on the DFSMSrmm EDGRMMxx parmlib OPTION SMSTAPE(PURGE) operand value specified. If ASIS is specified, DFSMSrmm does not change the output field. If YES, we set to P, if NO we set to K.

Volume-Not-In-Library Specific Processing

The DFSMSrmm EDGLCSUX programming interface returns information for a volume, indicating whether the volume is a logical exported volume and the stacked volume on which it is exported. The sample CBRUXVNL exit is updated to call EDGLCSUX to issue a WTOR for all logical volumes. EDGLCSUX issues message EDG8123D to provide information about the stacked volume.

Table 38 describes volume-not-in-library processing.

Table 38. Processing for the Volume-Not-In-Library Parameter List

Variable	Field Name	Input	Output
Library name	UXNLIB	Contains library name into which the volume should be entered, or blanks. DFSMSrmm adds this library name to message EDG8121D so that the operator can enter the volume into the correct library. A value of blanks is displayed as the value **ANY**.	Input only

With volume-not-in-library processing, you can set flags to enable DFSMSrmm to retrieve volume information and to request that DFSMSrmm uses WTOR processing to communicate with the operator. When EDGLCSUX is called from CBRUXVNL, you can set either LCSUP_VNL flag or LCSUP_ACTVNL flag. You must first specify LCSUP_VNL to enable DFSMSrmm to retrieve volume information. Your second request, which is optional, must specify LCSUP_ACTVNL to request that DFSMSrmm uses WTOR processing to communicate with the operator. Between calls to EDGLCSUX from CBRUXVNL, do not modify any of the data returned by DFSMSrmm because the information is used in the messages sent to the operator.

When DFSMSrmm is first called from CBRUXVNL, DFSMSrmm retrieves information about the subject volume from the DFSMSrmm control data set if the volume is defined to DFSMSrmm. DFSMSrmm passes back volume location, movement, and status information in output fields in the EDGLCSUP parameter list. When DFSMSrmm is called a second time, and you have set the LCSUP_ACTVNL flag, DFSMSrmm issues the following messages which prompt the operator to enter the volume into the identified library. If the LCSUP_ACTVNL flag is not set, DFSMSrmm does not issue these WTORs for the operator to move the volume into the system-managed library.

- **EDG8121D** VOLUME *req_volser* RACK *rack_number* LOCATION *loc_name* BIN *bin_number* HOME LOCATION *home* - ENTER VOLUME INTO LIBRARY *lib_name* AND REPLY ″RETRY″, OTHERWISE REPLY ″CANCEL″ OR ″CONTINUE″
- **EDG8122D** VOLUME *req_volser* RACK *rack_number* LOCATION *loc_name* BIN *bin_number* HOME LOCATION *home* - ENTER VOLUME INTO LIBRARY *lib_name* AND REPLY ″RETRY″, OTHERWISE REPLY ″CANCEL″

For volumes residing in a IBM TotalStorage Peer-to-Peer Virtual Tape Server (PtP VTS), DFSMSrmm issues the message EDG8123D.

- **EDG8123D** LOGICAL VOLUME *req_volser* EXPORTED IN STACKED VOLUME *stack_volser* LOCATION *loc_name* SHELF *shelf_number* HOME LOCATION *home* - IMPORT VOLUME TO LIBRARY *lib_name* AND REPLY ″RETRY″, OTHERWISE REPLY ″CANCEL″

Based on the reply, the return code for OAM, is set in field LCSUP_LCSRC ready to be passed back direct to OAM by the CBRUXVNL exit. The LCSUP_LCSRC field returns the following values.

Reply	**Return Code**
CONTINUE	0 - UXNNORML
RETRY	4 - UXNRETRY
CANCEL	8 - UXNFAIL in PROTECT mode, UXNNORML in other modes.

Table 39 defines the OAM return codes generated when the EDGLCSUX return code is not zero. The return code is the value in register 15 on return from EDGLCSUX.

Table 39. DFSMSrmm OAM Return Codes from EDGLCSUX Register 15

Return Code	Description	Related Message Number	OAM Return Code
LCSUP_RC_OK R0 is LCSUP_RS_OK	Processing successful	none	See reason code in LCSUP_LCSRS field
LCSUP_RC_OK R0 is LCSUP_RS _NOACTION	Subsystem not required by installation	none	UX*x*NOCHG[1],[2] UXNNORMAL[3]
LCSUP_RC_OK R0 is LCSUP_RS_DONT	DFSMSrmm not licensed by installation	none	UX*x*DONT [1]
LCSUP_RC_SSNA	Subsystem not available	EDG8102D EDG8108D EDG8110D	UX*x*FAIL [1]
LCSUP_RC _LERR	Logic error in DFSMSrmm processing	EDG8105I EDG8106I EDG8107I	UX*x*FAIL [1]
LCSUP_RC_ENV	Environment error detected	EDG8101I EDG8111I EDG8112I	UXxFAIL [1]

Notes:

1. *x* can be:
   - C for change use return codes
   - E for volume entry
   - J for volume eject return codes
   - N for not in volume return codes
2. UX*x*NOCHG is issued for change use, volume entry and volume eject
3. UXNNORML is issued for not in library only

Environment

EDGLCSUX executes in the same environment as the OAM exits, but in AMODE(31) RMODE(ANY).

Processing Fetch and Mount Messages: EDGMSGEX

The system uses DFSMSdfp MSGDISP to update the display screens on tape drives. DFSMSrmm supplies Assembler source code for IGXMSGEX, the MSGDISP installation exit, that you can modify to include your own functions or you can enable to run with another product's supplied exit code.

IGXMSGEX calls EDGMSGEX, the program that supports the MSGDISP interface. EDGMSGEX is a callable program interface to DFSMSrmm that you can use from the MSGDISP exit. You can run this exit in parallel with another copy of the exit. See "Setting Up Parallel Processing" on page 216 for more information. The following sections describe how to use the interface.

Input

The invocation environment must be identical to that provided at entry to the MSGDISP exit IGXMSGEX.

Output

The output is MSGDISP parameters that are updated as appropriate. Register 15 is always zero.

Processing

For nonspecific mount requests, where a specific scratch pool is required, DFSMSrmm updates the message text with the pool details. If you use the EDGUX100 installation exit to select a specific pool, DFSMSrmm updates the message text and checks the cartridge loader status. If you requested the loader be disabled for a specific pool, DFSMSrmm sets bit 7 of the format control byte to zero.

For specific mount requests, DFSMSrmm checks the volume and uses the volume serial number to determine the rack number and updates the message text with the rack number. If you use the installation exit EDGUX100 to specify a rack number or an external volume serial number for a volume that DFSMSrmm should ignore, DFSMSrmm uses the value you specify to update the message with the rack number.

Environment

EDGMSGEX runs only from the MSGDISP exit IGXMSGEX. It runs in AMODE(31) RMODE(31).

Processing JES3 Messages: EDG3X71

The DFSMSrmm-supplied IATUX71 USERMOD calls EDG3X71 which is the program that supports JES3 message processing. EDG3X71 is a callable program interface to DFSMSrmm that you can use from the IATUX71 exit. The following sections describe how to use the interface.

Input

The invocation environment must be identical to that provided at entry to the exit IATUX71.

Output

R15 on exit is determined by DFSMSrmm processing. The possible values for the return code are the same codes described for IATUX71 return code. The intention is that you pass the EDG3X71 return code to JES3 as the IATUX71 return code.

0	DFSMSrmm sets this return code when the volume is not managed by DFSMSrmm or when DFSMSrmm is not currently in use.
4	DFSMSrmm sets this return code when the volume serial number is to be replaced by the rack number or pool prefix. MSGDISP text is also provided. DFSMSrmm sets this return code when you have coded the MNTMSG definitions to specify that the rack number is to be placed within the JES3 message text.
8	DFSMSrmm sets this return code when the rack number or pool prefix is to be added to the end of the JES3 message. MSGDISP text is also provided. DFSMSrmm sets this return code when you have: • Coded MNTMSG definitions so that DFSMSrmm appends the rack number after the JES3 message text. • Have not coded a MNTMSG definition for this message.
16	DFSMSrmm never sets this return code.

Processing

EDG3X71 can only be called from the JES3 exit IATUX71. If EDG3X71 is called when DFSMSrmm is not started or in use, EDG3X71 sets return code 0 for IATUX71. EDG3X71 determines whether information should be inserted or appended for JES3 messages and determines the value to be used for MSGDISP tape drive display processing.

To check that the IATUX71 exit modification is installed correctly, you can run some batch jobs that use both specific and nonspecific tape requests. You should check that messages displayed on the consoles, in SYSLOG, in DLOG and MLOG, and in JESMSG contains the expected DFSMSrmm updates.

Environment

EDG3X71 runs under a JES3 subtask in the JES3 address space on the JES3 global. EDG3X71 must be link edited in an APF-authorized library. It runs in AMODE(31) RMODE(31).

Setting Up Parallel Processing

When you are converting to DFSMSrmm, you might want to run two versions of your installation exits at the same time. The CBRUXENT exit, the CBRUXEJC exit, and the IGXMSGEX exit provide support that enables your existing exits to control processing before DFSMSrmm gains control. You can set up parallel processing using an SMP/E USERMOD or outside of SMP/E.

The CBRUXENT exit and the CBRUXEJC exit samples shipped with DFSMSrmm can ensure that the DFSMSrmm processing is run after your existing exits are run and that DFSMSrmm records any information modified by your existing exits. The tape management decisions made by your existing exits are used instead of any tape management decisions that DFSMSrmm might make.

The IGXMSGEX exit includes support that enables your existing exit to perform processing before DFSMSrmm gains control. You use the IGXMSGEX exit to update tape drive displays. You only need either one system or the other to be updating the drive display. As long as you use the same scratch pool names on both systems, either your existing exit or DFSMSrmm can be used to update the drive display.

Setting Up Parallel Processing Using SMP/E

Create and install an SMP/E USERMOD as shown in Figure 75 to set up parallel running capability.

```
//RMMSTUFF  JOB  ,'SLIP IT IN',MSGCLASS=H,MSGLEVEL=(1,1)
//STEP1     EXEC SMPEMVS,REGION=6120K
//SMPCNTL   DD   *
   SET BDY(GLOBAL) .
   RECEIVE .
/*
//SMPPTFIN  DD   DATA,DLM=##
++USERMOD (VMRMM03) REWORK(2000287) .
++VER (Z038) FMID(HDZ11D0) PRE(UWxxxxx) /*
     Change FMID as needed.
     Edit PRE as needed or use the BYPASS(PRE)
     operand on the APPLY command       */ .
++SRC(CBRUXENT) TXLIB(AEDGSRC1) DISTLIB(AEDGSRC1) /*
     AEDGSRC1 points to latest source level
     assumption is that PTFs are already accepted
     if not, use SMPSTS as the TXLIB            */ .
++SRC(CBRUXEJC) TXLIB(AEDGSRC1) DISTLIB(AEDGSRC1) .
++SRC(IGXMSGEX) TXLIB(AEDGSRC1) DISTLIB(AEDGSRC1) .
##
//STEP2     EXEC SMPEMVS,REGION=6120K
//SMPCNTL   DD   *
   SET BDY(GLOBAL) .
   UCLIN .
   ADD UTILITY(ASMSYSP) NAME(ASMA90)
   PARM(XREF(FULL),NOOBJECT,DECK,RENT,SYSPARM(YES)) .
   ADD OPTIONS(ASMSYSP)
             ASM(ASMSYSP)  .
   ENDUCL .
/*
//STEP3     EXEC SMPEMVS,REGION=6120K
//SMPCNTL   DD   *
   SET BDY(TGT1) OPTIONS(ASMSYSP) .
   APPLY SELECT(VMRMM03) .
/*
```

Figure 75. Sample USERMOD for Setting Up Running Exits in Parallel

To set up parallel processing using SMP/E:

1. Copy the DFSMSrmm-supplied IGXMSGEX exit to a source library and modify the exit if necessary
2. Copy the source of the IGXMSGEX exit from your existing tape management system after the end of the DFSMSrmm-supplied source and change the name of the CSECT to ANOMSGEX or any other meaningful name.
3. Install the SMP/E USERMOD. You install both the DFSMSrmm-supplied exit and your exit from your existing tape management system.
4. To implement changes to the CBRUXENT exit and the CBRUXEJC exit, you do not need to IPL. You can copy the updated load modules into LINKLIB and refresh LLA.

5. To implement changes to the IGXMSGEX exit, you must copy IGX00030 into LPALIB and re-IPL with CLPA.

   Recommendation: Because you might not want to IPL at the time you switch DFSMSrmm to production, careful planning of the change to IGX00030 is required. Replace the parallel running IGXMSGEX exit while you continue running in parallel when you are sure that DFSMSrmm is providing the expected scratch pooling support. DFSMSrmm should be running in warning mode and should not issue any error messages about volumes from incorrect scratch pools.
6. When you no longer need to run both exits, remove the USERMOD and install the standard load modules again.

To edit the source code to set your own selected alternate exit load module names, copy the latest DFSMSrmm source code from either SMPSTS or AEDGSRC1, and alter the TXTLIB to point to your source library that contains your modified source code.

Setting Up Parallel Processing Outside of SMP/E

To set up parallel processing for the OAM tape exits CBRUXENT and CBRUXEJC, perform the following steps:

1. Rename the existing exit load module names as follows:
   - CBRUXENT to ANOUXENT
   - CBRUXEJC to ANOUXEJC
2. Assemble the DFSMSrmm-supplied exits with PARM='SYSPARM(YES)'.
3. Link the DFSMSrmm-supplied exits.

To set up the parallel processing for the message display exit IGXMSGEX, follow these steps:

1. Assemble the DFSMSrmm-supplied exits with PARM='SYSPARM(YES)'.
2. Link the DFSMSrmm-supplied exit together with the renamed IGXMSGEX exit that is provided by the existing tape management system as shown in Figure 76 on page 219. The first step of the link-edit is to extract the IGXMSGEX exit that is provided by the existing tape management system from IGX00030 and rename it to ANOMSGEX. The second step of the link-edit is to update the existing load module with the DFSMSrmm-supplied IGXMSGEX exit and to add in the ANOMSGEX section.

```
//LINK1    EXEC PGM=IEWL,REGION=2M,
   // PARM='LET,LIST,XREF,NCAL,RENT,NCAL'
   //LPALIB   DD DISP=SHR,DSN=SYS1.LPALIB
   //*
   //SYSLMOD  DD DISP=SHR,DSN=DFRMM1.NEW.LOAD
   //AEDGMOD1 DD DISP=SHR,DSN=SYS1.AEDGMOD1
   //*
   //SYSPRINT DD SYSOUT=*
   //SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
   //SYSLIN   DD *
    REPLACE EDGMSGEX
    REPLACE IGXMSG01
    REPLACE IGX00030
    RENAME  IGXMSGEX,ANOMSGEX
    CHANGE  IGXMSGEX(ANOMSGEX)
    INCLUDE LPALIB(IGX00030)
    NAME    ANOMSGEX(R) RC=4
    INCLUDE AEDGMOD1(IGXMSGEX)
    INCLUDE LPALIB(IGX00030)
    INCLUDE SYSLMOD(ANOMSGEX)
    ORDER   IGX00030,IGXMSG01
    ORDER   IGXMSGEX,EDGMSGEX
    MODE    RMODE(ANY)
    ENTRY   IGX00030
    NAME    IGX00030(R) RC=4
```

Figure 76. Sample JCL for Setting Up Running Exits in Parallel

You can also edit the DFSMSrmm-supplied exits as shown in Figure 77. The example shows the CBRUXENT exit to set the &ANOEXIT variable with the new name of the existing exit and the &PARALLEL variable to YES. Then you can assemble and link-edit the updated DFSMSrmm-supplied exits.

```
CBRUXENT TITLE 'DFSMSRMM CBRUXENT SAMPLE USER EXIT'
&ANOEXIT  SETC 'ANOUXENT'    Replace ANOUXENT with required name
&PARALLEL    SETC '&SYSPARM'    Replace &SYSPARM with YES if req'd
             AIF  ('&PARALLEL   ' EQ '').SETNO
             AIF  ('&PARALLEL   ' EQ 'YES').SETYES
 .SETNO    ANOP
 &PARALLEL    SETC 'NO'
 .SETYES   ANOP
```

Figure 77. Sample JCL for Setting Up Running Exits in Parallel

Chapter 11. Using DFSMSrmm Installation Exits

DFSMSrmm provides installation exits EDGUX100 and EDGUX200. For usage information, see “Using the DFSMSrmm EDGUX100 Installation Exit” and “Using the DFSMSrmm EDGUX200 Installation Exit” on page 248.

Using the DFSMSrmm EDGUX100 Installation Exit

Use the DFSMSrmm EDGUX100 installation exit to perform the following tasks:

- Plan for scratch pools as described in “Planning to Manage Scratch Pools with EDGUX100”
- Perform pooling management as described in “Managing Scratch Pools” on page 223 to:
  - Manage scratch pools based on job name and data set name.
  - Select a pool to use for a nonspecific tape volume request.
  - Select a specific pool to use for a nonspecific tape volume request and request that the tape drive cartridge loader is disabled.
- Ignore foreign or duplicate volumes as described in “Using EDGUX100 to Ignore Duplicate or Undefined Volume Serial Numbers” on page 225 and optionally provide an external volume serial number for use in messages intercepted and updated by DFSMSrmm.
- Use JCL special expiration dates to manage volumes by setting vital record specification management values to retain data sets as described in “Using Vital Record Specification Management Values to Retain Tape Volumes” on page 228.
- Pass pooling decisions to pre-ACS processing so that they can be used in ACS routines to assign storage group and management class as described in “Using the EDGUX100 Installation Exit from Pre-ACS Processing” on page 231.
- Set any expiration date, including a zero value, for a data set as described in “Assigning Expiration Dates” on page 237.
- Obtain information that is described in “Creating Sticky Labels” on page 231 to modify input from DFSMSrmm disposition processing.
- Request the recording of only the first file on a multifile volume as described in “Controlling Tape Volume Data Set Recording” on page 233.
- Change the location for a volume as described in “Changing Location Information with EDGUX100” on page 234.
- Support the use of the storage group name for pooling for volumes that reside in manual tape libraries as described in “Using Storage Group for Manual Tape Library Pooling” on page 241.

Planning to Manage Scratch Pools with EDGUX100

You define pools to DFSMSrmm in parmlib that the EDGUX100 installation exit selects for new tape data sets that are to be created on nonspecific volumes. During OPEN processing DFSMSrmm uses the pool that you specify to validate that a scratch volume has been mounted from the selected pool. You can define each pool with different attributes and each pool can either be a scratch pool or a rack pool. You can use DFSMSrmm ACS storage group support instead of the EDGUX100 exit selected pooling. Refer to “Using SMS Tape Storage Groups for DFSMSrmm Scratch Pooling” on page 71 for information about using the DFSMSrmm storage group support.

If you plan to use EDGUX100 to select volumes from specific scratch pools with DFSMSrmm, you need to consider the following:

Selecting Pool Types

The pools that you select using the EDGUX100 installation exit must match to a VLPOOL rack number prefix defined in the DFSMSrmm parmlib member as described in “Defining Pools: VLPOOL” on page 162.

You can use EDGUX100 to set up a specific pool for a particular user of your systems. You can define a scratch or rack pool and can define any type of volume in that pool. You do not need to have two pools for a user; one for scratch volumes and one for private volumes. For example, define some volumes that will cycle between scratch and master status throughout their life, and you could also define specific, private volumes that are never to be used as scratch.

You use scratch pools with DFSMSrmm system based scratch pooling. You can define scratch volumes in rack pools and use the RMM GETVOLUME subcommand to claim them or assign them to a user. You can also use EDGUX100 to use rack pools or scratch pools for nonspecific tape output requests.

When you define pools for use with EDGUX100, you also might need to prevent the pools from being used at the wrong times. For example, if your EDGUX100 exit does not make a pool selection you might want to ensure that DFSMSrmm accepts only scratch volumes which are not in your specific pools.

Here are three ways you can prevent pools from being used at the wrong time:

1. Ensure that your EDGUX100 exit always provides a specific pool. Specify a default pool prefix of ’*’ in the exit to use the DFSMSrmm default pool for all requests that have no specific pool identified.
2. Use only rack pools for your EDGUX100 scratch pools. You can also have DFSMSrmm scratch pools or a default scratch pool for use with those requests where your exit does not provide a specific pool.
3. Define all the specific scratch pools that use a SYSID value that does not match any of your systems. This ensures that, if the exit does not supply a specific scratch pool and DFSMSrmm uses its own pooling, all the specific pools would be ignored.

Controlling Pool Selection

The sample EDGUX100 installation exit provides a working solution for application type scratch pools based on job names and data set names.

Tip: The exit does not validate the field contents; follow the job naming conventions or your entry will never match real job names.

The exit does not validate the field contents, so be sure to follow the job naming conventions or your entry will never match real job names.

Managing Tape Drive Availability

You can use the EDGUX100 exit to request DFSMSrmm to disable the tape drive cartridge loader to prevent specific scratch pool requests from emptying the loaders through volume rejection. Alternatively, you can direct requests to the correct drives or run the loaders in a mode that prevents them being automatically indexed when a mount is received.

Defining Operator Messages for Tape Drive Display

To use ACS routines or exit selected scratch pooling, you must define messages IEC501A, IEF233A, IAT5110, and IAT5210 using the DFSMSrmm MNTMSG commands in parmlib as described in “Defining Mount and Fetch Messages: MNTMSG” on page 132. These messages provide the pool identifiers to the operator, and the job name and data set name information to the EDGUX100 exit.

The sample EDGUX100 exit obtains the job name and data set name for a nonspecific volume request from the text of the message. The EDGUX100 exit includes the data set name only if the MVS MONITOR DSNAME option is in effect. If you plan to use the sample EDGUX100, you must activate the MVS MONITOR DSNAME option through an operator command. See *z/OS MVS System Commands* for more information about the MVS MONITOR command.

Managing Scratch Pools

You define pools to DFSMSrmm in parmlib that the EDGUX100 installation exit can select for new tape data sets. The new tape data sets are to be created on nonspecific volumes. During OPEN processing DFSMSrmm uses the pool that you specify to validate that a scratch volume has been mounted from the selected pool. Each pool can be defined with different attributes, and can either be a scratch pool or a rack pool.

You can specify whether you want DFSMSrmm to disable the autoloader when the EDGUX100 installation exit selects a scratch pool. By using the exit to disable the loader, you can prevent the loader from being emptied when it contains volumes from another pool. If you are using multiple scratch pools and a tape drive is allocated for a request that requires a pool that has not been pre-loaded, DFSMSrmm rejects pre-loaded volumes from other pools.

You can pass the EDGUX100 exit selected scratch pool prefix to your ACS routines for making allocation decisions and for assigning storage group names for system-managed tape data sets. You can do this using the pre-ACS call to the EDGUX100 installation exit. See “Using the EDGUX100 Installation Exit from Pre-ACS Processing” on page 231 for information about using the EDGUX100 installation exit from pre-ACS processing. You can replace the EDGUX100 processing by using the DFSMSrmm calls to SMS ACS routines requesting a storage group name. Refer to “Using SMS Tape Storage Groups for DFSMSrmm Scratch Pooling” on page 71 for information about DFSMSrmm ACS support for storage group.

Step 1: Define the Pools

For each pool, you need to decide whether this pool is to be used only for specifically identified purposes or whether it can be also selected based on system scratch pools. The easiest way to prevent a pool from being used as a system–based scratch pool is to use only rack pools for your exit driven scratch pooling. If you do not do this, or you do not use one of the other methods described in “Planning to Manage Scratch Pools with EDGUX100” on page 221, you must use the EDGUX100 installation exit to always set a scratch pool to prevent an incorrect scratch pool from being accepted by DFSMSrmm.

DFSMSrmm identifies pools by using a pool prefix which is a one-to-five character name followed by an asterisk. See “Defining Pools: VLPOOL” on page 162 for more information.

For example, if you want to define a pool with a prefix of ABC* for use by a specific application, define the pool to DFSMSrmm in parmlib using one of the statements shown in Figure 78.

```
VLPOOL PREFIX(ABC*) TYPE(R) RACF(Y) EXPDTCHECK(N) -
                 DESCRIPTION('Application XyZ')
VLPOOL PREFIX(ABC*) TYPE(S) RACF(Y) EXPDTCHECK(N) -
     SYSID(NONE) DESCRIPTION('Application XyZ')
```

Figure 78. Defining Pools for a Specific Application

The parameter SYSID(NONE) coded in the second statement in the example, prevents the pool from being used by the DFSMSrmm system scratch pooling as long as no DFSMSrmm system is defined with SYSID(NONE).

For each pool, decide if you will run with scratch volumes from this pool loaded in a cartridge loader. For those pools that are not loaded, consider using the EDGUX100 installation exit to disable the cartridge loader whenever those pools are requested.

Step 2: Tailor the Sample EDGUX100 Installation Exit

Update the sample EDGUX100 installation exit based on the pools you define using the VLPOOL command in parmlib. The supplied sample EDGUX100 installation exit supports pooling based on job name and data set name. Modify the supplied exit if you set up pools that are based on other criteria.

To use the exit you must follow these steps:

1. Copy the sample EDGUX100 installation exit. If you are already using the EDGUX100 installation exit to support another function, start using your latest copy of the exit source, or merge in any changes that you have previously made to the exit.
2. Update the exit, bearing in mind the following:
   - Only perform your processing when the PL100_CAN_POOL bit is set to B'1'.
   - Your decisions on pooling must be possible whether the exit is called for MNTMSG processing, pre-ACS processing, or for OPEN processing. Your pooling decisions must work when a WTO address is provided during MNTMSG processing, an ACERO is provided during pre-ACS processing, or a JFCB address is provided during OPEN processing.
   - Based on your pooling decisions, optionally select a pool for the current request and set the PL100_POOL field. You must also set the PL100_SET_POOL flag to B'1' for DFSMSrmm to use the PL100_POOL field. You can also optionally set the PL100_SET_ACLOFF flag to B'1' to request that the tape drive cartridge loader is not indexed for the request. See “Supplying a Scratch Pool Name” on page 239 and “Using the System Name to Select a Scratch Pool” on page 240 for additional information.

Step 3: Activate the EDGUX100 Installation Exit

See “Installing the EDGUX100 Routine” on page 242 for information about building an SMP/E USERMOD to apply the updated source code for EDGUX100 so that it supersedes any old EDGUX100 USERMODs. Include the necessary JCLIN statements as shown in Figure 86 on page 242.

Step 4: Define MNTMSG Parmlib Options

The sample MNTMSG parmlib options includes all the messages that are required to successfully run the scratch pooling function provided by the sample exit.

However, refer to “Operator Messages and Tape Drive Displays” on page 66 for information about the MNTMSG requirements so that you can ensure your system has the required options already in use.

Step 5: Set Up Cartridge Loaders

Set cartridge loaders that are used with DFSMSrmm to system mode or manual mode.

Step 6: Updating JES3 Code (Optional)

This step is only required if you have JES3 managed tape devices and do not use deferred tape mounting. With deferred tape mounting, either allocation processing issues or OPEN processing issues the mount requests and the tape drive displays are correct.

JES3 updates the drive display before it issues the IAT5210 mount message, so the drive display might not contain the pool that you select in your exit.

DFSMSrmm provides three methods for updating the IAT5210 message and the tape drive display. Using the DFSMSrmm supplied USERMOD EDGUX71 to IATUX71 is the preferred method for updating the IAT5210 message with the correct exit selected pool. The exit selected pool is used to update the tape drive displays when selection is based on the IAT5210 message. See Chapter 13, “Running DFSMSrmm with JES3,” on page 271 for information about installing and using DFSMSrmm-supplied JES3 USERMODs.

Step 7: Activate MONITOR DSNAME

The EDGUX100 installation exit depends on the system mount messages containing data set name information. If you plan to use data set name based pooling, issue the MVS MONITOR DSNAME command.

For testing purposes, issue this command at an operator terminal. However, for production implementation, ensure that the command is always issued by including it as a command in the COMMNDxx member of the parmlib. Any other method that results in the monitor option being active before any tape requests are issued is also acceptable.

When a non-specific volume request is received, use the exit to determine the correct pool to be used.

Using EDGUX100 to Ignore Duplicate or Undefined Volume Serial Numbers

When DFSMSrmm ignores a volume, DFSMSrmm does not perform these management functions for the volume:
- Record information about the volume in the DFSMSrmm control data set.
- Validate that the correct volume has been mounted.

For volumes that are defined to DFSMSrmm and that are ignored, DFSMSrmm performs inventory management based on previously recorded information.

DFSMSrmm ignores volumes that are not defined to DFSMSrmm for specific mount requests. For nonspecific requests, DFSMSrmm ensures that a DFSMSrmm-managed scratch volume is mounted in response to the request.

To request that DFSMSrmm ignore a volume, perform the following steps:
1. Tailor the EDGUX100 DFSMSrmm installation exit to use undefined volumes or duplicate volumes. When you use the EDGUX100 installation exit, DFSMSrmm

calls the exit each time a volume is opened. The sample installation exit checks the JCL–specified EXPDT value for the special date 98000 or for the ACCODE value *x*CANORES. If the 98000 special date or the ACCODE value *x*CANORES is found, the EDGUX100 exit uses the installation exit parameter list to request that DFSMSrmm ignore the volume.

2. Activate the exit.
3. Define a RACF FACILITY class entity, STGADMIN.EDG.IGNORE.TAPE.*volser*. If you want to distinguish between volumes managed by DFSMSrmm and volumes not managed by DFSMSrmm, use STGADMIN.EDG.IGNORE.TAPE.RMM.*volser* and STGADMIN.EDG.IGNORE.TAPE.NORMM.*volser* to check user authorization.
4. Authorize users to the STGADMIN.EDG.IGNORE.TAPE.*volser*, STGADMIN.EDG.IGNORE.TAPE.RMM.*volser*, and STGADMIN.EDG.IGNORE.TAPE.NORMM.*volser* resources.

Use the DFSMSrmm EDGUX100 installation exit to have DFSMSrmm ignore a volume when any of the following conditions exist:

- You specify the parmlib REJECT ANYUSE(*) command to reject an undefined volume.
- You use a volume is in an automated tape library and it is automatically defined to DFSMSrmm.
- You want to ignore a volume used for a non-specific output request.

If these conditions are not present, DFSMSrmm automatically ignores all volumes that are not defined to it without using the DFSMSrmm sample installation exit.

Before DFSMSrmm ignores the volume, it ensures that the user who opened the volume is authorized to request this function. DFSMSrmm uses a security resource in RACF FACILITY class, and issues a SAF RACROUTE TYPE=AUTH request with one of the following entity names.

- STGADMIN.EDG.IGNORE.TAPE.*volser*
- STGADMIN.EDG.IGNORE.TAPE.RMM.*volser*
- STGADMIN.EDG.IGNORE.TAPE.NORMM.*volser*

where *volser* is the volume serial number of the mounted volume or requested tape volume.

STGADMIN.EDG.IGNORE.TAPE.RMM.*volser* volumes are volumes that meet one of these criteria.

- The volume serial number is defined in the control data set and there is no HDR1 tape label for the volume.
- The volume serial number is defined in the control data set and no labels are used (NL, NSL, or BLP with an NL tape).
- The 17 characters read from the HDR1 label of the mounted volume match the last 17 characters of the data set name in the control data set.

STGADMIN.EDG.IGNORE.TAPE.NORMM.*volser* volumes are volume that meet one of these criteria.

- The volume serial number is not defined in the control data set.
- The 17 characters read from the HDR1 label of the mounted volume do not match the last 17 characters of the data set name in the control data set.

Example: If you want DFSMSrmm to ignore volume A00001 that is defined to DFSMSrmm, define the STGADMIN.EDG.IGNORE.TAPE.*.* UAC(NONE). DFSMSrmm then checks the second profile:

```
STGADMIN.EDG.IGNORE.TAPE.A00001
STGADMIN.EDG.IGNORE.TAPE.RMM.A00001
```

Step 1: Tailor the DFSMSrmm EDGUX100 Installation Exit

Based on the decisions you have made about how to manage duplicate or undefined volume serial numbers, tailor the supplied sample EDGUX100 installation exit as follows:

1. Make a copy of the sample installation exit to use as a base for your exit.
2. Update the exit. Only perform your processing when the PL100_CAN_IGNORE bit is set to B'1'. Decide whether to support the pre-ACS call to obtain a VRS management value or scratch pool prefix for use as the MSPOLICY and MSPOOL variables in ACS processing. You must ignore the call if the volume is to be ignored. During the pre-ACS call, an ACERO is passed instead of a JFCB. See *z/OS DFSMS Installation Exits* for information about the ACERO. See “Assigning Expiration Dates” on page 237 for additional processing information.
   - Use the value in the JFCB expiration date field, JFCBXPDT, to determine if a special date has been specified.You could also use the PL100_ACCODE field to check if the special ACCODE value has been supplied.
   - Set one of the following flags to B'1' so that the volume is ignored if the special date 98000 or the special ACCODE=xCANORES value has been specified. The flags are
     – PL100_SET_IGNORE
     – PL100_SET_IGNORE_MOUNTED
     – PL100_SET_IGNORE_REQUESTED
   - Update the WTO messages with a rack number. Place the rack number value in the PL100_RACKNO field
   - Clear the expiration date in the JFCB copy that DFSMSrmm uses to prevent DFSMSrmm from using the 98000 date as a volume expiration date. You do not need to update the JFCB. The information is not passed on to DFSMSrmm. Because it is a copy of the JFCB used solely by DFSMSrmm, no other component can make use of your changes. However, because the EDGUX100 installation exit is called for system-managed volumes, but nonspecific system-managed volumes cannot be ignored, the DFSMSrmm sample exit clears the expiration date.

Step 2: Activate the EDGUX100 Installation Exit

See “Installing the EDGUX100 Routine” on page 242 for information about building an SMP/E USERMOD to apply the updated source code for EDGUX100 so that it supersedes any old EDGUX100 USERMODs. Include the necessary JCLIN statements as shown in Figure 86 on page 242.

Step 3: Define a RACF FACILITY Class Entity to Check Authorization

Define a RACF FACILITY class entity to protect volumes. DFSMSrmm checks that the user trying to use a duplicate or undefined volume is authorized to request that DFSMSrmm ignore the volume. The RACF FACILITY class entities are:

- STGADMIN.EDG.IGNORE.TAPE.*volser*
- STGADMIN.EDG.IGNORE.TAPE.RMM.*volser*
- STGADMIN.EDG.IGNORE.TAPE.NORMM.*volser*

Only users with the correct authorization to the entities can request that DFSMSrmm ignore volumes.

Define multiple resources, as required, for individual volumes or groups of volumes by using RACF generic resource names. GLOBALAUDIT(SUCCESS) ensures that RACF maintains an audit trail of all successful uses of this facility.

Example:

```
RDEFINE FACILITY STGADMIN.EDG.IGNORE.TAPE.X* UACC(NONE)
RALTER FACILITY STGADMIN.EDG.IGNORE.TAPE.X* GLOBALAUDIT(SUCCESS)
RDEFINE FACILITY STGADMIN.EDG.IGNORE.TAPE.810* UACC(NONE)
RALTER FACILITY STGADMIN.EDG.IGNORE.TAPE.810* GLOBALAUDIT(SUCCESS)
RDEFINE FACILITY STGADMIN.EDG.IGNORE.TAPE.RMM.Y* UACC(NONE)
RALTER FACILITY STGADMIN.EDG.IGNORE.TAPE.RMM.Y* GLOBALAUDIT(SUCCESS)
RDEFINE FACILITY STGADMIN.EDG.IGNORE.TAPE.RMM.710* UACC(NONE)
RALTER FACILITY STGADMIN.EDG.IGNORE.TAPE.RMM.710* GLOBALAUDIT(SUCCESS)
RDEFINE FACILITY STGADMIN.EDG.IGNORE.TAPE.NORMM.Z* UACC(NONE)
RALTER FACILITY STGADMIN.EDG.IGNORE.TAPE.NORMM.Z* GLOBALAUDIT(SUCCESS)
RDEFINE FACILITY STGADMIN.EDG.IGNORE.TAPE.NORMM.910* UACC(NONE)
RALTER FACILITY STGADMIN.EDG.IGNORE.TAPE.NORMM.910* GLOBALAUDIT(SUCCESS)
```

Step 4: Authorize Users

Permit those users authorized in your installation to the appropriate FACILITY CLASS resources.

Example:

```
PE STGADMIN.EDG.IGNORE.TAPE.X* CLASS(FACILITY) ID(user1) ACCESS(level)
PE STGADMIN.EDG.IGNORE.TAPE.810* CLASS(FACILITY) ID(user2) ACCESS(level)
PE  STGADMIN.EDG.IGNORE.TAPE.RMM.Y* CLASS(FACILITY) ID(user1) ACCESS(level)
PE  STGADMIN.EDG.IGNORE.TAPE.RMM.710* CLASS(FACILITY) ID(user2) ACCESS(level)
PE  STGADMIN.EDG.IGNORE.TAPE.NORMM.Z* CLASS(FACILITY) ID(user1) ACCESS(level)
PE  STGADMIN.EDG.IGNORE.TAPE.NORMM.910* CLASS(FACILITY) ID(user2) ACCESS(level)
```

In the example, the values that are specified have the following meaning:

user1, user2
The user ID of the user given access.

level The granted access level; one of READ or UPDATE.

READ access is required for a volume to be opened for input requests. UPDATE access is required for a volume to be opened for output requests.

Using Vital Record Specification Management Values to Retain Tape Volumes

You can use the EDGUX100 installation exit to assign vital record specification management values to new tape data sets. When a volume is opened, DFSMSrmm records the vital record specification management value you assign to new tape data sets. You then define vital record specifications using data set name masks that match the vital record specification management values you define for special dates. These vital record specifications define the management policies for volumes with special dates.

You can use vital record specification management values to be the basis for assigning management class for system-managed tape. DFSMSrmm calls EDGUX100 to obtain a vital record specification management value. During pre-ACS processing, the vital record specification management value is passed to your ACS routine to so that allocation decisions can be made. During OPEN processing for new tape data sets, this call is optional call and is not made for a non-system-managed tape data sets if you have assigned a management class during the RMMVRS ACS call.

Step 1: Define Vital Record Specification Management Values

A vital record specification management value is a single qualifier name that you define in a vital record specification to tell DFSMSrmm how to manage and retain your tape data sets. The vital record specification management value can be up to eight alphanumeric characters, and must begin with an alphabetic character. DFSMSrmm matches a data set to a vital record specification management value during vital records processing when the data set does not match a vital record specification with a data set name mask and optionally a job name.

Use the RMM ADDVRS subcommand with the DSNAME operand or the DFSMSrmm Add Data Set VRS panel in the DFSMSrmm ISPF dialog to define data set vital record specifications. Use the data set name masks that match the vital record specification management values you have defined. See *z/OS DFSMSrmm Guide and Reference* for more information on defining vital record specifications.

For example, you can use the vital record specification management value D99000, for the special date 99000 and define a vital record specification using the data set name, D99000. Figure 79 defines a vital record specification for managing the special date 99000. You could also set the ACCODE=xCACATLG value in the DD statement to manage the special date 99000.

```
RMM ADDVRS DSNAME('D99000') WHILECATALOG
```

Figure 79. Managing Special Date 99000 with Vital Record Management Value

You could define a vital record specification data set name mask that matches multiple vital record specification management values. For example you could define a data set name mask of D9900% as shown in Figure 80 to cover several vital record specification management values. With this data set name mask you could manage special dates in the range 99001 through 99009.

```
RMM ADDVRS DSNAME('D9900%')
```

Figure 80. Specifying Data Set Masks for Vital Record Management Values

Step 2: Tailor the Sample EDGUX100 Installation Exit

Update the sample EDGUX100 installation exit based on vital record specification management values you define and perform these tasks:

1. Copy the sample EDGUX100 installation exit and use the copy as a base for your exit.
2. Update the exit. Only perform your processing when the PL100_CAN_VRS bit is set to B'1'.
   - If an ACERO is passed to the exit, use the ACEROEXP and ACERORTP fields to decided if a special date is specified. If a JFCB is passed to the exit, use the value in the JFCB expiration date field, JFCBXPDT to determine if a special date has been specified. Test the JFCBEXP flag to determine if the user specified an expiration date or a retention period. You can also test to see if the user wants the date used as real expiration date by checking for the existence of the dummy file NOTKEYD8.

     The sample EDGUX100 installation exit includes code for using this method specifying the DD name NOTKEYD8. You can also use the PL100_ACCODE field to check if the special ACCODE value has been supplied. Specifying ACCODE=xCACATLG is the same as using EXPDT=99000.

- Based on the special date like 99000 or the ACCODE value, select the chosen vital record specification management value and use it to set the field PL100_VRS.
- If you want DFSMSrmm to use a true expiration date, rather than the special date from the JCL, update the JFCBXPDT field with the date or with zero. If you update the field with a zero, DFSMSrmm uses the default retention period to calculate an expiration date. See “Assigning Expiration Dates” on page 237 for additional information. If you use in JCL only the ACCODE value, you do not need to alter the JCL expiration date.

The DFSMSrmm EDGUX100 sample provides support for catalog control special dates during pre-ACS and OPEN time and provides the checking necessary for true expiration dates. Modify the sample exit if you wish to support other special dates.

Tip: DFSMSrmm provides a second sample for EDGUX100. This sample is called EDGCVRSX. It is different from the EDGUX100 sample because the special date and pooling function is table driven and you can change the table dynamically. Refer to EDGCMM01 and the IBM Redbook *Converting to Removable Media Manager: A Practical Guide* for documentation on using EDGCVRSX for EDGUX100.

3. Make any changes required for using vital record management values before building the USERMOD.

Step 3: Activate the EDGUX100 Installation Exit

See “Installing the EDGUX100 Routine” on page 242 for information about building an SMP/E USERMOD to apply the updated source code for EDGUX100 so that it supersedes any old EDGUX100 USERMODs. Include the necessary JCLIN statements as shown in Figure 86 on page 242.

To apply vital record specification management values to a new tape data set, you can either specify the special date 99000 in the EXPDT keyword or use corresponding special values in the ACCODE keyword of the DD statement. The EDGUX100 sample installation exit checks the ACCODE value for xCACATLG. If there is a matching value, DFSMSrmm performs the same actions that it performs when it processes 99000. If both ACCODE and EXPDT are used to specify special values, then DFSMSrmm uses the ACCODE value.

Using the ACCODE keyword, you can use the xCAUSER and xCAPERM special values. EDGUX100 treats these two values the same as the never expire date 99365 specified in the EXPDT parameter. A vital record specification management value D99365 is assigned to the data set which ensures that the data set is retained indefinitely.

Step 4: Run DFSMSrmm Inventory Management Vital Record Processing

Run DFSMSrmm inventory management vital record processing to identify which volumes should be retained based on the vital record specification management values you define. DFSMSrmm uses the vital record specification management value assigned to the data set to select the appropriate vital record specification, if inventory management vital record processing does not find a match on the data set mask. See “How Vital Record Processing Works” on page 295 for information about vital record processing.

Using the EDGUX100 Installation Exit from Pre-ACS Processing

You can use ACS routines to automatically determine the target storage group and assign data classes, storage classes, and management classes to SMS-managed data sets. You can use the pre-ACS interface to provide additional information like vault destination or pool information to the ACS routines. You can use the EDGUX100 installation exit for pre-ACS processing. See “Defining System Options: OPTION” on page 134 for information about the DFSMSrmm EDGRMMxx PARMLIB OPTION PREACS operand and the OPTION SMSACS operand that you can use to control how storage group and management class values are assigned. When called during pre-ACS processing, the values selected by the exit are used as input to the ACS routine. Then, when the EDGUX100 installation exit is called at OPEN or when a volume is mounted, the exit can provide the same values it passed during the pre-ACS processing or provide different vital record specification management values and pool values.

The pre-ACS routine passes the address of the ACERO to the EDGUX100 installation exit. The ACERO is mapped by the IGDACERO macro and is the input to the pre-ACS installation exit IGDACSXT. You can use any of the values in the ACERO as input to the EDGUX100 installation exit. The sample EDGUX100 exit uses these values:

- ACEROJOB, which is the job name.
- ACERODSN, which is the data set name.
- ACEROEXP, which is the expiration date. The expiration date is used only if the retention period is not set.
- ACERORTP, which is the retention period.

The values are used to perform PL100_CAN_IGNORE, PL100_CAN_VRS, and PL100_CAN_POOL processing. All other functions are performed only when the EDGUX100 is not called by the pre_ACS routine.

If your installation’s version of the EDGUX100 installation exit provides values for scratch pooling (PL100_POOL) or vital record specification management value (PL100_VRS), DFSMSrmm updates the ACS input values for MSPOOL and MSPOLICY when they have not already been set by the IGDACSXT pre-ACS installation exit.

To perform pre-ACS processing, check the PL100_ACEROPTR field for the address of the ACERO.

Creating Sticky Labels

If you use DFSMSrmm disposition processing you do not need to use EDGUX100 to implement sticky label support because DFSMSrmm provides sticky label support as part of disposition processing. However, you can optionally use EDGUX100 to modify the default DFSMSrmm processing. Refer to Chapter 19, “Setting Up DFSMSrmm Disposition Processing,” on page 389 for more information. If you do not use disposition processing, you can use the EDGUX100 installation exit to implement sticky label support in the following ways.

- By using information that the installation exit provides for a sticky label routine you have written. The steps for tailoring the installation exit are described in “Step 1: Tailor the DFSMSrmm EDGUX100 Installation Exit” on page 232.
- For modifying the default labels produced by DFSMSrmm disposition processing described in “Modifying DFSMSrmm Label Output” on page 232.

- For suppressing the default labels produced by DFSMSrmm disposition processing described in “Modifying DFSMSrmm Label Output.”

Step 1: Tailor the DFSMSrmm EDGUX100 Installation Exit

You can create sticky labels using the DFSMSrmm EDGUX100 installation exit and a sticky label routine you have written. DFSMSrmm calls the EDGUX100 installation exit whenever a tape data set is closed or reaches end of volume when the PL100_ITS_CLOSE option is set. When the exit is called with PL100_ITS_CLOSE set, there is information in the EDGPL100 parameter list that allows you to create sticky labels. See “Installation Exit Mapping Macro: EDGPL100” on page 416.

To change the sticky labels that DFSMSrmm generates for disposition processing, or create your own sticky labels, add your sticky label routine directly to the exit or add a LOAD or LINK statement to call your sticky label routine externally. Figure 81 shows where your routine for creating sticky labels can be called from EDGUX100.

```
JFCBPOOL EQU   *
******************************************************************
* Now we'll check if we come from CLOSE/END OF VOLUME.
* If you want to perform sticky label support, add your code right
* before the statement 'NOTCLOSE  EQU  *'.
******************************************************************
         TM    PL100_VALID,PL100_ITS_CLOSE Is it a call from C/EOV
         BZ    NOTCLOSE            No, continue
         L     R3,PL100_LABINFO    Address Label info block
         USING PL100_LABDS,R3      Addressability
         TM    PL100_OFLAG,PL100_FOUT Is it OUTPUT ?
         BZ    NOTCLOSE            No, continue
*  ...   your code here - continue at NOTCLOSE when finished
*        WTO   'Sticky label routine called . . . . '
*  Refer to PL100_LABDS DSECT in EDGPL100 macro for an explanation
*  of all the fields you may need for sticky label processing.
*
******************************************************************
NOTCLOSE EQU   *
```

Figure 81. Sample EDGUX100 Installation Exit Sticky Label Support

Step 2: Activate the EDGUX100 Installation Exit

See “Installing the EDGUX100 Routine” on page 242 for information about building an SMP/E USERMOD to apply the updated source code for EDGUX100 so that it supersedes any old EDGUX100 USERMODs. Include the necessary JCLIN statements as shown in Figure 86 on page 242.

Modifying DFSMSrmm Label Output

Use the EDGUX100 installation exit to modify the label that DFSMSrmm produces as part of disposition control processing. You can set new values for the number of rows, the length of each row in the label. You cannot exceed the maximum label area size of 2000 characters. You can set an LRECL other than the default LRECL of 80. The LRECL you set must be the same length or less than the number of columns in each row in the label. You can change the values by modifying the label area passed to the EDGUX100 exit based on the PL100_LABDATA field.

You can also suppress the production of labels by setting the PL100_SET_NOLABEL bit to B'1'. DFSMSrmm checks for this bit at CLOSE time. See “Selecting the Method Used for Label Processing” on page 393 for information about modifying labels.

Controlling Tape Volume Data Set Recording

Use the EDGUX100 installation exit to request that DFSMSrmm records information about the first data set only and keeps track of the statistics at the volume level. DFSMSrmm then performs data set name checking on the first data set only. DFSMSrmm does not keep track of the number of data sets on the volume when data set recording is turned off for a volume. DFSMSrmm keeps track of some items like the use count to reflect the number of times the volume is read or written. You must request that DFSMSrmm records only the first data set each time you rewrite the first data set. Otherwise DFSMSrmm resumes recording information about all the data sets on the volume.

You might want to limit DFSMSrmm recording of tape data set information for the following reasons:

- A tape volume is assigned for use by a specific application and the application tracks the tape contents.
- A tape volume contains a large number of files and you do not need to know details of all the files.

If you have applications that create multiple cataloged data sets on tape, and you suppress the DFSMSrmm recording of other than the first data set on the volume, ensure that the data sets other than the first data set are uncataloged. When the recording of data set information is suppressed, DFSMSrmm does not uncatalog these data sets during inventory management processing when the parmlib option UNCATALOG is specified as S or Y. Since DFSMSrmm cannot uncatalog the data sets, another method must be used to uncatalog them. Leaving non-existent data sets cataloged could lead to processing problems later.

Step 1: Tailor the DFSMSrmm EDGUX100 Installation Exit

During the OPEN processing for the first file on a volume, the EDGUX100 installation exit retrieves the data set name, job name and job step program name from system control blocks. The sample EDGUX100 installation exit then scans the EDMTAB table for a match. If there is a match, the EDGUX100 installation exit sets the PL100_SET_IGNORE_FILE2_TON bit to request DFSMSrmm only record data set details for the first file.

To use the EDGUX100 exit for this function, define a table as shown in Figure 82 on page 234. The order in which the table entries are listed is important because the exit scans the table until it finds the first entry where the job name, data set name and program name masks match the current request. You can change the priority of matching by changing the order of the table entries.

```
EDMTAB   DS    0F                  start of table
*                 jobname        data set name
         DC    CL8'*       ',CL44'BACKUP*'
         DC    CL8'ABC*'             program name
*
         DC    CL8'STSGWD* ',CL44'*'
         DC    CL8'A*'               program name
*
         DC    CL8'STSG%%* ',CL44'STSG%%.BACKUP.*'
         DC    CL8'DEF%MAIN'         program name
*
         DC    CL8'STSGDPW ',CL44'DAVE.TOOMUCH.DATA'
         DC    CL8'AB999*'           program name
*
         DC    CL8'EDMEND'         end of table marker
```

Figure 82. Sample Table for Controlling Data Set Recording

The table contains:

jobname
: One-to-eight alphanumeric or national characters including % and *.
    % can be used to ignore a positional character in the job name.
    * can be used to ignore all remaining characters in the job name. A jobname of * means that the entry applies to all jobs.

data set name
: Can be up to forty-four characters, following MVS data set naming conventions, including % and *.
    The character % can be used to ignore a positional character in the data set name.
    The character * can be used to ignore all remaining characters in the data set name. A data set name of * means that the entry applies to all data sets.

    The use of the character * is not the same as in the generic data set names supported by DFSMSrmm for vital records specifications and search data set masks. Here the * works like the characters *.*might in a generic data set name mask.

Program name
: A value up to eight alphanumeric character including % and *.
    % can be used to ignore a positional character in the program name.
    * can be used to ignore all remaining characters in the program name. A program name of * means that the entry applies to all programs.

Step 2: Activate the EDGUX100 Installation Exit

See “Installing the EDGUX100 Routine” on page 242 for information about building an SMP/E USERMOD to apply the updated source code for EDGUX100 so that it supersedes any old EDGUX100 USERMODs. Include the necessary JCLIN statements as shown in Figure 86 on page 242.

Changing Location Information with EDGUX100

You can use the EDGUX100 installation exit to perform these tasks:

- Specify a new location name or location type for a volume.
- Clear the location name to prevent a volume from being assigned to a location.
- Add a location name when one was not originally assigned in the disposition control file.

You can change the contents of the PL100_NAME and PL100_LOCTYPE fields to change location or name information. On input, the fields contain values set from the disposition control file.

DFSMSrmm checks the storage location name and type that you specify against the list of storage locations defined in the DFSMSrmm EDGRMMxx parmlib LOCDEF command. If you specified a storage location that is not defined to DFSMSrmm, DFSMSrmm treats the location as a loan location.

Use the PL100_INFO_CMOVE field to determine if a volume move is to be confirmed by checking for the bit that indicates that a later confirm move is required or is marked as already completed. Set PL100_SET_CMOVE or PL100_SET_NOCMOVE for move confirmation action. PL100_SET_CMOVE forces a later move. PL100_SET_NOCMOVE marks the move as already completed.

EDGUX100 Exit Routine Processing

EDGUX100 gets called at several points if DFSMSrmm does not already have the required information. For example, if the required information for the current data set and volume combination was provided in a previous call of the exit, EDGUX100 is not called again. The points that EDGUX100 can be called are as follows:

- At pre-ACS processing time for new allocations, a pointer to the ACERO is passed in PL100_ACEROPTR.
- At OPEN time in the address space of the program issuing the OPEN request. During OPEN processing the address of a copy of the JFCB is provided in PL100_JFCBPTR.

  EDGUX100 can also be called at CLOSE or EOV if the information that DFSMSrmm requires for processing is not available. For example, EDGUX100 can be called if DFSMSrmm is restarted while tape volumes are in use.
- At CLOSE and EOV time in the address space of the program processing the tape file. PL100_ITS_CLOSE is set and PL100_LABINFO provides information useful for creating labels.
- When DFSMSrmm updates a write to operator message defined by the EDGRMMxx parmlib MNTMSG command.

  You use the MNTMSG operands to identify the write to operator messages that you want DFSMSrmm to update with external volume serial number and pool information. Messages are processed for both specific and non-specific volume serial numbers. If the JES3 IATUX71 exit is used with the DFSMSrmm EDG3X71 exit, EDGUX100 is called during JES3 fetch and mount message processing.

  EDGUX100 can also be called from MSGDISP processing if the information that DFSMSrmm requires for processing is not available. For example, EDGUX100 can be called if DFSMSrmm is restarted while tape volumes are in use.

DFSMSrmm has not yet checked if the volume is defined in its control data set, if the volume will be rejected, or if the volume is subject to any other DFSMSrmm processing.

After the installation exit has been called, DFSMSrmm performs the following processing:

- The DFSMSrmm sample EDGUX100 installation exit performs no function when the supplied JFCB address, supplied ACERO address, and the supplied WTO address are zero.

- If the expiration date is updated by the exit, EDGUX100 updates the date in the copy of the JFCB for the current request. DFSMSrmm does not update the real JFCB control block.
- If a vital record specification management value is returned, DFSMSrmm records this value together with other details for the data set in the DFSMSrmm control data set, for use during inventory management.
- During pre-ACS processing if a pool or vital record specification management value is returned, DFSMSrmm sets the MSPOOL and MSPOLICY variables into the ACERO if the pre-ACS installation exit IGDACSXT has not already done so.
- If a specific pool name is returned and it meets the pool naming conventions, DFSMSrmm uses the value to update messages intercepted for MNTMSG processing, to update tape drive displays through MSGDISP processing and to perform pool validation of scratch volumes at OPEN time. If JES3 is used and IATUX71 is in use, the pool is also passed to JES3 for use in JES3 fetch and mount messages. If you request that DFSMSrmm prevents the tape drive cartridge loader from being indexed, DFSMSrmm resets the automatic cartridge load flag during MSGDISP processing. If the selected pool does not meet the naming conventions, DFSMSrmm uses the pool it has already selected.
- If the volume must be ignored, DFSMSrmm ensures that the current user is authorized to request that DFSMSrmm ignores this volume. A SAF request is used to check authorization:

  ```
  RACROUTE REQUEST=AUTH,ENTITY=STGADMIN.EDG.IGNORE.TAPE.volser,
           ACCESS=value,LOG=ASIS
  ```

  where:

  volser Is the current volume serial number of the mounted volume

  value Is either READ or UPDATE

  If the user is authorized, DFSMSrmm ignores all further activity for this volume until end-of-volume processing or until a data set on the volume is closed. This means that DFSMSrmm does not validate that the correct volume is mounted, does not record volume usage in its control data set and cannot provide any management functions for the volume based on the data about to be created.

  If authorization fails and DFSMSrmm is running in protect mode, DFSMSrmm rejects the volume. For non-specific mount requests, another volume is requested. For specific volume requests the OPEN request fails.

  If the resource does not exist, the request is treated as not authorized.

  If authorization fails and DFSMSrmm is running in record-only mode, DFSMSrmm ignores the volume and issues the information message, EDG4047I.

  If authorization fails and DFSMSrmm is running in warning mode, DFSMSrmm ignores the volume but issues error and warning messages.
- If a rack number is returned, DFSMSrmm uses this value as if it were a normal DFSMSrmm rack number for use in messages that are updated as defined by the parmlib MNTMSG definitions.
- DFSMSrmm validates the information you set in the parameter list as follows:

  PL100_VRS
  Must meet the MVS data set naming standards for a single data set qualifier.

  PL100_RACKNO
  For specific volume requests, this value must be uppercase, alphanumeric, national, or special characters.

PL100_POOL
For non-specific volume requests the pool prefix must be uppercase, one to five alphanumeric, national, or special characters ending in *.

PL100_LOCATION
If the location type is set to store or library, DFSMSrmm uses the EDGRMMxx LOCDEF definitions to validate the location. If the location name is not valid, DFSMSrmm treats the location name as a loan location.

- If you request no data set recording for a volume, DFSMSrmm updates the volume record during OPEN processing to identify that the volume does not record all data sets.
- If you specify the DFSMSrmm EDGRMMxx OPTION DISPDDNAME operand, DFSMSrmm uses the location, location type, and volume move confirmation values from the PL100 parameter list to update the volume record location and destination fields. If you request ″no confirm move″, DFSMSrmm sets the location, otherwise DFSMSrmm sets the destination for the volume. If the location type is a loan location, DFSMSrmm only records the loan location.
- If the EDGUX100 exit abnormally ends or if the exit detects an incorrect parameter list and sets return code 16, DFSMSrmm initiates an SDUMP and issues a WTOR EDG0303D which prompts the operator to reply RETRY, CANCEL, CONTINUE, or DISABLE.

  If the operator replies CANCEL, DFSMSrmm fails the current request, but processes all other requests. If the operator replies DISABLE, DFSMSrmm continues with the current request. All future requests are processed by DFSMSrmm without use of the installation exit. The installation exit is disabled. Tape processing continues. If the reply is CONTINUE, DFSMSrmm processes the current request ignoring the failure of the installation exit. All future requests are processed using the installation exit. If the reply is RETRY, DFSMSrmm retries the current request.

Assigning Expiration Dates

When a JFCB address is supplied, the sample EDGUX100 installation exit checks to see whether an expiration date, rather than a retention period, is specified by the user. If an expiration date was specified:

- If the JCL expiration date is 98000, EDGUX100 clears the expiration date so that DFSMSrmm uses the DFSMSrmm parmlib default retention period to calculate the expiration date. This prevents DFSMSrmm from treating a special expiration date as an actual expiration date. If the exit parameter list indicates that the exit can request the volume is ignored, EDGUX100 requests that DFSMSrmm ignores the volume.
- If the JCL expiration date is 99000, the sample EDGUX100 installation exit clears the expiration date field, so that DFSMSrmm uses the DFSMSrmm parmlib default retention period to calculate an expiration date. This prevents DFSMSrmm from treating a special expiration date as an actual expiration date. If the exit parameter list indicates that the exit can specify a vital record specification management value, the sample EDGUX100 installation exit returns a vital record specification management value of D99000.
- Before checking for any other expiration date values, EDGUX100 checks to see if you want the expiration dates to be used as actual expiration dates or special dates. EDGUX100 checks for the existence of a dummy DD statement with the name NOTKEYD8.

  ```
  //NOTKEYD8 DD DUMMY
  ```

Code a NOTKEYD8 dummy DD statement to indicate that the date is an actual expiration date. When you do not code the NOTKEYD8 DD statement in the current job step or do not code it as a dummy DD statement, the expiration date is treated as a special date.

If you want to use a DD name other than NOTKEYD8, change the DD name that is coded in the sample EDGUX100 installation exit. Ensure that you specify a DD name padded to 8 characters with trailing blanks.

The sample EDGUX100 installation exit does not perform any further special date processing. However, if you want to add checks for additional special dates you can do so.

Supplying a Scratch Pool Name

When the JFCB address, ACERO address, or the WTO address is supplied, the EDGUX100 installation exit can supply a scratch pool name under the following conditions:

- If the supplied ACERO address is nonzero, the jobname and data set name are taken from the ACERO.
- If the supplied JFCB address is non-zero, the job name and data set name are extracted from system control blocks.
- If the supplied WTO address is non-zero, the job name and data set name are extracted from the WTO message text.

To use the EDGUX100 installation exit for pool selection, enter the job names and data set names into a table in the assembler source code as shown in Figure 83.

DFSMSrmm scans a scratch pool table for the extracted job name and data set name to determine if this request requires a specific scratch pool. If a specific scratch pool is required, the exit requests that DFSMSrmm use the pool selected by the exit. If the cartridge loader is not to be used for the selected pool the exit requests that DFSMSrmm prevent the cartridge loader being indexed.

Specify the entries in the table in the order that you want them to be matched by the exit. The exit scans the table until it finds an entry where both the job name and data set name masks match the current request. You can change the priority of matching by changing the order of the table entries.

```
POOLTAB  DS    0F                 start of pool table
*                 jobname        data set name
         DC    CL8'*       ',CL44'BACKUP*'
         DC    CL6'A02*'            pool name
         DC    AL1(PL100_SET_ACLOFF) do not use loader
*
         DC    CL8'STSGWD* ',CL44'*'
         DC    CL6'A*'              pool name
         DC    AL1(0)               use loader
*
         DC    CL8'STSG%%* ',CL44'STSG%%.BACKUP.*'
         DC    CL6'12*'             pool name
         DC    AL1(0)               use loader
*
         DC    CL8'STSGDPW ',CL44'DAVE.POOL1.DATA'
         DC    CL6'AB999*'          pool name
         DC    AL1(PL100_SET_ACLOFF) do not use loader
*
         DC    CL8'POOLEND'       end of pool table marker
```

Figure 83. Sample EDGUX100 Pool Selection Table

Each table entry consists of:

jobname
: An 8-byte field that can contain up to 8 alphanumeric or national characters including % and *.
    The character % to ignore a positional character in the job name.
    The character * to ignore all remaining characters in the job name. A job name of * means that the entry applies to all jobs.

  Examples of job names in the table are: STSGWD*, STSG%%*, STSGDPW.

data set name
: A forty-four byte field that can contain up to forty four characters, following MVS data set naming conventions, including % and *.

The character % to ignore a positional character in the data set name.
The character * to ignore all remaining characters in the data set name. A data set name of * means that the entry applies to all data sets.

The use of the character* is not the same as in the generic data set names supported by DFSMSrmm for vital records specifications and search data set masks. The * works like the characters *.** might in a generic data set name mask.

Examples of data sets in the table are: STSGWD*, STSG%%.BACKUP.*, DAVE.POOL1.DATA.

pool name
A 6-byte field that can contain up to five alphanumeric characters ending in *. If the pool name that you specify does not match the pool naming conventions, DFSMSrmm ignores the exit selected pool and uses the DFSMSrmm selected pool.

If your exit selects a pool that does not match a VLPOOL prefix value, all volumes that are mounted are rejected.

Examples of pool names in the table are: A*, 12*, AB999*.

loader status
A 1-byte field that contains a flag indicating whether the tape drive cartridge loader is to be disabled for the named pool.

The sample EDGUX100 installation exit ORs this flag byte into the EDGUX100 parameter list PL100_FUNCTION field.

The valid values are either X'00' or X'01'. You can set the correct value using one of the following assembler statements:

```
DC   AL1(0)
DC   AL1(PL100_SET_ACLOFF)
```

You can use EDGUX100 exit selected pools for your non-system-managed volumes including volumes managed using BTLS. See "Using the EDGUX100 Installation Exit from Pre-ACS Processing" on page 231 for information about using the EDGUX100 exit to manage non-system-managed volumes using pre-ACS processing.

For system-managed volumes, use ACS routines to assign SMS constructs to new tape data sets at allocation time. DFSMSrmm does not perform pool validation for system managed volumes. As a result, if the exit selects a pool for a system-managed allocation, DFSMSrmm ignores the selection during volume validation.

If the operator mounts a volume that is not from the pool the exit specifies, the volume is rejected, and DFSMSrmm issues message EDG4021I.

```
EDG4021I VOLUME volser REJECTED. IT IS NOT IN AN ACCEPTABLE SCRATCH POOL
```

Using the System Name to Select a Scratch Pool

You can specify multiple scratch pool selection tables in your EDGUX100 installation exit. This allows you to select scratch pools based on the name of the running system.

The sample installation exit uses the SYSNAME value from the IEASYSxx parmlib member to determine the system name. This is the CVTSNAME field.

Figure 84 is an example showing how you can add system names to the existing system name table in the EDGUX100 installation exit. You can add system names in any order. If you want to remove a system from the list, delete the entry or set the system name to blanks.

```
SYSTAB   DS    0F                  start of SYSTEM table
SYSCOUNT DC    A(SYSTABL/SYSTENTL) count of entries
SYSENT1  DS    0F                  DO NOT CHANGE THIS LINE
         DC    CL8'        ',A(PTBLSYS1)    Add the names of the
         DC    CL8'        ',A(PTBLSYS2)    systems you want to
         DC    CL8'W98MVS1 ',A(PTBLSYS3)    use into the
         DC    CL8'        ',A(PTBLSYS4)    prepared entries on
         DC    CL8'        ',A(PTBLSYS5)    the left. There is no
         DC    CL8'SYSA    ',A(PTBLSYS6)    need to change table
         DC    CL8'        ',A(PTBLSYS7)    names on the right.
         DC    CL8'        ',A(PTBLSYS8)
*    To add more system names to the table, just repeat the last
*    table entry, specify a new system name, and the name of
*    new pool table. For example:
*        DC    CL8'ANOTHER ',A(PTBLSYS9)
*    Then build the new table by copying how one of the existing
*    PTBLSYSx tables are defined.
SYSEND   DS    0F                  end of table
SYSTABL  EQU   SYSEND-SYSENT1      length of table
```

Figure 84. Sample EDGUX100 Installation Exit System Name Table

You can define a pool selection table for each system name that contains data set names, pool names, and automatic cartridge loader controls as shown in Figure 83 on page 239. If there is no match found in the system table for the current system, DFSMSrmm uses the default selection pool POOLTAB. Figure 85 shows a selection table for a specific system.

```
*******************************************************************
* POOL TABLE For 3rd System
*******************************************************************
PTBLSYS3 DS    0F                  start of pool table
*                 jobname        data set name
         DC    CL8'WOODMW* ',CL44'WOODMW.SYS3.*'
         DC    CL6'S03*'           pool name
         DC    AL1(PL100_SET_ACLOFF) bypass acl load flag
*
         DC    CL8'POOLEND'        end of pool table marker
```

Figure 85. Pool Selection Table for System 3

Using Storage Group for Manual Tape Library Pooling

The sample EDGUX100 exit is written to prevent the use of storage groups for manual tape library scratch pools. Use the EDGUX100 exit as-is for manual tape library volumes to be pooled based on DFSMSrmm system-based pooling or on exit-selected pooling.

To use the storage group assigned at allocation time for pooling decisions, modify the sample EDGUX100 exit. Remove the code that overrides using storage group for pooling decisions. Do not set the PL100_SET_IGNORE_SGNAME or PL100_SET_POOL parameters when the request is for a tape drive in a manual tape library.

Setting Up the EDGUX100 Routine Environment

To reactivate the exit, stop and restart the DFSMSrmm procedure once you have corrected any error in your exit or use the F DFRMM,REFRESH EXITS operator command. See *z/OS DFSMSrmm Guide and Reference* for information.

Installing the EDGUX100 Routine

Perform the following actions to update or replace the exit:

1. Build and install an SMP/E USERMOD to apply the updated source code for the EDGUX100 installation exit. Include the necessary JCLIN statements to get the EDGUX100 load module added to the LINKLIB target library.

   You can apply the exit using an SMP/E USERMOD as shown in Figure 86. Modify the FMID and PRE to reflect the release you are running.

   a. Allocate a user SAMPLIB data set. In Figure 86 the user SAMPLIB data set is defined as MY.RMM.SRCLIB and allocated to DD card SRCLIB.
   b. Copy the EDGUX100 source from SAMPLIB to the user SAMPLIB and modify, as needed, for your installation.
   c. SMP/E RECEIVE the USERMOD.
   d. SMP/E APPLY the USERMOD. Ensure that a DD card exists for the user SAMPLIB in the APPLY job, or as a DDDEF to SMP/E in the target zone.

   After performing these steps, the modified version of the EDGUX100 exit resides in both the user SAMPLIB and SYS1.SAMPLIB. IBM's original copy is only in the distribution libraries at this point. If you accept the USERMOD, only the modified version of the exit exists. The SMP/E target zone reflects RMID indicators of VMRMM01 for all of the following records:

   ```
   SAMP EDGUX100 RMID=VMRMM01 SYSLIB=SAMPLIB
   SRC  EDGUX100 RMID=VMRMM01 SYSLIB=SAMPLIB
   MOD  EDGUX100 RMID=VMRMM01 LMOD=EDGUX100
   LMOD EDGUX100              SYSLIB=LINKLIB
   ```

   The RMID of VMRMM01 for the SAMP record prevents IBM service from being installed. This results in an ID search and notification to you that IBM is the servicing exit.

```
//RMMSTUFF  JOB  ,'SLIP IT IN',MSGCLASS=H,MSGLEVEL=(1,1)
//STEP1     EXEC SMPEMVS,REGION=6120K
//SYSIN     DD   *
 SET BDY(GLOBAL) .
 RECEIVE .
/*
//SMPPTFIN  DD   DATA,DLM=##
++USERMOD (VMRMM01) REWORK(1997082) .
++VER (Z038) FMID(HDZ11D0) .
++JCLIN .
//EDGUX100  EXEC PGM=IEWL,PARM='LET,NCAL,RENT,REUS,REFR,LIST,XREF'
//SYSLMOD   DD   DISP=SHR,DSN=SYS1.LINKLIB
//SRCLIB    DD   DISP=SHR,DSN=MY.RMM.SRCLIB
//AEDGMOD1  DD   DISP=SHR,DSN=SYS1.AEDGMOD1
//SYSPRINT  DD   SYSOUT=*
//SYSLIN    DD   *
  INCLUDE AEDGMOD1(EDGUX100)
  ENTRY   EDGUX100
  NAME    EDGUX100(R)
++SRC(EDGUX100) TXLIB(SRCLIB) DISTLIB(ASAMPLIB) .
++SAMP(EDGUX100) TXLIB(SRCLIB) DISTLIB(ASAMPLIB) .
##
/*
```

Figure 86. Building an SMP/E USERMOD to Apply the Updated EDGUX100 Exit

2. Copy the new exit load module into the LNKLST library.
3. Refresh LLA.
4. Refresh the exit by issuing:

   ```
   F DFRMM,REFRESH EXITS
   ```

If DFSMSrmm is running in a shared environment, you must repeat this step on each system.

The EDGUX100 installation exit is loaded by DFSMSrmm each time DFSMSrmm is started and stays loaded until DFSMSrmm is stopped. It can be refreshed at any time by using the operator MVS MODIFY command to refresh exits:

```
F DFRMM,REFRESH EXITS
```

Removing the EDGUX100 Routine

To remove EDGUX100 from the system, you can delete the EDGUX100 load module from the LNKLIST libraries, and refresh LLA before issuing the operator MODIFY command.

If DFSMSrmm is running in a shared environment, you must repeat this step on each system.

Writing the EDGUX100 Routine

EDGUX100 runs in SYSTEM KEY 0 or 5 AMODE(31) RMODE(ANY) in the user's address space. KEY 0 is used when a WTO address or an ACERO address is provided, and KEY 5 is used when a JFCB address is provided. It is loaded using the MVS LOAD macro, and can be contained in any APF authorized LNKLST library.

You can write your installation exit so that it can issue commands or call other programs to update external inventories. You can also code the exit to issue a WTOR if needed. Do not issue messages that are contained in the MNTMSG table, or the EDGUX100 installation exit could be called recursively.

Registers on Entry to the EDGUX100 Exit Routine

Register	Contents
0	Not applicable
1	Address of a parameter list mapped by the macro EDGPL100
2-12	Not applicable
13	Address of register save area
14	Caller's return address
15	Address of EDGUX100 entry point

EDGUX100 Parameter List

All communication is done using the parameter list. The parameter list is mapped by the macro EDGPL100 as shown in "Installation Exit Mapping Macro: EDGPL100" on page 416.

The parameter list input values are:

PL100_VALID
This field defines the functions you can request during this call of the installation exit.

PL100_CAN_IGNORE
If set to B'1' you can request that DFSMSrmm ignores the volume.

PL100_CAN_VRS
If set to B'1' you can provide a vital record specification management value for DFSMSrmm to use for this data set.

PL100_CAN_RACKNO
If set to B'1' you can provide an external volume serial number or rack number for DFSMSrmm to use for this volume in any WTO messages DFSMSrmm updates.

PL100_CAN_IGNORE_FILE2_TON
If set to B'1' you can request that DFSMSrmm record the data set details only for the first file on a tape volume.

PL100_CAN_POOL
If set to B'1' you can provide a specific pool name for DFSMSrmm to use for this volume in any WTO messages DFSMSrmm updates. Also the pool name is used to validate that a correct scratch volume is mounted for a request.

PL100_ITS_CLOSE
If set to B'1', this indicates that DFSMSrmm called EDGUX100 because a tape data set was closed or an end-of-volume condition occurred.

PL100_REQ_VOLSER
This field contains one of: PRIVAT, SCRTCH or a volume serial number. For a nonspecific request PL100_REQ_VOLSER can contain either PRIVAT or SCRTCH. For a specific request PL100_REQ_VOLSER contains the actual volume serial number requested.

PL100_MOUNT_VOLSER
This field contains the volume serial number of the volume mounted to satisfy this request. This volume serial number is only available when a data set on the volume is opened or closed.

PL100_WTOPTR
This field contains the address of the WTO message that has been intercepted by MNTMSG processing. The address is zero during OPEN processing. See the PL100_WTOPTR field for information on the operator message being updated.

PL100_JFCBPTR
This field can contain the address of the JFCB copy or is set to zero. You can use it to locate the expiration date field JFCBXPDT. You can determine if the user specified an expiration date rather than a retention period by checking the JFCB for the JFCBEXP flag. If the exit is called during OPEN processing, the JFCB address is provided. During MNTMSG processing the JFCB address is set to zero and no JFCB is available.

PL100_POOL
When PL100_JFCBPTR is zero and PL100_CAN_POOL is set to B'1', this field contains the scratch pool that DFSMSrmm has already determined should be used for this request. It is determined using the VLPOOL definitions that you specify in the DFSMSrmm parmlib member. If you do not provide a replacement value and set the PL100_SET_POOL flag, this is the pool that DFSMSrmm uses.

PL100_LABINFO
When PL100_ITS_CLOSE is set to B'1', this field points to a data area mapped by PL100_LABDS. The data area contains information about the volume and file being processed.

PL100_ACCODE
This field contains either the value of the JCL specified ACCODE parameter or blanks if the ACCODE is specified in the reduced form 'ACCODE=' or if the JCL does not contain ACCODE. The field can be 1 to eight characters as described in *z/OS MVS JCL Reference*. The first character is the ISO/ANSI accessibility code. The remaining characters can be any characters you choose. If the ACCODE value specifies a special value, then your EDGUX100 exit can process the ACCODE value rather than the special date in the EXPDT keyword if it exists.

PL100_INFO
This field defines additional information.

PL100_INFO_IGNORE
If set to B'1' The volume is to be ignored by the EDGUX100 installation exit.

PL100_INFO_NOTRMM
If set to B'1', the volume is not defined to DFSMSrmm. This is only set at CLOSE or EOV time.

PL100_INFO_DISPDD
If set to B'1', a disposition file was found and has been processed for this DD name.

PL100_INFO_CMOVE
If set to B'1', a confirm must be performed and the location requested is set as the volume's destination, not the location.

PL100_INFO_USERDATA
If set to B'1', userdata was provided from the disposition file. The user data has been included in the default label created by DFSMSrmm processing.

PL100_INFO_MTL
If set to B'1', indicates that the allocated tape drive is an manual tape library tape drive and a storage group has been set by SMS ACS processing. DFSMSrmm uses the storage group name as the specific scratch pool unless you select a specific scratch pool or request that DFSMSrmm ignore the storage group name.

PL100_ACEROPTR
This field contains the address of the ACERO system control block during pre-ACS processing.

PL100_LAB_USERDATA
This field contains the user data from the disposition processing file for use in sticky labels.

PL100_LABPTR
This field contains the address of the sticky label prepared by DFSMSrmm.

PL100_LOCATION
This field contains the name of the location specified in the disposition file.

PL100_LOCTYPE
This field contains the type of the location.

The parameter list can be updated as follows:

PL100_FUNCTION
Use this field to request functions of DFSMSrmm

PL100_SET_IGNORE
Set the PL100_SET_IGNORE bit to indicate that you want this volume ignored until the current file reaches end-of-volume or is closed.

PL100_SET_IGNORE_MOUNTED
Set the PL100_SET_IGNORE_MOUNTED bit to indicate that you want this volume ignored based on the mounted volser until the current file reaches end-of-volume or when the current volume is closed.

PL100_SET_IGNORE_REQUESTED
Set the PL100_SET_IGNORE_REQUESTED bit to indicate that you want this volume ignored, based on the requested volser until the current file reaches end-of-volume or when the current volume is closed.

PL100_SET_IGNORE_FILE2_TON
Set the PL100_SET_IGNORE_FILE2_TON bit to indicate that DFSMSrmm is only to record data set details for the first file on the tape volume.

PL100_SET_POOL
Set the PL100_SET_POOL bit to indicate that you want to use a specific pool for the current non-specific volume request. When you set PL100_SET_POOL and provide a pool prefix for a tape drive in a manual tape library, DFSMSrmm does not use the storage group assigned by SMS ACS processing.

PL100_SET_ACLOFF
Set the PL100_SET_ACLOFF bit to indicate that you want DFSMSrmm to disable the cartridge loader for this request.

The other functions you can request are determined by the presence of data in the output fields.

PL100_JFCBPTR
If a JFCB address was provided as input, set the JFCB expiration date field to the new expiration date value you want DFSMSrmm to use for this data set. You can update the JFCB expiration date even if you do not provide a vital record specification management value.

Recommendation: Zero the expiration date field in the JFCB copy to allow DFSMSrmm to calculate a default expiration date.

PL100_VRS
Set the DFSMSrmm vital record specification management value you have selected. The values you use in this field should correspond to the RMM ADDVRS DSNAME subcommands you have used. The vital record specification management value you specify is only used during inventory management vital records processing once it has been established that use of the data set name has not produced a match.

Only update this field if the PL100_CAN_VRS flag is set, as it is only when this flag is set that DFSMSrmm makes use of the value you specify. DFSMSrmm records the vital record specification management value you specify in the control data set when:
- A new data set is created
- An existing data set is rewritten with DISP=OLD
- A data set not yet recorded by DFSMSrmm is read

When an existing data set is extended, and the data set already has a vital record specification management value, DFSMSrmm ignores the new value you specify and propagates the existing vital record specification management value.

In all other cases, the vital record specification management value you specify is not used by DFSMSrmm.

PL100_RACKNO
You can provide a 6 character value for DFSMSrmm to use as the rack number or an external volume serial number for adding to the mount messages that DFSMSrmm intercepts and updates. Provide this value when requesting that DFSMSrmm ignores the volume. This helps your operators retrieve the correct volume.

Only update this field if the PL100_CAN_RACKNO flag is set, as it is only when this flag is set that DFSMSrmm makes use of the value you specify.

PL100_POOL
You can use this field to provide a specific pool name to be used with a non-specific output request.

Provide a 6 character pool identifier for DFSMSrmm to use for adding to the mount messages and tape drive display requests that DFSMSrmm intercepts and updates and for use during volume validation. Provide this value when requesting that DFSMSrmm use a specific scratch pool. This helps your operators retrieve the correct volume.

Only update this field if the PL100_CAN_POOL flag is set. Also set PL100_SET_POOL to B'1', as it is only when this flag is set that DFSMSrmm makes use of the value you specify.

If your exit selects a pool that does not meet the pool naming conventions, DFSMSrmm uses the DFSMSrmm selected pool, and ignores the setting of the PL100_SET_ACLOFF flag.

If your exit selects a pool that does not match a VLPOOL prefix value, all volumes that are mounted are rejected.

PL100_LOCATION
The value is set by DFSMSrmm depending on the keyword LOC=, or OUT= used in the disposition control file. You can change the value during EDGUX100 processing and DFSMSrmm validates it on return from the exit.

PL100_FUNCTION2
You can use this field to change some of the processing decisions that were made during disposition control processing. DFSMSrmm uses the PL100_LOCATION value to set the destination location or the current location. When DFSMSrmm sets the destination location, you must confirm the volume move later using the RMM CHANGEVOLUME subcommand. See Chapter 19, “Setting Up DFSMSrmm Disposition Processing,” on page 389 for more information about disposition control.

PL100_SET_CMOVE
Set PL100_SET_CMOVE to B'1' when you want to confirm the volume move at a later time. DFSMSrmm uses the PL100_LOCATION value to set the volume destination and not the current location. If the destination is bin-managed, DFSMSrmm sets the required location. DFSMSrmm assigns a bin number to the volume during storage location management processing.

PL100_SET_NOCMOVE
Set PL100_SET_NOCMOVE to B'1' and DFSMSrmm confirms the volume move immediately. DFSMSrmm uses the PL100_LOCATION value to set the current location when it does not need to assign a bin number for the volume.

PL100_SET_IGNORE_SGNAME
Set this field to B'1' when you want to use DFSMSrmm system-based pooling instead of the storage group name set by SMS ACS processing when the tape drive is in a manual tape library.

Registers on Return from the EDGUX100 Exit Routine

Register	Contents
0	Not applicable
1-14	Restored to contents at entry
15	Return code

EDGUX100 Installation Exit Return Codes

Table 40 shows the contents of register 15 upon return from the exit.

Table 40. EDGUX100 Installation Exit Return Codes

Return Code	Description
0	Processing was successful. The parameter list might have been updated by the exit.
16	The exit determined that the parameter list passed to it did not conform to the correct specifications.

Using the DFSMSrmm EDGUX200 Installation Exit

Use the DFSMSrmm EDGUX200 installation exit to perform these tasks:
- Return a volume to scratch status in an external inventory
- Prevent a volume from returning to scratch status
- Request that DFSMSrmm ignores data set information recorded for a volume

EDGUX200 Exit Routine Processing

The DFSMSrmm sample EDGUX200 installation exit performs these functions:
- Validates the parameter list
- Returns immediately for a system-managed volume

The EDGUX200 installation exit is called during inventory management expiration processing in the DFSMSrmm started procedure address space. It is called by DFSMSrmm each time a volume is identified for the return to scratch action. The volume has not yet been returned to scratch in either the TCDB or the DFSMSrmm control data set. If the exit requests that the volume is not returned to scratch status, DFSMSrmm leaves the volume in pending release status. If the exit requests that DFSMSrmm ignores the data set name information, all data set information for the volume is removed from the DFSMSrmm control data set and the volume is returned to scratch status.

| The EDGUX200 installation exit is not called for volumes that are under manual
| scratch control until the scratch release action has been confirmed because of

| VLPOOL AUTOSCRATCH(NO). Use the EDGUX200 installation exit to implement
| control of return to scratch without using the VLPOOL AUTOSCRATCH(NO) feature.
| For example, you can use EDGUX200 to check if the scratch action has been
| confirmed before you allow EXPROC to scratch the volume. When
| MVFLGE.MVRETSCR is set, the scratch action has not been confirmed. If you want
| to manually cleanup volumes on another system, use this flag and set the
| PL200_SET_NOSCRATCH to prevent return to scratch. When
| MVFLGE.MVRETSCR is off, manual actions are performed and confirmed, and you
| can allow the volume to return to scratch.

If the exit abnormally ends or if the exit detects an incorrect parameter list and sets return code 16, DFSMSrmm initiates an SDUMP and issues a WTOR EDG0303D which prompts the operator to reply RETRY, CANCEL, CONTINUE, or DISABLE. If the operator replies CANCEL, the volume is not returned to scratch. If the operator replies CONTINUE or DISABLE, the volume is returned to scratch and any functions that were set in the parameter list by the exit are ignored.

Setting Up the EDGUX200 Routine Environment

To reactivate the exit you can either stop and restart the DFSMSrmm procedure once you have corrected any error in your exit, or you can use the `F DFRMM,REFRESH EXITS` operator command. See *z/OS DFSMSrmm Guide and Reference* for information.

Installing the EDGUX200 Exit Routine

Perform the following actions to update or replace the exit:

1. Build and install an SMP/E USERMOD to apply the updated source code for the EDGUX200 installation exit.

   Include the necessary JCLIN statements to get the EDGUX200 load module added to the LINKLIB target library.

   You can apply the exit using an SMP/E USERMOD as shown in Figure 87 on page 250. Modify the FMID and PRE to reflect the release you are running.

   a. Allocate a user SAMPLIB data set. In Figure 87 on page 250 the user SAMPLIB data set is defined as MY.RMM.SRCLIB and allocated to DD card SRCLIB.
   b. Copy the shipped EDGUX200 source from SAMPLIB to the user SAMPLIB and modify as needed for your installation.
   c. SMP/E RECEIVE the USERMOD.
   d. SMP/E APPLY the USERMOD. Ensure that a DD card exists for the user SAMPLIB in the APPLY job or as a DDDEF to SMP/E in the target zone.

   After performing these steps, the modified version of the EDGUX200 exit resides in both the user SAMPLIB and SYS1.SAMPLIB. IBM's original copy is only in the distribution libraries at this point. If you accept the USERMOD, only the modified version of the exit exists. The SMP/E target zone reflects RMID indicators of VMRMM02 for all of the following records:

   ```
   SAMP EDGUX200 RMID=VMRMM02 SYSLIB=SAMPLIB
   SRC  EDGUX200 RMID=VMRMM02 SYSLIB=SAMPLIB
   MOD  EDGUX200 RMID=VMRMM02 LMOD=EDGUX200
   LMOD EDGUX200              SYSLIB=LINKLIB
   ```

   The RMID of VMRMM02 for the SAMP record prevents IBM service from being installed. This results in an ID search and notification to you that IBM is the servicing exit.

```
//RMMSTUFF  JOB  ,'SLIP IT IN',MSGCLASS=H,MSGLEVEL=(1,1)
//STEP1     EXEC SMPEMVS,REGION=6120K
//SYSIN     DD   *
 SET BDY(GLOBAL) .
 RECEIVE .
/*
//SMPPTFIN  DD   DATA,DLM=##
++USERMOD (VMRMM02) REWORK(1997082) .
++VER (Z038) FMID(HDZ11D0) .
++JCLIN .
//EDGUX200  EXEC PGM=IEWL,PARM='LET,NCAL,RENT,REUS,REFR,LIST,XREF'
//SYSLMOD   DD   DISP=SHR,DSN=SYS1.LINKLIB
//SRCLIB    DD   DISP=SHR,DSN=MY.RMM.SRCLIB
//AEDGMOD1  DD   DISP=SHR,DSN=SYS1.AEDGMOD1
//SYSPRINT  DD   SYSOUT=*
//SYSLIN    DD   *
  INCLUDE AEDGMOD1(EDGUX200)
  ENTRY   EDGUX200
  NAME    EDGUX200(R)
++SRC(EDGUX200) TXLIB(SRCLIB) DISTLIB(ASAMPLIB) .
++SAMP(EDGUX200) TXLIB(SRCLIB) DISTLIB(ASAMPLIB) .
##
/*
```

Figure 87. Building an SMP/E USERMOD to Apply the Updated EDGUX200 Exit

2. Copy the new exit load module into the LNKLST library.
3. Refresh LLA.
4. Refresh the exit by issuing:

   ```
   F DFRMM,REFRESH EXITS
   ```

   If DFSMSrmm is running in a shared environment, you must repeat this step on each system.

The EDGUX200 installation exit is loaded by DFSMSrmm each time DFSMSrmm is started and stays loaded until DFSMSrmm is stopped. It can be refreshed at any time by using the operator MODIFY command to refresh exits:

```
F DFRMM,REFRESH EXITS
```

Removing the EDGUX200 Routine

To remove EDGUX200 from the system, you can delete the EDGUX200 load module from the LNKLIST libraries, and refresh LLA before you issue the operator MODIFY command.

If DFSMSrmm is running in a shared environment, you must repeat this step on each system.

Writing the EDGUX200 Exit Routine

EDGUX200 runs in PROBLEM KEY, SUPERVISOR STATE AMODE(31) RMODE(ANY) in the address space of the DFSMSrmm started procedure. It is loaded using the MVS LOAD macro, and can be contained in any APF authorized LNKLST library.

You can write your installation exit to issue commands or call other programs to get external inventories updated.

Registers on Entry to the EDGUX200 Exit Routine

Register	Contents
0	Not applicable

1	Address of a parameter list mapped by the macro EDGPL200
2-12	Not applicable
13	Address of register save area
14	The caller's return address
15	Address of EDGUX200 entry point

EDGUX200 Parameter List

All communication is done using the parameter lists fields. The parameter list is mapped by the macro EDGPL200 as shown in "Installation Exit Mapping Macro: EDGPL200" on page 420.

The parameter list input values are:

PL200_VALID
This field defines which functions you can request during this call of the installation exit.

PL200_CAN_SCRTCH
If set to B'1' DFSMSrmm is returning the volume to scratch and you can request DFSMSrmm not to do this or can request DFSMSrmm to ignore data set name information for the volume.

PL200_VOLSER
This field contains the volume serial number of the volume being returned to scratch.

PL200_RACK_NUMBER
This field contains the rack number of the volume being returned to scratch.

PL200_MEDIA_NAME
This field contains the media name used for the volume being returned to scratch.

PL200_LOCATION
This field contains the location name used for the volume being returned to scratch. It can be any value that is valid for volumes in the installation. It can be any valid location name, including storage locations defined as Home locations, but not a regular storage location.

PL200_DSNAME
This field contains the name of the first data set on the volume. There might be other data sets on the volume, but this information is not available to the exit.

PL200_VOLUME_FLAGS
This flag byte is used to give you information about the volume.

PL200_SMS_VOL
If set to B'1' this volume is a system-managed volume. For system-managed volumes DFSMSrmm dynamically updates the TCDB so you do not need to.

| **PL200_HOME_LOCDEF**
| This flag byte is used to give you information that the volume is in the storage
| location defined as home.

| **PL200_MANUAL_SCRATCH**
| This flag byte is used to give you information that the volume is in VLPOOL
| with AUTOSCRATCH(NO).

| **PL200_EDGSVREC_ADDR**
| This flag byte is used to give you information about the address of the volume.

| **PL200_CATSYSID**
| This flag byte is used to give you information about the CATSYSID list for the
| running system.

PL200_DESCRIPTION
This field contains descriptive information about the volume that is returning to scratch status.

PL200_OWNER
This field contains the owner ID of the volume owner.

The parameter list can be updated as follows:

PL200_FUNCTION
This field describes functions that can be updated.

PL200_SET_NOSCRTCH
Set the PL200_SET_NOSCRTCH bit if you do not want DFSMSrmm to return the volume to scratch status at this time. You can control return to scratch processing each time inventory management expiration processing is run.

PL200_SET_IGNORE_DSN
Set the PL200_SET_IGNORE_DSN bit if you do not want DFSMSrmm to use the data set information for this volume. Setting this flag allows you to control the validation that DFSMSrmm performs at OPEN time. If DFSMSrmm cannot use the data set name information it cannot ensure that the volume has not changed since it was last used. DFSMSrmm will still use the internal volume label for validation.

Registers on Return from the EDGUX200 Exit Routine

Register	Contents
0	Not applicable
1-14	Restored to contents at entry
15	Return code

EDGUX200 Installation Exit Return Codes

Table 41 shows the contents of register 15 upon return from the exit.

Table 41. EDGUX200 Installation Exit Return Codes

Return Code	Description
0	Processing was successful. The parameter list might have been updated by the exit.
16	The exit determined that the parameter list passed to it did not conform to the correct specifications.

Chapter 12. Running DFSMSrmm with DFSMShsm

DFSMSrmm can provide enhanced management functions for the tape volumes that DFSMShsm uses for each of its tape functions. The way the two products work together depends on how you are using each of them in your installation. Run DFSMSrmm with DFSMShsm to enhance the management of the volumes DFSMShsm uses. For example, DFSMSrmm can manage the movement of tapes that must be sent out of the library for disaster recovery. Using DFSMSrmm and DFSMShsm together, you can use the scratch tape pool rather than a DFSMShsm tape pool.

DFSMSrmm provides the EDGTVEXT, and EDGDFHSM programming interfaces that can be used by products like DFSMShsm, OAM, and Tivoli Storage Manager. Use these programming interfaces for DFSMSrmm tape management so that you can maintain correct volume status. DFSMSrmm treats DFSMShsm like any other tape volume user and retains DFSMShsm volumes based on vital record specifications and retention period. DFSMShsm automatically calls EDGTVEXT so you do not need to perform any special set up for DFSMShsm to communicate with DFSMSrmm. You can use the TVEXTPURGE parmlib option in the DFSMSrmm EDGRMM*xx* parmlib member to control the action DFSMSrmm takes when DFSMShsm calls EDGTVEXT. A benefit of this interaction is that DFSMSrmm can prevent DFSMShsm from overwriting its own control data set backup, automatic dump, ABARS, and copies of backup or migration tapes. Although DFSMShsm checks its own migration and its own backup tapes, DFSMSrmm checks them as well. For more information on these programming interfaces, see “Managing DFSMShsm Tapes: EDGDFHSM” on page 201 and “Releasing Tapes: EDGTVEXT” on page 199.

Defining DFSMShsm to RACF

To run DFSMShsm with DFSMSrmm, define DFSMShsm to RACF. Define the DFSMShsm user ID with the STARTED class. See *z/OS DFSMShsm Implementation and Customization Guide*.

DFSMShsm issues the ARC0516I error message if DFSMShsm cannot successfully load the EDGTVEXT exit or if an ABEND occurs. The error message will not scroll off the screen until the operator responds to the message. If an ABEND occurs, the EDGTVEXT exit becomes disabled so that you can correct the problem. Issue the RELEASE RMM command to reactivate the DFSMShsm invocation of EDGTVEXT.

Authorizing DFSMShsm to DFSMSrmm Resources

Before you can use DFSMSrmm with DFSMShsm, you are required to authorize DFSMShsm to STGADMIN.EDG.MASTER, STGADMIN.EDG.OWNER.user, and STGADMIN.EDG.RELEASE.

If you have multiple DFSMShsm USER IDs, for example in a multi-system or multi-host environment, and any DFSMShsm ID can create tapes or return tapes to scratch status or return tapes to the DFSMShsm tape pool, you must authorize each DFSMShsm USER ID. Define STGADMIN.EDG.OWNER.hsmid for each DFSMShsm USER ID and give the other DFSMShsm USER IDs UPDATE access to it.

See Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171 for information about authorizing resources.

Table 42 shows the authorization required to use scratch tapes with DFSMShsm.

Table 42. Authorization Required to Use Scratch Tapes with DFSMShsm

Resource	Access Required
STGADMIN.EDG.RELEASE	READ
STGADMIN.EDG.MASTER	READ
STGADMIN.EDG.OWNER.hsmid	UPDATE

Table 43 shows the authorization required to use DFSMShsm with a DFSMShsm-managed scratch tape pool.

Table 43. Authorization Required to Use DFSMShsm with a DFSMShsm Scratch Pool

Resource	Access Required
STGADMIN.EDG.MASTER	UPDATE
STGADMIN.EDG.OWNER.hsmid	UPDATE

Authorizing ABARS to DFSMSrmm Resources

To use DFSMSrmm with DFSMShsm ABARS, you must assign ABARS IDs the correct levels of authorization to STGADMIN.EDG.MASTER, STGADMIN.EDG.OWNER.user, and STGADMIN.EDG.RELEASE.

If you have multiple ABARS USER IDs, for example in a multi-system environment, and any ABARS ID can return tapes to scratch status, you must authorize each ABARS USER ID. Define STGADMIN.EDG.OWNER.abarsid for each ABARS USER ID and give the other ABARS USER IDs UPDATE access to it. This allows one ABARS ID to release the tapes initially obtained from scratch by the other ABARS ID.

See Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171 for information about authorizing resources.

Table 44 shows the authorization required to use DFSMSrmm with ABARS.

Table 44. Authorization Required to Use DFSMSrmm with ABARS

Resource	Access Required
STGADMIN.EDG.RELEASE	READ
STGADMIN.EDG.MASTER	READ
STGADMIN.EDG.OWNER.abarsid	UPDATE

Setting DFSMSrmm Options When using DFSMShsm

You use the DFSMSrmm parmlib EDGRMMxx to specify the installation options for DFSMSrmm as described in “Defining System Options: OPTION” on page 134. If you are using expiration dates to manage tapes, you should consider the values you specify for the parmlib OPTION command MAXRETPD operand and the VLPOOL command EXPDTCHECK operand.

- The MAXRETPD operand specifies the maximum retention period that a user can request for data sets on volumes and is described in “Defining System Options: OPTION” on page 134.
- The TVEXTPURGE operand specifies how you want to handle the release of DFSMShsm tape volumes and is described in “Defining System Options: OPTION” on page 134.
- The VLPOOL command EXPDTCHECK operand described in “Defining Pools: VLPOOL” on page 162 tells DFSMSrmm how to manage a volume based on the expiration date field in the volume label. See “A Pooling Example” on page 73 for information about setting up pools.

When DFSMShsm sets 99365 as the expiration date to manage its tapes, 99365 means permanent retention or to never expire. If you choose to use DFSMShsm expiration date protected tape volumes, DFSMShsm sets the date 99365 to prevent DFSMSrmm from considering the volumes for release at any time. You must specify the MAXRETPD(NOLIMIT) operand to ensure that DFSMSrmm honors the 99365 date.

You also use the DFSMSrmm parmlib MAXRETPD operand value to reduce the expiration date for all volumes including DFSMShsm volumes. If you want to reduce the 99365 permanent retention expiration date, specify the MAXRETPD with a value between 0 and 9999 days.

Recommendation: Use DFSMSrmm vital record specifications instead of using the 99365 permanent retention date to retain DFSMShsm volumes. See “Defining Vital Record Specifications to Manage DFSMShsm Tapes” on page 257 for information on setting up vital record specifications.

If you use expiration dates see “Retaining DFSMShsm Tapes using Expiration Dates” on page 257 for details on how DFSMSrmm provides facilities to avoid reinitializing tapes before reuse or having the operator reply to IEC507D messages.

Setting DFSMShsm Options When using DFSMSrmm

As DFSMShsm uses a tape volume, DFSMSrmm records information about data sets and multivolume data sets at OPEN time. DFSMSrmm can use this information to manage the volumes based on the DFSMSrmm policies you define.

DFSMShsm uses the DCB Open/EOV volume security and verification exit to ensure that an acceptable volume is mounted for DFSMShsm's use. DFSMShsm uses this exit to reject unacceptable volumes. For example, DFSMShsm rejects volumes already in use by DFSMShsm and volumes that are not authorized for use by DFSMShsm. DFSMSrmm records information only for those volumes not rejected by DFSMShsm.

DFSMSrmm provides facilities so that DFSMShsm can tell DFSMSrmm when it no longer requires a tape volume or when a tape volume changes status. The benefit is that DFSMShsm cannot mistakenly overwrite one of its own tape volumes if an operator mounts a tape volume in response to a request for a non-specific tape volume.

Setting DFSMShsm System Options

Example: The DFSMShsm system options that relate to using a tape management system with global or private scratch pools are shown in the following example:

```
SETSYS EXITON(ARCTVEXT)/EXITOFF(ARCTVEXT) -
       SELECTVOLUME -
       TAPEDELETION -
       TAPESECURITY -
       PARTIALTAPE
```

Setting DFSMShsm Dump Definitions

Example: The DFSMShsm dump definitions are shown in the following example:

```
DEFINE DUMPCLASS(class -
       AUTOREUSE -
       TAPEEXPIRATIONDATE(date) -
       RETENTIONPERIOD(day))
```

DFSMSrmm Support for DFSMShsm Naming Conventions

DFSMSrmm supports all DFSMShsm options and any of the naming conventions for DFSMShsm tape data sets except for password security on tape volumes. See "Recommendations for Using DFSMSrmm and DFSMShsm" on page 269 for information about the DFSMShsm options to use with DFSMSrmm.

DFSMSrmm Support for Retention and Pooling

With DFSMShsm, you can use DFSMSrmm system-based scratch tape pools, exit-selected scratch pools, or DFSMShsm-managed tape pools.

Recommendation: Use a DFSMSrmm scratch pool. Use the DFSMShsm-managed pool only when necessary. For example, use the DFSMShsm-managed pool if you want to keep DFSMShsm-managed pools for Enhanced Capacity Cartridge System Tapes, as DFSMShsm fully uses a tape's capacity.

You must let DFSMShsm decide whether a tape volume contains valid data and whether it should return to the DFSMSrmm scratch pool or DFSMShsm-managed tape pool.

Define vital record specifications to retain tape volumes until DFSMShsm finishes with them. DFSMSrmm uses the retention period determined by the vital record specification to extend any expiration date or retention period previously set for the volume. Additionally, you can use vital record specifications to identify volumes that should be moved out of the installation media library for safe keeping, or moved from an automated to manual library.For DFSMShsm-managed tapes, you do not have to respond to the IEC507D messages that are issued for expiration data protected tapes because DFSMShsm can override expiration for its own tapes. If you choose to use DFSMShsm expiration date protected tape volumes, DFSMShsm sets the expiration date 99365 which means permanent retention to prevent DFSMSrmm from considering the volumes for release at any time.

Retaining DFSMShsm Tapes using Expiration Dates

Example: To use DFSMShsm expiration date protection, specify the DFSMShsm startup option as shown in the following example:

```
SETSYS TAPESECURITY(EXPIRATION)
```

If you use a system scratch tape pool for DFSMShsm tapes, you need a way to manage tapes protected with expiration dates that are set by DFSMShsm. To help you manage this situation, DFSMSrmm lets you automate the responses to expiration date protection messages for scratch pool tape volumes. Use the parmlib member VLPOOL command to setup this automation. Set the VLPOOL EXPDTCHECK operand to EXPDTCHECK(N) as described in “Defining Pools: VLPOOL” on page 162. DFSMSrmm automatically lets your users reuse the volumes in the pool without operator intervention and without creating data integrity exposures.

If you use a DFSMShsm-managed tape pool, DFSMShsm validates and overrides expiration dates on its emptied, previously used tapes.

Defining Vital Record Specifications to Manage DFSMShsm Tapes

Movement and retention policies are defined using vital record specifications by specifying data set names or volume serial numbers. To define vital record specifications, use the RMM ADDVRS subcommand.See *z/OS DFSMSrmm Guide and Reference* for information about the RMM ADDVRS subcommand and the operands you can specify.

When specifying the RMM ADDVRS command operands, it might be helpful to think about the operands in three categories:

- Operands to define retention policies including COUNT and CYCLES that are used in the following examples
- Operands to define movement policies including DELAY, LOCATION, and STORENUMBER that are used in the following examples
- Operands to manage the vital record specification itself including DELETEDATE and OWNER which are defaults in the RMM ADDVRS subcommand

– DELETEDATE(1999/365) specifies the date when the vital record specification no longer applies. The default value is 1999/365 which means the vital record specification is permanent. It can only be manually deleted if it is no longer appropriate.
– OWNER(*owner*) specifies the user ID that owns the vital record specification.

As shown in the examples that follow, you can specify data set name masks in vital record specifications to manage migration, backup, dumps, TAPECOPY, DUPLEX tape feature, tapes written by ABARS, ABARS accompany tapes, and control data set version backups. You can tailor the examples to define policies for your DFSMShsm tapes. As you gain more experience defining vital record specifications, you will see that there might be several ways to define the retention and movement policies you desire.

The examples use the current DFSMShsm data set naming formats. Some old name formats were in use prior to APAR OY20664. APAR OY20664 required you to use PATCH commands to use the new format. The new format is standard in DFSMShsm. If you have occurrences of the old naming formats, you might need to define vital record specifications that use both naming formats. Delete the vital record specifications with the old naming format once the old names no longer exist. See the *z/OS DFSMShsm Implementation and Customization Guide* for current DFSMShsm data set naming formats.

Retaining All DFSMShsm Tapes

You can retain all DFSMShsm tapes that require no movement, with the exception of tapes that are written by ABARS, as shown in Figure 88 and Figure 89 on page 259. See “Retaining and Moving Tapes Written by ABARS” on page 264 and “Retaining and Moving ABARS Accompany Tapes” on page 265 for examples for moving and retaining ABARS tapes.

```
RMM ADDVRS DSNAME('mprefix.**') COUNT(99999) CYCLES
RMM ADDVRS DSNAME('bprefix.**') COUNT(99999) CYCLES
RMM ADDVRS DSNAME('authid.**') COUNT(99999) CYCLES
```

Figure 88. Retaining DFSMShsm Tapes that Require No Movement

DSNAME(’*mprefix*.’)**
Specifies the DFSMShsm-defined migrated data set prefix.

DSNAME(’*bprefix*.’)**
Specifies the DFSMShsm-defined backup and dump data set prefix.

DSNAME(’*authid*.’)**
Specifies the DFSMShsm prefix used for control data set backups.

COUNT(99999)
Specifies to retain all data sets forever or until DFSMShsm releases the volume by notifying DFSMSrmm through the EDGTVEXT installation exit.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

Retaining Open Data Sets

You can define vital record specifications to control the retention of data sets that might have been left open by a system failure or that might be open during DFSMSrmm inventory management. Using the OPEN or ABEND data set name masks in a vital record specification allows you to you define specific policies for these data sets so that they are not retained like normal data sets.

Specify the JOBNAME operand with the DFSMShsm and ABARS procedure names in the vital record specifications that you define to manage open data sets. This ensures that DFSMShsm and ABARS volumes are always retained permanently under DFSMSrmm until DFSMShsm releases them.

```
RMM ADDVRS DSNAME('ABEND') JOBNAME(hsm_proc) COUNT(99999) CYCLES
RMM ADDVRS DSNAME('ABEND') JOBNAME(abars_proc) COUNT(99999) CYCLES
RMM ADDVRS DSNAME('OPEN') JOBNAME(hsm_proc) COUNT(99999) CYCLES
RMM ADDVRS DSNAME('OPEN') JOBNAME(abars_proc) COUNT(99999) CYCLES
```

Figure 89. Retaining DFSMShsm Tapes with the DFSMShsm and ABARS Procedure Name

DSNAME('ABEND')
A reserved data set name mask to manage all data sets closed as a result of an abnormal end in a task.

DSNAME('OPEN')
A reserved data set name mask to manage all data sets open when inventory management vital record processing is run or open when the system failed.

JOBNAME(*hsm_proc*)
DFSMShsm procedure name.

JOBNAME(*abars_proc*)
ABARS procedure name.

COUNT(99999)
Specifies to retain all data sets forever or until DFSMShsm releases the volume by notifying DFSMSrmm through the EDGTVEXT installation exit.

CYCLES
Specifies that DFSMSrmm should retain data sets based on cycles or copies of a data set.

Retaining Single File Format Migration Tapes

Figure 90 shows how to retain all single file format tapes created as original tapes by DFSMShsm migration until DFSMShsm releases the tapes.

```
RMM ADDVRS DSNAME('mprefix.HMIGTAPE.DATASET') -
    COUNT(99999) CYCLES
```

Figure 90. Keeping All Single File Format Migration Tapes

mprefix
Specifies the DFSMShsm-defined migrated data set prefix.

COUNT(99999)
Specifies to retain all data sets forever or until DFSMShsm releases the volume by notifying DFSMSrmm through the EDGTVEXT installation exit.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

Retaining Multifile Format Migration Tapes

Figure 91 shows how to keep multifile format migration tapes. Multifile format applies to reels and not to cartridges.

```
RMM ADDVRS DSNAME('mprefix.HMIG.T%%%%%%.**') -
    COUNT(1) CYCLES
```

Figure 91. Keeping Multifile Format Migration Tapes

mprefix
Specifies the DFSMShsm-defined migrated data set prefix.

% Represents one character of a data set name.

****** Represents zero or more qualifiers of a data set name.

COUNT(1)
Specifies to keep a single cycle. There should never be more than one cycle as DFSMShsm generates the data set names using dates and times. COUNT(99999) can be specified and provide the same results.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

Retaining Single File Format Backup Tapes

Figure 92 shows how to retain all single file format tapes created as originals by DFSMShsm backup until DFSMShsm releases the tapes.

```
RMM ADDVRS DSNAME('bprefix.BACKTAPE.DATASET') -
    COUNT(99999) CYCLES
```

Figure 92. Keeping Single File Format Backup Tapes

bprefix
Specifies the DFSMShsm-defined backup and dump data set prefix.

COUNT(99999)
Specifies to retain all data sets forever or until DFSMShsm releases the volume by notifying DFSMSrmm through the EDGTVEXT installation exit.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

Retaining Multifile Format Backup Tapes

The example in Figure 93 on page 261 shows how to keep multifile format backup tapes. Multifile format applies to reels and not to cartridges.

```
RMM ADDVRS DSNAME('bprefix.BACK.T%%%%%%.**') -
    COUNT(1) CYCLES
```

Figure 93. Keeping Multifile Format Backup Tapes

bprefix
: Specifies the DFSMShsm-defined backup and dump data set prefix.

% Represents one character of a data set name.

****** Represents zero or more qualifiers of a data set name.

COUNT(1)
: Specifies to keep a single cycle. There should never be more than one cycle as DFSMShsm generates the data set names by using dates and times. COUNT(99999) can be specified and provide the same results.

CYCLES
: Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

Retaining and Moving TAPECOPY Tapes or DUPLEX Tapes

The DFSMShsm TAPECOPY command and the DUPLEX tape feature create tapes that are called alternate tapes. Both the TAPECOPY command and the DUPLEX tape feature use the same naming convention for the alternate tapes.

To retain alternate tapes and to move them to another location,you can define two vital record specifications, one for backup copies and one for migration copies as shown in Figure 94.

```
RMM ADDVRS DSNAME('mprefix.COPY.HMIGTAPE.DATASET') -
    COUNT(99999) CYCLES LOCATION(REMOTE) STORENUMBER(99999)

RMM ADDVRS DSNAME('bprefix.COPY.BACKTAPE.DATASET') -
    COUNT(99999) CYCLES LOCATION(REMOTE) STORENUMBER(99999)
```

Figure 94. Keeping Tapes Created by the DFSMShsm TAPECOPY Command and DUPLEX Tape Feature

mprefix
: Specifies the DFSMShsm-defined migration data set prefix.

bprefix
: Specifies the DFSMShsm-defined backup and dump data set prefix.

COUNT(99999)
: Specifies to retain all data sets forever or until DFSMShsm releases the volume by notifying DFSMSrmm through the EDGTVEXT installation exit.

CYCLES
: Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

LOCATION(REMOTE)
: Specifies the storage location called REMOTE to which the number of volumes for the data sets specified in STORENUMBER should be moved.

STORENUMBER(99999)
: Specifies that all copies should be kept in the specified storage location until they are returned to scratch by DFSMShsm. Specifying STORENUMBER as

99999 in the two examples ensures that when DFSMShsm release the original volumes, it also returns the alternate tapes.

As you create copies of migration and backup tapes with the DFSMShsm TAPECOPY command, DFSMSrmm recognizes the tapes, retains them, and identifies them for movement to the named storage location. During DFSMShsm RECYCLE processing, DFSMShsm releases an alternate tape when it releases the original tape. DFSMSrmm identifies that these volumes should return to the library after DFSMShsm releases the volume. You can use this mechanism to get copy tapes ejected from an automated tape library after creation and returned to the library after RECYCLE.

Use of DFSMSrmm to manage movement is only available for those alternate volumes you have created since implementing the new DFSMShsm TAPECOPY data set name format. The old naming format used for alternate volumes was the same as that used for the base volumes. If you have copies which were created with the old naming convention we suggest that you use DFSMShsm RECYCLE against the base volumes or take a new copy of the base volume. See “Disaster Recovery Using DFSMShsm Alternate Tapes with DFSMSrmm” on page 268 for information on use of alternate tapes during recovery.

Retaining and Moving Dump Tapes

When defining vital record specifications for retaining dump tapes, consider these conditions:

- The dump class definitions because the dump classes and the vital record specifications you define must work together.
- The dump class and DASD volume serial numbers when you define the data set names you use in the data set name masks.

  The data set name format used by DFSMShsm for dump tapes includes the dump class and the DASD volume serial number. These are the two most likely variables in the data set name on which you will base your retention and movement policies.
- The dump classes that are managed the same.
- The dump classes that require separate management.
- The dump classes that need to be stored off site.
- The need to define additional vital record specifications to support the old data set naming conventions for a period of time.

Figure 95 is the minimum you are required to specify in a vital record specification for DFSMSrmm to keep dump tapes and to prevent DFSMSrmm from releasing DFSMShsm volumes.

```
RMM ADDVRS DSNAME('bprefix.DMP.**') -
    COUNT(1) CYCLES
```

Figure 95. Keeping Tapes Used for Dump

bprefix
: Specifies the DFSMShsm-defined backup and dump data set prefix.

******
: Represents zero or more qualifiers of a data set name.

COUNT(1)
: Specifies to retain a single cycle. In the example, the COUNT(1) is used to

retain a single dump cycle. Since DFSMShsm generates a unique name for each dump cycle, COUNT(1) retains all dump cycles.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

If you want to be more specific and manage cycles of dumps, using differing policies you can extend the data set name mask as shown in Figure 96.

```
RMM ADDVRS DSNAME('bprefix.DMP.class.V%%%%%.**') -
    COUNT(1) CYCLES
```

Figure 96. Managing Cycles of Dumps

bprefix
Specifies the DFSMShsm-defined backup and dump data set prefix.

class
Specifies the DUMPCLASS you have defined to DFSMShsm.

V%%%%%
The six character volume serial number.

** Represents zero or more qualifiers of a data set name.

COUNT(1)
Specifies to retain a single data set that matches the filter mask.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

You can also define vital record specifications to manage different types of dump classes having different cycles. Figure 97 shows the use of a pseudo-GDG data set name to identify dump cycles for movement to storage locations for disaster recovery. A DFSMSrmm pseudo-GDG is a collection of data sets, using the same data set name, that DFSMSrmm manages like a GDG. A pseudo-GDG data set name contains the ¬ as a placeholder for the characters in the pattern that change with each generation.

In the example, ¬ is used to mask the DFSMShsm generated date and time in the data set names so that all generations of a dump can be logically managed together.

```
RMM ADDVRS DSNAME('bprefix.DMP.class.V%%%%%.T¬¬¬¬¬¬.D¬¬¬¬¬') -
    COUNT(100) CYCLES LOCATION(REMOTE) STORENUMBER(2) -
    DELAY(1)
```

Figure 97. Retaining and Moving Volumes by Cycles

bprefix
Specifies the DFSMShsm-defined backup and dump data set prefix.

V%%%%%
The six character volume serial number.

¬ Is a place holder for a single character in a data set name mask for a pseudo-GDG data set name. When ¬ is used in a data set name mask, DFSMSrmm manages the data sets matching the data set name mask like a generation data group.

COUNT(100)
Specifies to retain all dump cycles managed by DFSMShsm. 100 is the limit for the number of dump cycles.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

LOCATION(REMOTE)
Specifies the storage location called REMOTE to which the number of volumes for the data sets specified in STORENUMBER should be moved.

STORENUMBER(2)
Two cycles are kept in the storage location REMOTE. All other cycles up to the COUNT(100) value are retained in the HOME location.

DELAY(1)
The current cycle is kept in the library for one day before being removed to the storage location.

In Figure 97 on page 263 all 100 cycles of a dump taken for a specific volume and class combination are kept if produced by DFSMShsm. For each matching vital record group:

- The current cycle is kept in the library for one day before being removed to the storage location.
- Two cycles are kept in the storage location.
- All other cycles are returned to and kept in the library until DFSMShsm releases them.
- You can use any combination of LOCATION, STORENUMBER, NEXTVRS, and DELAY to provide the required level of service.
- COUNT(100) is coded so DFSMSrmm keeps all cycles that DFSMShsm produces. You can tailor the example by using different values for the class name.

You might need to define additional vital record specifications to support the old data set naming conventions for a period of time.

Retaining and Moving Tapes Written by ABARS

Figure 98 on page 265 shows how to keep 10 tapes written by ABARS that were created under DFSMS, and provide storage location management for them based on cycles as if they are a generation data group definition.

When you define a vital record specification as shown in Figure 98 on page 265, DFSMSrmm keeps 10 versions of each of the aggregate backup output data sets and any copies ABARS created. DFSMSrmm retains the latest 10 versions in the REMOTE location and any others in the home location. If there is only one copy of each aggregate backup version produced, and there are 10 versions of each aggregate backup, DFSMSrmm keeps 10 versions of each aggregate backup. If there are 2 copies of each, DFSMSrmm keeps 10 versions of each copy of each aggregate backup.

In the example, the COUNT and STORENUMBER are the same. You could use different values where STORENUMBER is less than or equal to COUNT when you want to store a number of tapes offsite.

```
RMM ADDVRS DSNAME('outputdatasetprefix.%.C%%V¬¬¬¬') -
    COUNT(10) CYCLES -
    LOCATION(REMOTE) STORENUMBER(10) -
```

Figure 98. Keeping ABARS Tapes

outputdatasetprefix
: Specifies the same value as entered in the ISMF aggregate processing application as the prefix to be used for output data set.

% Represents the characters D, C, O, and I that are used by DFSMShsm ABARS processing.

C%%
: Represents the copy number.

V¬¬¬¬
: Represents the version number DFSMShsm maintains as a pseudo-generation data group.

COUNT(10)
: Specifies the number of versions of the aggregate backup to retain.

CYCLES
: Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

LOCATION(REMOTE)
: Specifies the storage location called REMOTE to which the number of volumes for the data sets specified in STORENUMBER should be moved.

STORENUMBER(10)
: Represents a number of versions to be kept in a storage location.

Retaining and Moving ABARS Accompany Tapes

Retaining ABARS accompany tapes applies equally to old ABARS tapes and ABARS tapes created under DFSMS. You can retain and manage ABARS accompany tapes based on the naming conventions your installation uses. Modify the example shown in Figure 99 with data set names, COUNT, and STORENUMBER values for your installation. In the example shown in Figure 99, we assume that all accompany tapes end with '.COPY'.

To keep ABARS accompany tapes, specify the following command:

```
RMM ADDVRS DSNAME('app11.**.COPY') COUNT(10) CYCLES -
    LOCATION(REMOTE) STORENUMBER(10) DELAY(1)
```

Figure 99. Keeping ABARS Accompany Tapes

app11..COPY**
: Specifies the data set name mask that identifies the application data set names to retain. DFSMSrmm keeps all data sets that begin with prefix app11 and end with COPY. You can use any data set name you choose

and can select vital record specification options to best match those selected for your aggregate groups or other retention and movement policies.

COUNT(10)
Specifies the number of versions of the data set to retain.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

LOCATION(REMOTE)
Specifies the storage location called REMOTE to which the number of volumes for the data sets specified in STORENUMBER should be moved.

STORENUMBER(10)
Represents a number of versions to be kept in a storage location.

Retaining DFSMShsm Control Data Set Backup Tapes

To keep all cycles of DFSMShsm control data set and journal backup tapes until DFSMShsm releases them, issue RMM ADDVRS subcommands as shown in Figure 100:

```
RMM ADDVRS DSNAME('authid.%CDS.BACKUP.V¬¬¬¬¬¬¬') -
    COUNT(99999) CYCLES
RMM ADDVRS DSNAME('authid.JRNL.BACKUP.V¬¬¬¬¬¬¬') -
    COUNT(99999) CYCLES
```

Figure 100. Keeping DFSMShsm Control Data Set and Journal Backup Tapes

authid*.*
Specifies the DFSMShsm prefix used for control data set backups.

% Represents the characters M, B, or O used by DFSMShsm for each of its control data sets.

V¬¬¬¬¬¬¬
Represents the version number DFSMShsm maintains as a pseudo-generation data group.

COUNT(99999)
Specifies to retain all data sets forever or until DFSMShsm releases the volume by notifying DFSMSrmm through the EDGTVEXT installation exit.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

All versions of each of the control data set and journal backups are retained until released when DFSMShsm calls DFSMSrmm to release the tape volume. You can optionally use any of the RMM ADDVRS subcommand operands to meet your movement and retention requirements. For example, you can use the LOCATION and STORENUMBER operands to identify that movement is required.

Retaining Cycles of Dump Tapes

You can move dump cycles to storage locations for disaster recovery. Use additional options for pseudo-generation data groups, to retain and move by CYCLES using the old naming format as shown in Figure 101 on page 267.

```
RMM ADDVRS -
DSNAME('bprefix.DMP.T¬¬¬¬¬¬.class.D¬¬¬¬¬.V%%%%%%') -
    COUNT(100) CYCLES LOCATION(REMOTE) STORENUMBER(2) -
    DELAY(1)
```

Figure 101. Moving Dumps to Storage Locations

bprefix Specifies the data set name mask that identifies the application.

T* Is the time mask to mask the time the dump was created.

class Specifies the DUMPCLASS you have defined to DFSMShsm.

D* Is the date mask to mask the date the dump was created.

V%%%%%%
The six character volume serial number.

COUNT(100)
Specifies to retain all dump cycles managed by DFSMShsm. 100 is the limit for the number of dump cycles.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

LOCATION(REMOTE)
Specifies the storage location called REMOTE to which the number of volumes for the data sets specified in STORENUMBER should be moved.

STORENUMBER(2)
Represents a number less than the total number of versions to be kept. All other cycles are retained, up to the COUNT value in the HOME location.

DELAY(1)
The current cycle is kept in the library for one day before being removed to the storage location.

You might need to use both types of dump vital record specifications until you have replaced all old retained volumes.

Retaining ABARS Backup Tapes

For ABARS backups taken prior to DFSMS, the naming convention was different and used true GDG names. Figure 102 shows an example for keeping 10 ABARS backup tapes using GDG names.

```
RMM ADDVRS DSNAME('outputdatasetprefix.%') GDG -
    COUNT(10) CYCLES LOCATION(REMOTE) STORENUMBER(10) -
    DELAY(1)
```

Figure 102. Keeping ABARS Backup Tapes Using GDG Names

outputdatasetprefix
Specifies the same value as entered in the ISMF aggregate processing application as the prefix to be used for output data set.

% Represents the characters D, C, O, and I used by DFHSM ABARS processing.

GDG
Specifies that the data set name is a generation data group name.

COUNT(10)
Specifies the number of versions of the aggregate backup to retain.

CYCLES
Specifies that DFSMSrmm retain data sets based on cycles or copies of a data set.

LOCATION(REMOTE)
Specifies the storage location called REMOTE to which the number of volumes for the data sets specified in STORENUMBER should be moved.

STORENUMBER(10)
Represents a number of versions to be kept in a storage location.

DELAY(1)
The current cycle is kept in the library for one day before being removed to the storage location.

Retaining DFSMShsm Tapes Extra Days Retention

You can use DFSMSrmm to release DFSMShsm tapes that are requested to be purged by DFSMShsm. By default, the expiration date protection for DFSMShsm tapes is done by DFSMShsm. DFSMShsm uses 1999/365 as the expiration date for permanent retention. To enable extra days retention for purged DFSMShsm tape volumes, you need to set up retention options in the vital record specifications that are used to retain the tape volumes.

1. Ensure that the DFSMSrmm EDGRMMxx parmlib OPTION MAXRETPD operand is set to NOLIMIT to prevent DFSMSrmm from reducing the expiration date used for the DFSMShsm tape volumes. See "Defining System Options: OPTION" on page 134 for information about the MAXRETPD, VRSEL, and TVEXTPURGE operands.
2. Specify the DFSMSrmm EDGRMMxx parmlib OPTION VRSEL(NEW) operand and the OPTION TVEXTPURGE(EXPIRE) operand.
3. Define a name vital record specification that specifies the EXTRADAYS retention type and chain the name vital record specification to the vital record specifications used to retain DFSMShsm tape volumes. Also use the COUNT operand to specify the number of days you would like the tape volumes to be retained.
4. Include the UNTILEXPIRED retention type in the vital record specifications you use to retain DFSMShsm tape volumes and chain these vital record specifications to the name vital record specifications that include the EXTRADAYS retention type.

```
RMM ADDVRS DSNAME('mprefix.**') UNTILEXPIRED NEXTVRS(HSMEXT)
RMM ADDVRS DSNAME('bprefix.**') UNTILEXPIRED NEXTVRS(HSMEXT)
RMM ADDVRS DSNAME('authid.**') UNTILEXPIRED NEXTVRS(HSMEXT)
RMM ADDVRS NAME(HSMEXT) EXTRADAYS COUNT(N)
```

Disaster Recovery Using DFSMShsm Alternate Tapes with DFSMSrmm

DFSMShsm supports the creation of a duplicate cartridge tape, called an alternate, for each migration tape or each backup tape. The major use of this is to support remote storage of the alternate so that it can easily be used in case of a disaster.

The recommended process includes the following steps related to tape usage:

1. Have a copy of the DFSMShsm control data sets at the disaster site.
2. Perform a TAPEREPL by specifying the DISASTERALTERNATE parameter. This flags each existing alternate tape as a disaster alternate.

3. Place DFSMShsm in DISASTER mode. When in disaster mode DFSMShsm dynamically checks before mounting an input tape whether the needed data resides on a tape having a disaster alternate. If it does, the disaster alternate is requested.

DFSMSrmm recognizes when DFSMShsm is opening tape data sets and tolerates the data set names that DFSMShsm uses as long as the last 17 characters of the data set name match.

When you perform a true replacement by using the TAPEREPL command, without the DISASTERALTERNATE keyword, DFSMShsm uses the data set name it uses for the original tapes.

Securing Tapes When Running DFSMShsm and DFSMSrmm

How you use the DFSMShsm SETSYS option TAPESECURITY influences the TPRACF value in the DFSMSrmm parmlib member EDGRMMxx.DFSMShsm uses RACF TAPEVOL profiles to secure its volumes. You can combine the DFSMShsm method for securing volumes with DFSMSrmm RACF options as described in "Defining System Options: OPTION" on page 134 to ensure complete security for your tape data.

Select the appropriate combinations for your environment from the following options:
- Use RACF with TAPEVOL and TAPEDSN.
- Use DFSMShsm TAPESECURITY with RACF or EXPIRATIONINCLUDE.
- Use DFSMSrmm TPRACF with A, P, or N and VLPOOL RACF with Y or N.

Recommendations for Using DFSMSrmm and DFSMShsm

Related Reading: See *z/OS DFSMShsm Implementation and Customization Guide* for details on DFSMShsm data set naming formats.

Follow these guidelines when running DFSMSrmm with DFSMShsm:
- Run DFSMShsm with scratch tapes that DFSMSrmm manages so you can have a single scratch pool for all users of tape and so you can gain any benefits available from pre-mounting of scratch tapes in cartridge loaders.
- Use RACF tape security to ensure data security on your system.
- Set the following suggested DFSMShsm system option parameters.

  ```
  SETSYS EXITOFF(ARCTVEXT) -
         SELECTVOLUME(SCRATCH) -
         TAPEDELETION(SCRATCH) -
         TAPESECURITY(RACF) -
         PARTIALTAPE(MARKFULL)
  ```

  Specify EXITOFF(ARCTVEXT) if DFSMSrmm is your only tape management product because DFSMShsm always calls the DFSMSrmm programming interface EDGTVEXT.

  Specify SELECTVOLUME(SCRATCH) to cause DFSMShsm to issue a nonspecific mount request for output tapes which can be satisfied by any empty tape acceptable to both DFSMShsm and DFSMSrmm.

  Specify PARTIALTAPE(MARKFULL):
  - If you are using automatic cartridge loaders so DFSMShsm marks partially used volumes as full. This allows the first output tape of each task to be a nonspecific mount, which means that you get faster tape mounts. You can specify migration and backup options separately.

- If you want to get migration and backup data duplicated and moved to a storage location in a timely manner. Marking the tapes full prevents a task's last tape, that is only partially filled, from remaining in the library an additional 24 hours before being duplicated the next day for disaster protection.
- If you use TAPECOPY or the DUPLEX tape feature to duplicate tapes and ship them to a storage location, use MARKFULL to end one day's cycle and begin another day's cycle. TAPECOPY only processes full volumes which means that the original tapes have no copy until they are full. This means that a copy might not be available at the storage location as quickly as expected. Using the DUPLEX tape feature means that both the original and copy tapes are created at the same time. DFSMSrmm ships the DUPLEX copy to the storage location as quickly as possible. This might create problems if DFSMShsm expects to continue the writing to the tape the next day. Marking the DUPLEX tapes full allows DFSMSrmm to process both TAPECOPY copies and DUPLEX tapes in the same way.

• Set the following suggested dump definitions:

```
DEFINE DUMPCLASS(class -
      AUTOREUSE -
      RETENTIONPERIOD(days))
```

Use RETENTIONPERIOD instead of TAPEEXPIRATIONDATE to allow the tapes to be reused or written to by any other program without their needing to be reinitialized after DFSMShsm expires them.

Use AUTOREUSE to have the tapes returned to a scratch pool as soon as the data on the tape are invalidated. This option is necessary for DFSMShsm to call DFSMSrmm to release the tape volume. Without AUTOREUSE, a DELVOL command must be issued for each tape after it is returned to the locale of the tape drives.

Do not use AUTOREUSE for a DFSMShsm-managed pool when also removing tapes from the vicinity of the tape drives. This is because DFSMShsm might select a remotely located tape before DFSMSrmm is able to cause the tape's physical return to the locale of the tape drives.

Chapter 13. Running DFSMSrmm with JES3

DFSMSrmm provides SMP/E USERMODs in SAMPLIB that you can apply to the standard JES3 user exits and other JES3 modules. Use EDG3UX71, EDG3UX29, and EDG3UX62 to set up DFSMSrmm with JES3. Install the USERMODs to prevent JES3 from validating certain volume mounts, to update JES3 fetch and mount messages, and to enable the use of no label tape volumes with JES3.

In a JES3 system, JES3 validates volume mounts when JES3 is managing tape drives and performing pre-processing setup. JES3 ensures that scratch tapes have reached their expiration date and that volumes are correctly write-enabled or protected. JES3 validation can conflict with DFSMSrmm and RACF processing. For example, you can use 'logical write protect' with IBM tape drives to prevent a user from writing to a tape even if the tape is physically write-enabled. Since DFSMSrmm ensures that all scratch tapes are valid and provides features to ignore the expiration dates on volumes, you might want to prevent JES3 from validating certain volume mounts.

Preventing JES3 from Validating Volumes

Use the EDG3UX29 USERMOD to prevent JES3 from validating volumes to avoid conflicts with DFSMSrmm validation.

In a JES3 system, the default value for EXPDTCHK is YES. This value can conflict with the tape management system function that allows you to use expiration date protected tapes. EDG3UX29 sets EXPDTCHK=NO.

In a JES3 system, the default value for RINGCHK is YES. This value can conflict with the tape management system function, DFSMSdfp, and RACF function that forces logical write protect or allows a user to open a tape data set for READ even though the tape is write enabled. EDG3UX29 sets RINGCHK=NO.

Updating JES3 Fetch and Mount Messages

DFSMSrmm provides USERMODs for use when updating JES3 fetch and mount messages with shelf location and pool information.

When you are using EDGUX100 for scratch pooling as described in "Using the DFSMSrmm EDGUX100 Installation Exit" on page 221, use the EDG3UX71 USERMOD to ensure the correct tape pool is requested.

In a JES3 complex all systems running DFSMSrmm must have the same EDGUX100 installation exit and the same parmlib VLPOOL operand values to ensure that a scratch volume is not rejected because it is from the wrong pool. DFSMSrmm also provides USERMODs EDG3LVVR and EDG3IIP1 that you can use to ensure that the correct tape pool is requested. If you do not use one of the USERMODs which allows DFSMSrmm to update a message or tape display, DFSMSrmm cannot provide tape pool information and the wrong tape pool might be requested. Any incorrect tape mounted is rejected by DFSMSrmm volume validation and a remount requested. The remount request does use the correct pool.

Steps for Using the EDG3UX71 USERMOD

Before you begin: Validate that the EDG3UX71 USERMOD does not conflict with any modifications that you have made to the IATUX71 exit.

DFSMSrmm supplies a USERMOD to IATUX71, called EDG3UX71, in SAMPLIB. Use EDG3UX71 to update IATUX71 in order to replace and append text to JES3 fetch-and-mount messages and to provide text for tape drive displays. Perform the following steps to use the EDG3UX71 USERMOD to update the IAT5210 message:

1. Install the USERMOD by using SMP/E.
2. Set the SETPARAM DSN option in the JES3 initialization deck to a value other than 0. For exit-selected scratch pooling, EDGUX100 depends on the data set name being included in the fetch-and-mount messages. Ensure that you have the DSN keyword coded on the JES3 SETPARAM statement in the initialization deck, and that the data set name length value is sufficient to provide selection control in EDGUX100. The value is the length of the data set name to be included in mount messages and fetch messages that are issued by JES3.

   Recommendations:

   a. Set the data set name length to 9 if you use only the first qualifier for pooling.
   b. Set the data set name length to 31 if you want the EDGUX100 installation exit to use the data set name for pool selection.

Result: The IAT5210 message is updated.

Using the EDG3IIP1 USERMOD

Use the EDG3IIP1 USERMOD in SAMPLIB to update IATIIP1 to force DEFER for all tape requests. If you use EDG3UX71, you do not need to use EDG3IIP1. The EDG3IIP1 USERMOD updates IATIIP1 to mark tape allocations for deferred processing. This is equivalent to coding UNIT=(unit,,DEFER) as a JCL keyword.

Before you begin: Validate that the EDG3IIP1 USERMOD does not conflict with any other modifications to the IATIIP1 module.

Install the USERMOD by using SMP/E.

Using the EDG3LVVR USERMOD

Use the EDG3LVVR USERMOD in SAMPLIB to update the tape drive display with the correct exit selected pool. The EDG3LVVR USERMOD updates IATLVVR to AWAIT MSGDISP for scratch mount. If you use EDG3UX71, you do not need to use EDG3LVVR.

Before you begin: Validate that the EDG3LVVR USERMOD does not conflict with any other modifications to the IATLVVR module.

The EDG3LVVR USERMOD updates IATLVVR to add a short wait to the verify function. The wait allows the setup function time to issue the IAT5210 message. A JES3 MCS console is required when using this USERMOD.

1. Define a JES3 MCS console.

   Refer to *z/OS JES3 Initialization and Tuning Guide* for details on how to define MCS consoles for use with JES3. You need to define an MCS console with logical association to JES3 and use it as the tape operator console to receive the mount messages as updated by DFSMSrmm. Use the JES3 CONSOLE statement with the TYPE=MCS keyword.

   Use the MCS console routing codes to select which JES3 issued messages are seen on the MCS console. JES3 routes the tape messages to MCS consoles with a route code of 3.

This method cannot be used in a multi-system JES3 complex where tape drives are attached to a JES3 local processor, because DFSMSrmm can only correctly update the tape drive display on the global processor.

2. Install the USERMOD by using SMP/E.

Using the EDG3UX62 USERMOD to Create and Mount No Label Tapes

Before you begin: Validate that the EDG3UX62 USERMOD does not conflict with any other modifications to the IATUX62 module.

Installing the EDG3UX62 USERMOD is optional. If you decide to install it, use SMP/E.

Install EDG3UX62 when you are using JES3 pre-execution setup for tape volumes.

After you implement EDG3UX62, you can accomplish these tasks:

- Create NL tape volumes from scratch volumes.
- Use duplicate volumes when using deferred tape mount processing.
- Mount a volume with standard labels for a non-specific NL request.
- Process any standard label volume when requesting a specific volume where the specific volume and the mounted volume have the same label type. For example, the user requests an AL tape volume and the operator mounts an AL tape volume.
- Specify volumes with duplicate volume serial numbers.

The EDG3UX62 USERMOD updates IATUX62 to override the JES3 decision to reject a tape when a non-specific NL tape is requested and a standard label write-enabled tape is mounted. During JES3 verify processing, JES3 ensures that the mounted volume matches the requested volume. If an NL tape is requested, JES3 ensures that an NL tape is mounted. For specific volume requests this is correct processing, but for non-specific requests DFSMSrmm forces all scratch volumes to contain a standard volume label. As a result, JES3 rejects any scratch volume that is mounted.

Chapter 14. Performing Inventory Management

DFSMSrmm Samples Provided in SAMPLIB

- EDGJHSKP Sample JCL for Using the EDGHSKP Utility
- EDGJHKPA Sample IBM Tivoli Workload Scheduler for z/OS Job for Allocating the Data Sets Required for Inventory Management

DFSMSrmm provides the EDGHSKP utility to help you perform inventory management. You can run inventory management functions in a single job step or can split requests into multiple steps and jobs if required. Running inventory management as a single step allows DFSMSrmm to optimize processing. The default processing for EDGHSKP is to run all inventory management functions in sequence as described in “EXEC Parameters for EDGHSKP” on page 283. Use this chapter to set up and run inventory management activities and to schedule all DFSMSrmm utilities.

You use the EDGHSKP utility to request that DFSMSrmm perform inventory management processing. The utility validates parameters, checks that the correct files are allocated and can be used, and requests that the DFSMSrmm subsystem performs the functions you request. The utility uses the EDGBKUP utility to perform backup processing.

During DFSMSrmm subsystem inventory management processing, you can see the address space executing EDGHSKP waiting for the subsystem processing to complete. Use the RMM LISTCONTROL subcommand with the CNTL operand to see which inventory management functions are active and when the individual functions were last successfully processed. If the batch job requesting an inventory management function is cancelled after the function has started, the function continues to run in the DFSMSrmm address space until it completes.

During inventory management the DFSMSrmm subsystem issues messages that describe processing. Messages are in the MESSAGE data set. During expiration processing, DFSMSrmm identifies data sets that might be open and puts them in a list in the MESSAGE file. See Figure 115 on page 305 for an example of this output.

Scheduling DFSMSrmm Utilities

You can schedule each DFSMSrmm utility to run on its own, or schedule many activities to run together in a sequence. You can schedule activities such as vital records processing to be performed daily. Other activities can be scheduled less frequently, for example on a weekly basis. You can use a job scheduling product like the IBM Tivoli Workload Scheduler for z/OS to run the utilities. See Chapter 20, “Running DFSMSrmm with the IBM Tivoli Workload Scheduler for z/OS,” on page 395 for information about using the IBM Tivoli Workload Scheduler for z/OS for scheduling DFSMSrmm utilities.

Table 45 on page 276 suggests how frequently to run DFSMSrmm utilities. References are listed to help you locate the information about the utilities.

Table 45. Scheduling DFSMSrmm Utilities

Activity	**Frequency**	**Reference**
Processing vital records	Daily	“JCL for EDGHSKP” on page 282 and “Running Vital Record Processing” on page 288
Performing expiration processing	Daily	“JCL for EDGHSKP” on page 282 and “Running Expiration Processing” on page 302
Performing storage location management	Weekly	“JCL for EDGHSKP” on page 282 and “Running Storage Location Management Processing” on page 300
Creating an extract data set	Daily	“JCL for EDGHSKP” on page 282
Creating volume movement and inventory reports	Weekly	See *z/OS DFSMSrmm Reporting*
Creating security and audit reports	Monthly	See *z/OS DFSMSrmm Reporting*
Backing up the DFSMSrmm control data set	Daily	“JCL for EDGHSKP” on page 282, “Backing Up the Control Data Set” on page 310, and “Backing Up the Control Data Set” on page 324
Backing up the DFSMSrmm journal	Daily	“JCL for EDGHSKP” on page 282, “Backing Up the Control Data Set” on page 310, and “Backing Up the DFSMSrmm Control Data Set and Journal” on page 324
Verifying the control data set	Monthly	“Using EDGUTIL for Tasks Such as Creating and Verifying the Control Data Set” on page 336
Initializing and erasing volumes	Weekly	Chapter 16, “Initializing and Erasing Tape Volumes,” on page 351

Running Inventory Management

If you decide to run inventory management in multiple steps, we suggest that you run inventory management as follows:

1. Back up the control data set and journal and keep multiple backup generations to aid recovery.
2. Run vital record processing first before expiration and storage location management processing. This is to identify which volumes to retain and where volumes should be moved, based on vital record specifications.
3. Run expiration processing to identify those volumes not required for vital records that are ready to expire. During expiration processing, release actions for volumes are noted.
4. Run storage location management to set a destination location for a volume. Optionally run storage location management to assign shelf locations in storage locations for volumes that are being sent out of or returned to the removable media library. You must run storage location management processing after vital record processing has been successfully run, but not necessarily in the same run of EDGHSKP.
5. Run an EDGINERS job after expiration processing completes to ensure that volumes that are waiting to be released and waiting to be initialized are returned to scratch. Use automatic processing to request EDGINERS to initialize and erase any volumes flagged in the control data set. Run EDGINERS before and after each run of EDGHSKP.
6. Create an extract data set to use as input to the DFSMSrmm report utility, EDGRPTD, and create a report that shows the new volume movements required.

DFSMSrmm prevents expiration processing from releasing volumes that have been updated since the last run of vital records processing. You normally need a minimum of two EDGHSKP runs to process an expired volume and return it to scratch, which involves at least a 24 hour delay if you are running expiration processing daily. During this time, you can reclaim an expired volume from pending release status by setting a new expiration date. You can also reclaim an expired volume from scratch status by using the RMM CHANGEVOLUME subcommand to change the volume status to master or user. When you use DFSMSrmm to reclaim a volume that resides in a system-managed tape library to master or user status, the volume is changed to PRIVATE in the TCDB.

Inventory Management Considerations

You can perform some inventory management activities only under specific conditions:

- In an environment where the DFSMSrmm control data set is shared, and you use system-managed tape, you must run inventory management on a system that has access to the TCDB. If the TCDB is not shared by each system that is sharing the control data set, and there are multiple TCDBs, run inventory management at least once against each TCDB on a system that has access to that TCDB.
- When either of the following conditions exist, you must run inventory management on a system that has access to the catalogs.
  - The DFSMSrmm control data set is shared, the DFSMSrmm control data set and catalogs are not synchronized, and you want to retain data sets that are based on the catalog information.

  - The DFSMSrmm parmlib OPTION command UNCATALOG is coded with Y or S to indicate that catalog processing is required.
- In an unshared user catalog environment, you must specify the DFSMSrmm EDGRMM*xx* parmlib OPTION command CATSYSID operand to use DFSMSrmm. When you use unshared catalogs, ensure that the DFSMSrmm control data set and the user catalogs are synchronized. You must run inventory management expiration processing on each different catalog environment.
- When DFSMSrmm is active, you cannot reorganize the control data set.
- The DFSMSrmm subsystem address space performs all inventory management functions, except backing up the control data set and backing up the journal. During inventory management processing, the address space running EDGHSKP waits until processing completes. Once the subsystem processing completes, EDGHSKP performs backup processing in its home address space; this can include the backup of the DFSMSrmm control data set and the journal.
- When inventory management functions are active, you can continue using RMM TSO subcommands and DFSMSrmm continues to record tape usage.
- You must define at least one vital record specification in order to run DFSMSrmm vital record processing. The EDGRMMxx parmlib OPTION command VRSMIN operand provides control over this processing.

  Example: Use the RMM ADDVRS subcommand to define a vital record specification.

  ```
  RMM ADDVRS DSN('**') WHILECATALOG
  ```

DFSMSrmm Inventory Management Considerations when Client/Server Support is Enabled

This topic describes utilities that provide restricted functions when run on a client or server system.

DFSMSrmm Utility	Considerations	Where to Find More Information
EDGHSKP	• You can run all inventory management functions except BACKUP on a client system. However, because there is no direct connection to the control data set, the elapsed times will be longer when run on a client system. We recommend you run all inventory management functions other than CATSYNCH and EXPROC on either the server or a standard DFSMSrmm system. • When you attempt backup on a client system, the utility ends with RC16 and DFSMSrmm issues message EDG6137I stating that it cannot be run on a client system	Chapter 14, “Performing Inventory Management,” on page 275

|
Allocating Data Sets for Inventory Management

DFSMSrmm Samples Provided in SAMPLIB

- EDGJHKPA Sample JCL for Allocating the Data Sets Required for Inventory Management
- EDGPHKPA Sample OPC for Allocating the Data Sets Required for Inventory Management

Before running EDGHSKP, you must define several data sets. The data sets that are used by both the EDGHSKP utility and the DFSMSrmm subsystem address space must be pre-allocated and cataloged as shown in Table 46. When enhanced data integrity function (EDI) is activated, you must include the pre-allocated, cataloged data sets associated with DFSMSrmm EDGHSKP utility processing in the parmlib member IFGPSEDI. The data sets that you must include are the ACTIVITY, MESSAGE, REPORT, REPTEXT, and XREPTEXT data sets.

Table 46. DFSMSrmm EDGHKSP Data Sets

DD Statement	Preallocated, Cataloged, DASD Data Set	Description
ACTIVITY	Yes	Contains detailed information about data set related changes DFSMSrmm makes to the control data set during inventory management. This data set is required when you specify the VERIFY parameter.
BACKUP	No	Contains the backup copy of the DFSMSrmm control data set. Specify this data set to run backup processing for the control data set. You can back up directly to tape when you specify the BACKUP(DSS) parameter even when DFSMSdss concurrent copy is not available.
DSSOPT	No	Contains DUMP or RESTORE command options used by DFSMSdss during backup processing. See “Customizing the DSSOPT DD Statement” on page 323 for information about changing the commands.
JRNLBKUP	No	Contains the backup copy of the DFSMSrmm journal. Specify this data set to run backup processing for the journal. DFSMSrmm uses IDCAMS to back up the journal when you specify the BACKUP(AMS) or BACKUP(DSS) parameter. You can back up directly to tape when you specify the BACKUP(DSS) parameter even when DFSMSdss concurrent copy is not available.
MESSAGE	Yes	Lists the messages the DFSMSrmm subsystem issues during inventory management. This data set is required.
REPORT	Yes	Contains a detailed report of DFSMSrmm vital record specification processing. Specify if you want a report when you have specified the VRSEL parameter.
REPTEXT	Yes	Contains the extract copy of the DFSMSrmm control data set. The extract copy is called the extract data set. You must specify either the REPTEXT DD statement or the XREPTEXT DD statement when you use the EDGHSKP RPTEXT parameter.
SYSPRINT	No	Contains the utility program messages that IDCAMS and ADRDSSU issue when backing up the DFSMSrmm control data set. The SYSPRINT data set is required when you specify the BACKUP parameter. This data set can be a SYSOUT file.

Table 46. DFSMSrmm EDGHKSP Data Sets (continued)

DD Statement	Preallocated, Cataloged, DASD Data Set	Description
XREPTEXT	Yes	Contains the extract copy of the DFSMSrmm control data set. The extract copy is called the extract data set. You must specify either the REPTEXT DD statement or the XREPTEXT DD statement when you use the EDGHSKP RPTEXT parameter.

To avoid enqueue contention consider the following conditions:

- These data sets cannot be created in the same job as they are used. If you plan to retain multiple versions of these data sets, consider using a subsequent job step to copy the data sets to a new GDG generation.
- The JCL used to run EDGHSKP should specify DISP=SHR for these data sets.

  Recommendation: For your regularly scheduled inventory management, use a data set to serialize the jobs that run EDGHSKP to ensure that only one copy of EDGHSKP is running on a system. You can run EDGHSKP with RPTEXT, EXPROC, or CATSYNCH options at the same time on different systems. You can also run EDGHSKP with the RPTEXT option at the same time on the same system. To run each of these tasks in parallel, for example, to run EXPROC on each system in a complex at the same time, use a different preallocated and cataloged data set for each job instead of including the ENQDSN DD statement.

  To avoid problems that might occur when multiple EDGHSKP jobs are running together, add a DD statement to the job step that runs EDGHSKP as shown in Figure 103:

```
//ENQDSN   DD DISP=OLD,DSN=RMM.HSKP.ENQ
```

Figure 103. JCL for Adding a DD Statement to the EDGHSKP Job Step

Alternatively you can use the sample RMM OPC application which uses OPC special resources to prevent multiple jobs running together as described in Chapter 20, “Running DFSMSrmm with the IBM Tivoli Workload Scheduler for z/OS,” on page 395.

To protect the data sets that must be preallocated, cataloged, and on DASD, ensure that the RACF user ID associated with the DFSMSrmm subsystem has authority to write to the data sets.

Recommendation: Always run DFSMSrmm with the optional files ACTIVITY and REPORT. Maintain these files, as well as the MESSAGE, BACKUP, JRNLBKUP, SYSPRINT, REPTEXT, and XREPTEXT files, as GDGs or copy them to GDGs. You might find that multiple versions of the files are helpful for use in diagnosing problems or for creating reports.

Creating an Extract Data Set

You can create an extract data set to use as input for creating reports. You can create two types of extract data sets. They both contain information from the DFSMSrmm control data set. The difference in the extract data sets is that the extract created when you specify the XREPTEXT DD statement, contains additional extended records that are a combination of data set information and volume information that can be useful as input to DFSMSrmm reporting tools.

DFSMSrmm reads sequentially through its control data set, and creates a record in the extract data set for each shelf location, volume, data set, software product, owner and vital record specification record. DFSMSrmm can convert information like the date information into a format you specify. The extract data set is a point-in-time copy of the control data set contents. Use the RMM TSO SEARCH and LIST subcommands to obtain the most current information.

You can use the extract data set as input to any of the following programs:
- DFSMSrmm Report Generator
- DFSMSrmm report utility EDGRPTD

You can use the extract data set containing extended records as input to any of the following programs:
- DFSMSrmm Report Generator
- DFSMSrmm report utility EDGRPTD
- DFSMSrmm-supplied exec EDGRRPTE

Refer to *z/OS DFSMSrmm Reporting* for information about using the DFSMSrmm Report Generator, EDGRPTD, and EDGRRPTE.

Calculating DASD Space and Placement for the Extract Data Set

Table 47 helps you calculate DASD space requirements for the extract data set specified by the REPTEXT DD statement or the XREPTEXT DD statement.

Table 47. DFSMSrmm Extract Data Set DASD Space Requirements

Extract Data Set Content	DASD Space
Data sets	• 475 KB for every 1000 data sets when REPTEXT DD is specified. • 1835 KB for every 1000 data sets when XREPTEXT DD is specified.
Shelf locations in the library that do not contain volumes	11 KB for every 1000 shelf locations
Shelf locations in storage locations	11 KB for every 1000 shelf locations
Owners	35 KB for every 1000 volumes
Software products	17 KB for every 1000 software products
Volumes	796 KB for every 1000 volumes
Vital record specification	20 KB for every 1000 vital record specifications

Convert the final figure into a space allocation. The extract data set has a record format of variable-length blocked records. Choose either a system-determined block size, or a block size suitable for the device your installation uses.

To calculate the space required:
1. Divide the total KB of space by 4. This calculation gives you the number to use in the space allocation.
2. Use a fraction of this total space as secondary space if you want to have an extract data set that extends to multiple extents.

You can use JCL as shown in Figure 104 to create the extract data set prior to running EDGHSKP.

```
//REPTEXT DD DISP=(NEW,CATLG),DSN=RMM.EXTRACT.DSET,
//        UNIT=SYSDA,AVGREC=U,SPACE=(4096,(pp,ss))
```

Figure 104. JCL for Specifying the Extract Data Set

where:

pp Specifies the calculated primary space.

ss Specifies a fraction used for secondary space.

Placing the Extract Data Set

You can place the extract data set on any volume.

JCL for Creating an Extract Data Set

To create an extract data set, specify the RPTEXT parameter. You can submit a job with JCL as shown in Figure 105. To create an extract data set that contains extended records, uncomment the XREPTEXT DD statement. When you specify both XREPTEXT and REPTEXT, DFSMSrmm uses the XREPTEXT DD statement.

```
//HSKP     EXEC PGM=EDGHSKP,
//     PARM='RPTEXT,DATEFORM(E)'
//MESSAGE  DD DISP=SHR,DSN=HSKP.MESSAGES
//REPTEXT  DD DISP=SHR,DSN=MASTER.EXTRACT
//*XREPTEXT DD DISP=SHR,DSN=MASTER.EXTENDED.EXTRACT
```

Figure 105. JCL for Creating an Extract Data Set

Use the DATEFORM parameter to specify the format for date fields in the report extract data set. The DATEFORM parameter can take any of the following values:

Value	Language	Format	Example
A	American	mm/dd/yyyy	12/15/1994
E	European	dd/mm/yyyy	15/12/1994
I	ISO	yyyy/mm/dd	1994/12/15
J	Julian	yyyy/ddd	1994/349
D	Default	Installation’s default in EDGRMMxx	Initially set to Julian

DFSMSrmm provides the format of the records in the extract data set in mapping macros. See *z/OS DFSMSrmm Reporting* for layouts of the macros. You can use DFSORT to sort the extract data set records to create many types of reports.

For example, you could select the extract records that show volumes with temporary-read errors. Sort the resulting list by descending number of errors. Use this list to identify the damaged volumes that should be replaced. Then, you could use the RMM CHANGEVOLUME subcommand with the RELEASEACTION(REPLACE) operand to update DFSMSrmm with the required action.

JCL for EDGHSKP

Figure 106 on page 283 shows example JCL for EDGHSKP.

```
//HSKP EXEC PGM=EDGHSKP,
// PARM='VRSEL,EXPROC,DSTORE,BACKUP(DSS),RPTEXT,DATEFORM(E)'
//SYSPRINT DD SYSOUT=A
//DSSOPT DD *
CONCURRENT OPTIMIZE(1) VALIDATE
/*
//BACKUP DD DSN=BACKUP.FILE.NAME,DISP=(NEW,CATLG),
// UNIT=SYSDA,AVGREC=U,SPACE=(4096,(1000,500)),
// LRECL=9000,BLKSIZE=0,RECFM=U
//JRNLBKUP DD DSN=JOURNAL.BACKUP.FILE.NAME,DISP=(NEW,CATLG),
// UNIT=SYSDA,AVGREC=U,SPACE=(4096,(1000,500)),
// LRECL=9000,BLKSIZE=0,RECFM=VB
//MESSAGE DD DSN=MESSAGE.FILE.NAME,DISP=SHR
//REPORT DD DSN=REPORT.FILE.NAME,DISP=SHR
//ACTIVITY DD DSN=ACTIVITY.FILE.NAME,DISP=SHR
//*REPTEXT DD DSN=REPORT.EXTRACT.FILE.NAME,DISP=SHR
//XREPTEXT DD DSN=REPORT.EXTENDED.EXTRACT.FILE.NAME,DISP=SHR
```

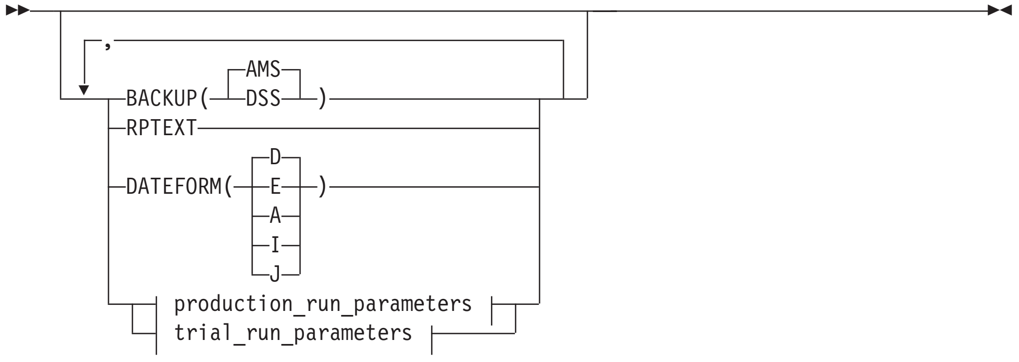
Figure 106. Example of JCL for EDGHSKP

EXEC Parameters for EDGHSKP

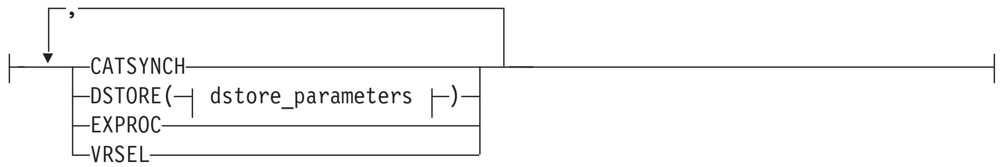
Figure 107 on page 284 shows the EXEC parameters for EDGHSKP. DFSMSrmm uses all the parameters except CATSYNCH, DATE, and VERIFY if you do not specify PARM on the EXEC statement.

You can specify the EXEC parameters in any order, but DFSMSrmm processes them in the following sequence:

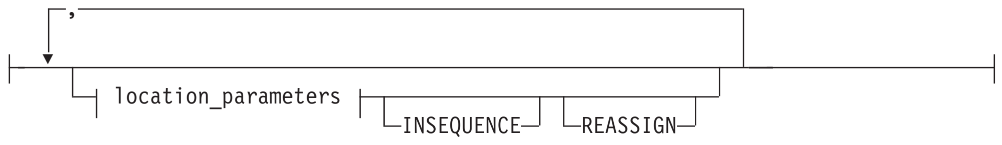
1. CATSYNCH processing
2. VRSEL vital record processing
3. DSTORE storage location management processing
4. EXPROC expiration processing
5. RPTEXT extract data set creation
6. BACKUP control data set and journal back up processing



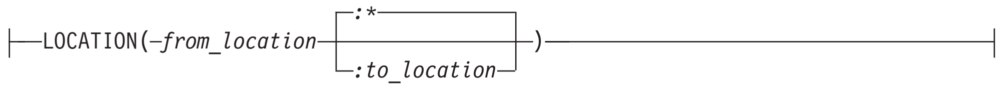
production_run_parameters:



dstore_parameters:



location_parameters:



trial_run_parameters:

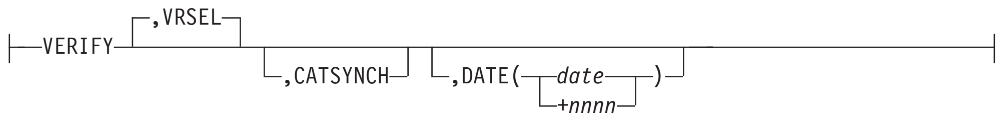


Figure 107. EDGHSKP EXEC Parameters

The parameters on the EXEC statement are:

BACKUP(AMS|DSS)
: Specify BACKUP to back up the control data set, to back up the journal or to back up both the control data set and the journal. When you specify BACKUP on the EXEC statement, you must also specify a DD statement. Specify BACKUP DD when you back up the control data set and JRNLBKUP DD when you back up the journal. You can also specify both DD statements in your EDGHSKP JCL.

BACKUP(AMS) is the default. DFSMSrmm uses the access method services REPRO command to perform backup processing. Specify BACKUP(AMS) to prevent DFSMSrmm from updating the control data set during control data set backup processing. Backup cannot be directly to tape.

Specify BACKUP(DSS) with the DFSMSdss concurrent copy environment set up to permit the update of the DFSMSrmm control data set during backup processing. This ensures that all tape activities can continue during backup processing. You can specify BACKUP(DSS) without setting up the concurrent copy environment. If you specify BACKUP(DSS), backup can be direct to tape.

For journal backup, DFSMSrmm uses IDCAMS to back up the journal even when BACKUP(DSS) is specified. When using EDGHSKP to perform back up, the journal can be cleared. Refer to “Steps for Automating Control Data Set Backup and Journal Clearing” on page 314 for information.

CATSYNCH

Specify CATSYNCH to update the DFSMSrmm control data set with information from available user catalogs. Synchronizing the DFSMSrmm control data set with the catalogs is normally a one-time setup exercise. See “Running DFSMSrmm Catalog Synchronization” on page 305 for implementation details. Before you can synchronize the DFSMSrmm control data set with user catalogs, define system IDs by using the EDGRMM*xx* parmlib OPTION CATSYSID operand as described in “Defining System Options: OPTION” on page 134.

Specify the EDGRMM*xx* parmlib OPTION CATSYSID(*) to indicate that catalogs are fully shared and that any data set can be processed by DFSMSrmm on any DFSMSrmm system. DFSMSrmm checks that the control data set has been synchronized prior to performing vital record processing or expiration processing. DFSMSrmm dynamically adds the CATSYNCH execution option when the control data set and catalogs are not synchronized and the EDGHSKP VRSEL or EXPROC parameters are specified.

If CATSYSID is specified with specific system IDs, you cannot run vital record processing until the control data set is synchronized with all user catalogs and you have run EDGUTIL with the SYSIN statement CONTROL CATSYNCH(YES). See “Creating or Updating the Control Data Set Control Record” on page 341 for information about marking the control data set for synchronization.

Prior to synchronizing the DFSMSrmm control data set with available user catalogs, you can specify the VERIFY parameter with the CATSYNCH parameter to find differences between the DFSMSrmm control data set and the user catalogs. DFSMSrmm reports all the differences so you can make updates, if required, to catalog information before running CATSYNCH without the VERIFY parameter.

When you run the CATSYNCH parameter with the VERIFY parameter on a DFSMSrmm control data set that is already synchronized, and there are differences found between the catalog and the DFSMSrmm information, the DFSMSrmm control data set is marked as not synchronized. The next time you run the CATSYNCH parameter without the VERIFY parameter or when CATSYNCH is added automatically by EDGHSKP, the DFSMSrmm control data set and catalogs are synchronized again.

DFSMSrmm adds information about the data sets that are synchronized in the ACTIVITY report. It is expected that CATSYNCH is setup for unshared catalogs in a client server environment.

DATE(*date*|*+nnnn*)
Specify DATE to set the date used for VERIFY processing.

To specify a date with *date*, supply the year and day in one of two forms:

- yyddd, where yy is the last two-digit number for the year and ddd is the three-digit number for the day of the year, such as 93001.
- yyyy/ddd, where yyyy is the four-digit number for the year and ddd is the three-digit number for the day of the year, such as 1993/001. The slash is required.

For dates in the year 2000 and or in the 21st century or higher, you can only use the yyyy/ddd format. If you use the yyddd format, DFSMSrmm defaults to the 20th century.

To specify a number of days to add to the current date, supply *+nnnn* to determine the actual date to be used for VERIFY processing. You specify the value as a plus sign and the number of days, for example, to use the date in 7 days time specify DATE(+7).

The current date is the default.

DATEFORM (A|E|I|J|D)
Specify DATEFORM to set the date format for records that are written to the extract data set, records written to the ACTIVITY file, and any messages issued during inventory management.

Value	Language	Format	Example
A	American	mm/dd/yyyy	12/15/1994
E	European	dd/mm/yyyy	15/12/1994
I	ISO	yyyy/mm/dd	1994/12/15
J	Julian	yyyy/ddd	1994/349
D	Default	Installation’s default in EDGRMMxx	Initially set to Julian

The default date format for all date fields is the value specified in the parmlib member EDGRMMxx. The value is initially set to J for Julian. To change the date format for each run of EDGHSKP, use the DATEFORM parameter.

DSTORE
Specify DSTORE for storage location management processing.

LOCATION(*from_location:to_location, ...*)
Specify this parameter if you want to perform storage location management processing by location.

Specify a pair of location names separated by a colon. You can specify 1 to 8 pairs of *from_location:to_location* names.

If you omit this parameter, DFSMSrmm performs storage location management processing for all volumes that are required to move.

The *from_location* is the name of the location from which the volume should move. The *to_location* is the name of the destination for the volume.

These location names can be specified in one of the following ways:

- Specify a specific location using 1 to 8 character names.
- Specify all locations using a single asterisk (*).

- Specify all locations that begin or end with specific characters, such as ATL* or *DR.
- Use the percent sign % in the location name to replace a single character. You can specify up to eight % in a location name mask.

DFSMSrmm does not validate the location names that you specify against the DFSMSrmm LOCDEF entries or the names of SMS libraries.

INSEQUENCE

Specify INSEQUENCE for volumes that are required to move from a non-bin-managed location to a bin-managed location.

Storage location management processing assigns volumes to available bins in volume sequence and bin sequence, starting with the lowest volume serial number and the lowest bin number.

All bins that become available in a single run of storage location management processing can be reused for other volumes. Bins can become available during storage location management processing under these conditions:

- Global confirm move processing.
- Volumes starting a move out of the bin with parmlib option REUSEBIN(STARTMOVE) specified.

If REASSIGN is also specified, the volumes that restart their move are merged in sequence with those volumes that have just started their move. Freed bins are merged with empty bins.

Bins are best utilized, if INSEQUENCE and REASSIGN are specified and parmlib option REUSEBIN(STARTMOVE) is also specified. Fewer empty bins need to be defined.

REASSIGN

Specify REASSIGN for volumes that are already moving from a storage location that is not bin-managed storage location and the required location is either a bin-managed storage location or is different from the destination. When you specify REASSIGN, you are canceling the move for these volumes and requesting that the move for the volumes is restarted so that DFSMSrmm could assign these volumes to other locations or bins.

If the LOCATION parameter is specified, DFSMSrmm reassigns a volume when at least one of the LOCATION subparameter pairs matches the volume’s current location name and destination name.

EXPROC

Specify EXPROC for expiration processing.

RPTEXT

Specify RPTEXT to create an extract data set. When you specify RPTEXT as the only execution parameter you enable DFSMSrmm to create your extract at the same time that other RPTEXT or inventory management requests are being processed. When there are multiple requests, you must provide a MESSAGE file and a REPTEXT file or an XREPTEXT file for each extract request.

VERIFY

Specify VERIFY to request that DFSMSrmm performs a trial run of selected processing. You can run a trial run for vital record processing and catalog synchronization. When you specify VERIFY, DFSMSrmm performs the requested processing, but does not update the DFSMSrmm control data set as a result of the processing except for CATSYNCH with VERIFY where differences between the catalogs and the DFSMSrmm control data set are found. The DFSMSrmm control data set is updated to show the catalogs and CDS are not synchronized.

You could specify VERIFY to perform a trial run to test new vital record specifications that you define to DFSMSrmm. Run VERIFY to confirm that your new and changed retention and movement policies achieve the expected results. If all you want to perform is a trial run of vital record processing, you can specify VERIFY with or without the VRSEL parameter. You can use the DATE parameter with VERIFY to set a date to perform VERIFY processing. When you specify VERIFY, you can use all the inventory management parameters except DSTORE and EXPROC. The ACTIVITY file is required when you select the VERIFY option.

The DFSMSrmm parmlib OPTION VRSCHANGE operand default of VERIFY prevents inventory management vital record processing when there are policy changes that have not been tested. You only need one successful run of VERIFY before you can continue with inventory management processing. If VERIFY completes successfully, but you do not obtain the vital record specification results you expected, you must continue to modify policies and run VERIFY until you obtain acceptable results.

VRSEL
Specify this parameter for vital record processing. To perform a trial run of vital record processing, you can also specify the VERIFY parameter and optionally the DATE parameter. Performing a trial run allows you to see how DFSMSrmm vital record processing changes affect data set and volume information before the control data set is updated.

Running Vital Record Processing

Vital records processing identifies the volumes that should be retained and how they show be moved based on the vital record specifications you define. Use the EDGRMMxx parmlib OPTION VRSMIN operand to specify the minimum number of vital record specifications that must be defined in order to run inventory management vital record processing. The default number of vital record specifications is 1. You can also set other parmlib OPTION operands to control how DFSMSrmm performs vital record processing as described in “Defining System Options: OPTION” on page 134. You can retain data sets and volumes in several ways. For example, you can retain volumes by generic volume serial number, data sets for as long as they are cataloged, or by the job name that created the data set. For more information about moving and retaining volumes with vital record specifications, see *z/OS DFSMSrmm Guide and Reference*.

DFSMSrmm can create a report describing data sets and volumes being retained and the location where the data set or volume resides. See Figure 110 on page 290 for an example of this report.

You can also request an ACTIVITY file as described in “Using the Inventory Management ACTIVITY File” on page 294. DFSMSrmm can write details to the ACTIVITY file about changes to data set information made during vital record processing. The ACTIVITY file is optional except when the VERIFY EXEC parameter is used to request that DFSMSrmm performs a trial run of vital record processing.

JCL for Vital Record Processing

To request vital record processing, specify the VRSEL parameter. You can submit a job with JCL as shown in Figure 108:

```
//HSKP     EXEC PGM=EDGHSKP,
//     PARM='VRSEL'
//MESSAGE  DD DISP=SHR,DSN=HSKP.MESSAGES
//REPORT   DD DISP=SHR,DSN=HSKP.VRS.REPORT
//ACTIVITY DD DISP=SHR,DSN=HSKP.ACTIVITY
```

Figure 108. Example of JCL for Vital Record Specification Processing

To run a trial run of vital record processing, you can submit a job with JCL as shown in Figure 109.

```
//HSKP     EXEC PGM=EDGHSKP,
//     PARM='VRSEL,VERIFY'
//MESSAGE  DD DISP=SHR,DSN=HSKP.MESSAGES
//REPORT   DD DISP=SHR,DSN=HSKP.VRS.REPORT
//ACTIVITY DD DISP=SHR,DSN=HSKP.ACTIVITY
```

Figure 109. Example of JCL for Trial Run Vital Record Specification Processing

You can request that DFSMSrmm create a REPORT file shown in “Using the Vital Records Retention Report.” DFSMSrmm creates a report that contains data sets and volumes being retained, the vital record specification that is retaining the data set or volume, and the required location for the data set or volume.

You can specify the LRECL for the REPORT file. The LRECL can be a value from 133 to 255. If you specify an LRECL that is 148 or higher, only a single line is displayed for a retained data set. If you specify a value less than 148, DFSMSrmm uses two lines to display data set information. The page, date, and time fields in the report are right aligned assuming a record length of 133 characters.

The minimum LRECL must include one byte for the ASA print control character. If you use variable length record format for the REPORT file, then you must add four bytes to the minimum record length to hold the descriptor word. This means that if your REPORT file contains variable length records with a data length of 132 bytes, then you must define the minimum LRECL as 137 bytes.

You can also request an ACTIVITY file as described in “Using the Inventory Management ACTIVITY File” on page 294. The ACTIVITY file is optional except when the VERIFY EXEC parameter is used to request that DFSMSrmm performs a trial run of vital record processing.

Using the Vital Records Retention Report

Use the Vital Records Retention Report to check the vital record specifications that match to data sets and to see which versions of the data sets are being retained. The report file lists the required location for each data set and volume. Use the EDGRPTD movement report to identify the destination selected for each volume after VRSEL processing is completed and DSTORE and RPTEXT have been run.

Figure 110 on page 290 shows a sample of the report produced by vital record processing.

```
REMOVABLE MEDIA MANAGER                                   VITAL RECORDS RETENTION REPORT                                PAGE         1
         (C) IBM CORPORATION 1993 1998                    ----- ------- --------- ------                    TIME 11:21:02 DATE   1998/085


JOB MASK  DATA SET OR VOLUME MASK                       OWNER    TYPE RETN   C X   DELETE   DLY COUNT STNUM LOCATION RLSE
_________________________________________________________________________________________________________________________
          STSGMA.AND.**                                 STSGMA   DSN  BYDAYC N N   1999/365   0     3     3 HOME     XI SI
          WHILEC                                        STSGMA   AND  CYCLES Y N   1999/365     99999

JOB NAME  DATA SET NAME      2ndVRS   2ndNAME  FSEQ DSEQ VOLUME VSEQ OWNER    CURRENT  REQUIRED PRTY  RETDATE   RETNAME
_______________________________________________________________________________________________________________________
          STSGMA.AND.G0002V00                     1    0 AND002    1 STSGMA   SHELF    SHELF    5000 CYCL/00003  *
NUMBER OF DATA SETS RETAINED (GROUP STORE) =                                                        1     0


JOB MASK  DATA SET OR VOLUME MASK                       OWNER    TYPE RETN   C X   DELETE   DLY COUNT STNUM LOCATION RLSE
_________________________________________________________________________________________________________________________
          STSGMA.ORR.**                                 STSGMA   DSN  BYDAYC N N   1999/365   0     3     3 HOME     SI
          WHILEC                                        STSGMA   NEXT CYCLES Y N   1999/365     99999 99999 HOME

JOB NAME  DATA SET NAME      2ndVRS   2ndNAME  FSEQ DSEQ VOLUME VSEQ OWNER    CURRENT  REQUIRED PRTY  RETDATE   RETNAME
_______________________________________________________________________________________________________________________
          STSGMA.ORR.G0009V00                     1    0 ORR009    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.G0008V00                     1    0 ORR008    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.G0007V00                     1    0 ORR007    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.G0006V00                     1    0 ORR006    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.G0005V00                     1    0 ORR005    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.G0004V00                     1    0 ORR004    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.G0002V00                     1    0 ORR002    1 STSGMA   SHELF    SHELF    5000 WHILECATLG WHILEC
NUMBER OF DATA SETS RETAINED (GROUP STORE) =                                                        7     0

          STSGMA.ORR.AAAAAAAA.BBBBBBBB.G0009V00
                                                  1    0 ORR009    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.AAAAAAAA.BBBBBBBB.G0008V00
                                                  1    0 ORR008    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.AAAAAAAA.BBBBBBBB.G0007V00
                                                  1    0 ORR007    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.AAAAAAAA.BBBBBBBB.G0006V00
                                                  1    0 ORR006    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.AAAAAAAA.BBBBBBBB.G0005V00
                                                  1    0 ORR005    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.AAAAAAAA.BBBBBBBB.G0004V00
                                                  1    0 ORR004    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.ORR.AAAAAAAA.BBBBBBBB.G0002V00
                                                  1    0 ORR002    1 STSGMA   SHELF    SHELF    5000 WHILECATLG WHILEC
NUMBER OF DATA SETS RETAINED (GROUP STORE) =                                                        7     0
```

Figure 110. Sample Vital Records Retention Report (Part 1 of 2)

```
JOB MASK  DATA SET OR VOLUME MASK                       OWNER    TYPE RETN   C X   DELETE   DLY COUNT STNUM LOCATION RLSE
_________________________________________________________________________________________________________________________
          STSGMA.MVO.**                                 STSGMA   DSN  BYDAYC N N   1999/365   0     3     3 HOME     XI

JOB NAME  DATA SET NAME      2ndVRS   2ndNAME  FSEQ DSEQ VOLUME VSEQ OWNER    CURRENT  REQUIRED PRTY  RETDATE   RETNAME
_______________________________________________________________________________________________________________________
          STSGMA.MVO.G0009V00
                             WHILEC               1    0 ORR009    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.MVO.G0008V00
                             WHILEC               1    0 ORR008    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.MVO.G0007V00
                             WHILEC               1    0 ORR007    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.MVO.G0006V00
                             WHILEC               1    0 ORR006    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.MVO.G0005V00
                             WHILEC               1    0 ORR005    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.MVO.G0004V00
                             WHILEC               1    0 ORR004    1 STSGMA   SHELF    SHELF    5000 CYCL/00003 *
          STSGMA.MVO.G0002V00
                             WHILEC   WHILEC      1    0 ORR002    1 STSGMA   SHELF    SHELF    5000 WHILECATLG
NUMBER OF DATA SETS RETAINED (GROUP STORE) =                                                        7     0
```

Figure 110. Sample Vital Records Retention Report (Part 2 of 2)

The data columns in the vital records retention report provide the following information:

JOB MASK
The job name mask defined for the vital record specification.

DATA SET OR VOLUME MASK
The data set name mask or generic volume serial number defined for the vital record specification. In addition to the data set name mask or the volume serial number, this field can contain descriptive text about a vital record specification chain.

OWNER
Owner ID of the vital record specification owner.

TYPE
Type of vital record specification: one of AND, DSN, GDG, NEXT, PGDG, or VOL.

- AND designates a NAME vital record specification which is pointed to by a vital record specification with the ANDVRS operand.
- DSN for data set vital record specification.
- GDG designates a data set vital record specification with the GDG operand specified.
- NEXT designates a NAME vital record specification which is pointed to by a vital record specification with the NEXTVRS operand.
- PGDG designates a data set vital record specification with a pseudo-generation data set name mask.
- VOL for volume vital record specification.

RETN
Type of retention for data set vital record specifications and name vital record specifications:

BYDAYC
Retention type is BYDAYSCYCLE.

CYCLES
Retention type is CYCLES.

DAYS
Retention type is DAYS since creation.

LRDAYS
Retention type is LASTREFERENCEDAYS.

XDAYS
Retention type is EXTRADAYS.

C Y if the WHILECATALOG option applies. N if the WHILECATALOG option does not apply.

X Y if the UNTILEXPIRED option applies. N if the UNTILEXPIRED option does not apply.

DELETE
The date DFSMSrmm deletes the vital record specification.

DLY
The number of days that DFSMSrmm delays the latest copy of a data set or volume before sending it to the named location

COUNT
Total number of retention days or cycles for a data set, total number of retention days for a single volume, total number of volumes for generic volume. The number of days, cycles, or volumes is listed for each vital record specification in the chain that contains retention information. The report lists the total number of data sets retained in the current group at the end of each group of retained data sets.

STNUM
Lists the vital record specification store number value. The store number value is the number of days to retain a data set or volume, the number of data set cycles to retain, or the number of volumes to retain. STNUM is blank for AND vital record specifications. The report lists the total number of data sets in the current group that are requested to be retained outside of their home location.

LOCATION
The location in which the data set or volume should be retained. LOCATION is blank for AND vital record specifications.

RLSE
Lists the vital record specification release options in the data set vital record specification. RLSE can be IX SI, IX, or SI. IX is EXPIRYDATEIGNORE and SI is SCRATCHIMMEDIATE.

JOB NAME
The job that created the data set.

DATA SET NAME OR VOLUME MASK
Data set name. For data set names that exceed 30 characters and that match to a primary vital record specification and a secondary vital record specification, DFSMSrmm splits the output line to display all the data set information.

2ndVRS
Lists the management value vital record specification or management class vital record specification when a data set matches to both a primary and secondary vital record specification. If the management class vital record specification or management value vital record specification is currently retaining the data set, DFSMSrmm puts the name of the vital record specification that is retaining the data set in the 2nd NAME field.

2ndNAME
Lists the name of the first vital record specification in the secondary vital record specification chain that retains the data set. This field is always filled when a data set is retained by a secondary vital record specification. When vital record specifications are chained using AND, DFSMSrmm puts the name of the first vital record specification in this field. This field is blank when a data set is retained by a primary vital record specification.

FSEQ
Physical file sequence number.

DSEQ
Data set sequence number on the named volume.

VOLUME
Volume serial number.

VSEQ
Volume sequence number.

OWNER
Owner ID of the volume and data set owner.

CURRENT
Current location of the volume.

REQUIRED
Destination of the volume based on the matching vital record specification. When there are multiple data sets on a volume, DFSMSrmm displays the destination for the volume using the priority of each of the required locations identified for the volume. The volume's destination is calculated based on the location priorities described in Table 48 on page 299. If there are multiple data sets on a volume or the volume is retained using multiple vital record specifications, the required location listed in the report might be different from the destination to which DFSMSrmm attempts to move the volume.

When the RETDATE is CATRETPD, this field contains the volume's current location.

PRTY
Defined or defaulted relative priority of the location.

RETDATE
Lists information for individual data sets and volumes. This field contains the retention date calculated by vital record processing. The meaning of the retention date depends on the retention type of the vital record specification. See *z/OS DFSMSrmm Guide and Reference* for information about how DFSMSrmm calculates the retention date.

BYDAYSCYCLE
For the BYDAYSCYCLE retention type, RETDATE is displayed as a special date format CYCL/*ccccc*. *ccccc* is the COUNT value for cycles used in the vital record specification.

CYCLES
For the CYCLES retention type, RETDATE is displayed as a special cycles date format, CYCL/*ccccc*, where *ccccc* is the COUNT value for cycles used in the vital record specification.

DAYS
For the DAYS retention type, RETDATE is displayed as the date calculated by vital record selection processing. This is the COUNT value added to the data set creation date or the volume assigned date.

EXTRADAYS
For the EXTRADAYS retention type, RETDATE is a date calculated by inventory management vital record processing. The RETDATE is the COUNT value in the vital record specification chain added to the date when the subchain started to retain the data set.

LASTREFERENCEDAYS
For the LASTREFERENCEDAYS retention type, RETDATE is displayed as the date calculated by the vital record selection processing. This is the COUNT value added to the date the data set or volume was last referenced.

UNTILEXPIRED
For the UNTILEXPIRED retention type, DFSMSrmm calculates the RETDATE based on the presence of a secondary vital record specification. When a secondary vital record specification retains a data set, the UNTILEXPIRED retention is based on the retention type in the secondary vital record specification. The retention type can be any of the date formats specified for CYCLES, DAYS, LASTRERENCEDAYS, WHILECATALOG, BYDAYSCYCLE, and EXTRADAYS. If the

secondary vital record specification contains WHILECATALOG, the retention date is WHILECATLG as long as the data set is cataloged.

WHILECATALOG

Retention date for WHILECATALOG can be the special catalog date format, WHILECATLG or CATRETPD. When the RETDATE is WHILECATLG, the data set is retained by a vital record specification with the WHILECATALOG retention type and the data set is cataloged. CATRETPD is used when the data set is not cataloged and is created within the period defined by the parmlib OPTION CATRETPD operand. DFSMSrmm retains the data set for the catalog retention period if the data set has never been cataloged. DFSMSrmm does not retain the data set if DFSMSrmm detected that the data set was cataloged and then uncataloged during the catalog retention period. DFSMSrmm sets the REQUIRED field to the volume's current location and does not include the data set in the cycle count.

RETNAME

RETNAME displays the name of the current vital record specification in the primary vital record specification chain that retains the data set as follows:

- The first vital record specification in the subchain for ANDVRS groups
- An * if the retaining vital record specification is a data set name vital record specification in a primary vital record specification
- A blank for when a secondary vital record specification retains a data set

NUMBER OF DATA SETS RETAINED (GROUP STORE)

Lists two numbers. The first is the number of data sets or volumes retained by the vital record specification. The second is the number of data sets or volumes retained in a storage location.

Using the Inventory Management ACTIVITY File

DFSMSrmm provides the VERIFY function to perform a trial run of vital record processing and synchronize catalog processing so you can see the results of processing on a production run.

The ACTIVITY file is optional except during VERIFY processing. During VERIFY processing, the ACTIVITY file is required so that you can analyze processing results before they are actually performed. The ACTIVITY file is a pre-allocated DASD data set, like the REPORT file. The ACTIVITY file is a variable blocked file with the record length set to the largest record created by DFSMSrmm. The block size is determined by the system.The ACTIVITY file is not intended to be a report, but to contain detailed information about changes made to data sets during DFSMSrmm processing. The DFSMSrmm supplied sample EDGJHKPA shows the JCL to allocate the ACTIVITY file.

DFSMSrmm writes an activity record for data set changes only when a change is identified in the ACTD_CHANGE section of the record. During processing if an ACTIVITY file is allocated, DFSMSrmm writes information about changes in the data set information, such as matching vital record specification, vital record status, retention date and catalog status to the ACTIVITY file. If VERIFY processing is being run, the changes are not actually made.

You can view the ACTIVITY file on-line. To print the ACTIVITY file, use a product like DFSORT or DFSORT ICETOOL to selectively format and print fields. DFSMSrmm provides a sample job EDGJACTP in SAMPLIB to print the ACTIVITY report. See *z/OS DFSMSrmm Reporting* for information about the reports you can produce with the sample job.

How Vital Record Processing Works

This section describes how vital record processing works.

1. You control how vital record processing works by using the EDGRMMxx parmlib OPTION command options CATRETPD, MOVEBY, RETAINBY, VRSJOBNAME, VRSCHANGE, VRSMIN, and VRSEL as described in “Defining System Options: OPTION” on page 134.
2. You must run vital record processing when you add or change a vital record specification for DFSMSrmm to apply the policy defined by the vital record specification. You should reclaim any volumes that are pending release or ready to return to scratch to avoid data loss. Use the RMM CHANGEVOLUME subcommand to change the status of these volumes to reclaim them
3. As part of vital record processing, DFSMSrmm checks that data set name masks and job name masks used in vital record specifications are specified according to DFSMSrmm naming conventions. *z/OS DFSMSrmm Guide and Reference* provides information on defining vital record specifications. DFSMSrmm compares the data set, job name, and volume information recorded in the control data set with information in the vital record specifications to determine which data sets and volumes to retain and the processing required. This includes any volumes with special JCL specified expiration dates used by your installation. See “Managing Volumes with Special Dates” on page 78.
4. DFSMSrmm deletes any vital record specifications that have reached their automatic deletion date from the control data set. When a vital record specification is deleted, no data set or volume information is changed. DFSMSrmm uses only the remaining vital record specifications to apply policies. If the data set or volume matches to another remaining vital record specification, DFSMSrmm applies those policies.
5. DFSMSrmm retains stacked volumes based on how the contained volumes are retained. The stacked volume is retained by a vital record specification if any contained volume is retained on a volume basis or data set basis. DFSMSrmm sets the stacked volume’s retention date to the highest retention date of all contained volumes. DFSMSrmm sets the required location of the stacked volume using location priority.
6. During vital record processing, DFSMSrmm records information about the matching vital record specification name and retention date that DFSMSrmm sets for the data set or volume. If the data set or volume does not match to any vital record specifications, and is no longer retained by a vital record specification, the data sets are eligible for expiration processing.
7. DFSMSrmm saves movement and retention information in each volume record. DFSMSrmm also saves retention information in each data set record.
   - When a data set is retained by a vital record specification, DFSMSrmm sets the data set retention date to the date calculated during vital record processing.
   - When a data set is no longer retained by a vital record specification, DFSMSrmm sets the data set retention date to the date vital record processing started.

- When a volume is pending release or scratch, or is no longer retained by a vital record specification, DFSMSrmm does not change the retention date that was already set.
- DFSMSrmm uses the highest data set retention date on the volume to set the volume retention date.

8. See "Confirming Global Volume Movement" on page 309 for information about the global move confirmation processing that takes place during vital record processing.
9. Any volume no longer retained by a vital record specification is updated to indicate that it is not retained by a vital record specification and the retention date is set to the date that inventory management is run.

How DFSMSrmm Processes Vital Record Specification Chains

DFSMSrmm validates vital record specification chains by checking for the existence of the next link in a vital record specification chain. If the next link in the vital record specification chain is missing, DFSMSrmm issues message EDG2230I.

When the EDGRMMxx parmlib member OPTION VRSEL(NEW) operand is used, you can define policies using individual vital record specifications and vital record specification groups or subchains. During inventory management DFSMSrmm processes vital record specifications based on the parmlib VRSEL operand value you define. When a data set matches to a vital record specification, DFSMSrmm uses the retention information to retain the data set. Once the retention criteria has been met, the next vital record specification in the chain is used, and so on, until the end of the chain is reached. Each time you run vital record processing, DFSMSrmm processes from the start of the chain in case any of the earlier retention criteria apply again.

How DFSMSrmm Processes Primary and Secondary Vital Record Specifications

You can define both a primary and secondary vital record specification to retain a data set. DFSMSrmm matches the data set to the vital record specification based on data set name, jobname, management class, vital record specification management value, the ABEND and OPEN reserved names, or the '**' data set name mask. DFSMSrmm matches the data set to the secondary vital record specification based on management class or vital record specification management value if the primary match is on data set name, but not '**'.

When you use the parmlib member EDGRMMxx OPTION VRSEL(NEW) operand, DFSMSrmm uses the following criteria to retain the data set:

- Retained by the primary data set name vital record specification

  Location is taken from the primary data set name vital record specification. Retention date is calculated based on the values set in the current subchain of the primary vital record specification. Release actions are determined by values that are set in the primary vital record specification.
- Retained by the primary data set name vital record specification including the UNTILEXPIRED retention type in the current subchain

  Location is taken from the primary data set name vital record specification. Retention date is calculated based on the values set in both the primary vital record specification and the secondary vital record specification. The retention date is the earliest date of the two current vital record specifications. Release actions are determined by values that are set in the primary vital record specification.

- Retained by the secondary vital record specification and not the primary vital record specification

  Location is taken from the secondary vital record specification. Retention date is the calculated based on the value set in the secondary vital record specification. Release actions are determined by values that are set in the secondary vital record specification.
- Not retained by any vital record specification

  The data set is not retained as a vital record. Release actions are determined by values that are set the last time the data set was retained as a vital record.

Combining Retention Types

When you use the parmlib OPTION VRSEL(NEW) operand, you can define vital record specifications so that DFSMSrmm retains the data set or volume by combining the retention information from two vital record specifications. When the EDGRMMxx parmlib OPTION VRSEL(NEW) operand is used, DFSMSrmm processes both primary and secondary vital record specifications to retain a data set. DFSMSrmm can combine retention information from the two vital record specifications and apply the combined retention to a data set. Use the UNTILEXPIRED retention type in the primary vital record specifications to combine its retention with the retention in the secondary vital record specification. When UNTILEXPIRED is specified in a primary vital record specification, DFSMSrmm looks at the secondary vital record specification to determine if UNTILEXPIRED is true or not. If DFSMSrmm finds no secondary vital record specification that matches a data set or if UNTILEXPIRED is specified in the secondary vital record specification, then DFSMSrmm uses the volume expiration date to determine if UNTILEXPIRED is true or not.

How DFSMSrmm Selects Retention and Movement Policies

DFSMSrmm retains data sets and volumes using policies you define in vital record specifications by matching data sets and volumes to vital record specifications. DFSMSrmm matches data sets with data set names and job names specified in the vital record specification.

You can define policies for specific data sets and volumes. You can also define system-wide policies for all data sets not covered by a specific policy.

The data set name masks ** and primary vital record specification *.** match to all data sets not covered by a more specific vital record specification. DFSMSrmm matches a vital record specification with the data set name mask *.** to a data set before matching to a vital record specification with a vital record specification management value. DFSMSrmm uses the vital record specification with the data set name mask ** to cover any data sets that do not match to any other vital record specifications. You can use these data set name masks to define a system-wide retention policy.

When you specify the parmlib OPTION VRSEL(NEW), DFSMSrmm matches a data set to a secondary vital record specification when the data set first matches to a vital record specification with a data set name mask *.**.

You can select how DFSMSrmm uses data set name and job name in matching with the parmlib OPTION VRSJOBNAME operand described in "Defining System Options: OPTION" on page 134. *z/OS DFSMSrmm Guide and Reference* provides information on defining vital record specifications and how to retain and move data sets and volumes.

Considerations for Retaining Data Sets and Volumes

Any volumes that are currently open or left open are listed in message EDG2404W. If you do not want to retain those volumes that are left open from processing errors or abends, you can use the RMM DELETEVOLUME *volser* RELEASE command to release those volumes.

You can also define vital record specifications with the reserved data set name mask OPEN to manage volumes that are left open. You can use the reserved data set name mask ABEND in a vital record specification to specify policies for data sets closed as a result of an abnormal end in a task. See *z/OS DFSMSrmm Guide and Reference* for information on defining vital record specifications.

Data sets and volumes that have reached or passed their expiration date and that are not to be retained are subject to normal expiration and release processing. You can control when DFSMSrmm releases volumes by using the parmlib OPTION VRSEL(NEW) operand. After specifying the VRSEL(NEW) operand, you can define vital record specifications that ignore the expiration date so that volumes are released before their expiration date. See "How Expiration Processing Works" on page 303 for more information.

Moving Volumes

During inventory management, DFSMSrmm identifies the destination for a volume. DFSMSrmm uses the location priority value to resolve movement conflicts when a volume is destined for more than one location. Priority values as shown in Table 48 on page 299 are purely relative and do not have any further significance. If more than one destination is identified, and the locations are not the same, the DFSMSrmm default priorities are based on the following:

- If none of the locations are storage locations, DFSMSrmm selects the most automated location that is based on the location priority.
- If one or more storage locations is specified for a single volume, DFSMSrmm stores the volume in the location based on the DFSMSrmm default location priority, the priority set in the LOCDEF command for the location, or the priority specified in a vital record specification. The lower priority numbers take precedence. For example, a volume would move to a location with a priority value 100 before moving to a location with a priority value of 200. You can set a priority value with the PRIORITY operand on the LOCDEF parmlib command to define the relative importance of locations.

  You can define PRIORITY on RMM ADDVRS subcommands to override default or assigned priorities at the data set level. When you do not set a priority value, DFSMSrmm uses the priority shown in Table 48 on page 299.

Table 48. DFSMSrmm Movement Priority Default Values

Priority Number	Location Name or Location Type
100	REMOTE DFSMSrmm built-in storage location name
200	DISTANT DFSMSrmm built-in storage location name
300	LOCAL DFSMSrmm built-in storage location
2000	Installation defined storage locations
4800	AUTO automated tape libraries
4900	MANUAL manual tape libraries
5000	SHELF location name

For stacked volumes, DFSMSrmm determines the required location using the required location of each of the volumes in the same container that is retained by a vital record specification. Conflicts in locations are resolved using movement priority.

Volumes requiring movement are identified for storage location management processing. Storage location management processing sets the destination for the volume and allocates shelf locations, based on the destination specified in the vital record specification. See "How Storage Location Management Processing Works" on page 300 for more information.

You can request that DFSMSrmm retain a volume in its existing location instead of specifying a storage location or the home location. You define the location in a vital record specification by using the CURRENT reserved location name. When you run vital record processing, DFSMSrmm keeps the volume in the current location rather than scheduling the volume for a move.

Inventory Management Trial Run

You can run a trial run of inventory management vital record processing before DFSMSrmm makes the changes. The trial run lets you see how the vital record specifications you defined would be processed in a production run except that DFSMSrmm does not make any changes to the control data set.

1. Allocate the ACTIVITY file that DFSMSrmm uses to record data set changes that would result during inventory management vital record processing.
2. Submit an inventory management job using JCL as shown in Figure 109 on page 289 to request a trial run of vital record processing. Perform step 3, 4, and 5 until you are satisfied that the correct policies are in place.
3. Analyze the ACTIVITY file to see what changes would have occurred based on the vital record specifications you defined.
4. Change vital record specifications by deleting and adding vital record specifications as needed to correct the policies you want in place.
5. Check the EDGRMMxx member OPTION VRSCHANGE operand value. Make sure VRSCHANGE(VERIFY) is specified so that DFSMSrmm issues message EDG2308I if any changes are made since the last run of inventory management.
6. Submit an inventory management job requesting a production run of vital record processing after you have completed making changes to the vital record specifications.

Running Storage Location Management Processing

Request storage location management processing when you are using vital record specifications to identify data sets and volumes to be moved between locations or using RMM TSO CHANGEVOLUME subcommands to move volumes between system-managed libraries and between storage locations. Manual moves using the RMM CHANGEVOLUME subcommand could be used to support dynamic shelf-management. Vital record processing sets the required location where a volume should be stored. Storage location management sets the destination for the volume and optionally assigns the exact shelf location to be used for the volume while it is in the storage location.

JCL for Storage Location Management Processing

To request storage location management processing, specify the DSTORE parameter. You can submit a job with JCL similar to Figure 111.

```
//HSKP     EXEC PGM=EDGHSKP,
//     PARM='DSTORE'
//MESSAGE  DD DISP=SHR,DSN=HSKP.MESSAGES
```

Figure 111. Example of JCL for Storage Location Management Processing

To move all volumes from one location to another location using the INSEQUENCE and REASSIGN parameters, you can submit a job with JCL that is similar to Figure 112.

```
//HSKP EXEC PGM=EDGHSKP,
//PARM='DSTORE(LOCATION(SHELF:VAULT),INSEQUENCE,REASSIGN)
//MESSAGE DD DISP=SHR,DSN=HSKP.MESSAGES
```

Figure 112. Example of JCL for Storage Location Management Processing Using the LOCATION, INSEQUENCE, and REASSIGN Parameters

How Storage Location Management Processing Works

Storage location management processing assigns volume destinations and sets required volume destinations that are based on the results of other inventory management functions. Storage location management must be run after vital record processing because the volume destinations are determined by vital record processing. See “DFSMSrmm Support for Stacked Volumes When Stacked Volume Support Is Enabled” on page 97 and “DFSMSrmm Support for Stacked Volumes When Stacked Volume Support Is Not Enabled” on page 101 for information about the processing sequence you must use when a VTS is in use. See “Confirming Global Volume Movement” on page 309 for information about the global move confirmation processing that takes place during vital record processing.

Storage location management processing assigns a shelf location, known as the bin number, to volumes moving to and among shelf-managed storage locations and sets a destination storage location. If the volume destination is a system-managed tape library, DFSMSrmm does not assign a bin number.

When you have enabled stacked volume support, storage location management is performed for stacked volumes, not for contained volumes. For contained volumes, the location of the stacked volume identifies the contained volume location. DFSMSrmm sets the required location for the stacked volume based on the required location defined for all volumes contained in the stacked volume. At the completion of export processing, the stacked volume has a required location. The required location is set during export based on the required location and priority of each exported volume. At the completion of export processing, the required location is used to set a destination if the location is not a shelf-managed storage location. If the destination location is shelf-managed, DFSMSrmm sets the destination during storage location management and assigns a bin number.

When you do not enable stacked volume support, DFSMSrmm does not consider the stacked volume when performing storage location management.

DFSMSrmm marks volumes that are not residing in a system-managed library as 'intransit'. DFSMSrmm marks volumes that are residing in system-managed libraries as 'intransit' only after you eject them.

Recommendation: After successful storage location processing, schedule a job or job step to eject all volumes to be moved using the bulk output station. To automatically eject physical volumes, add the job step shown in Figure 113 to the job that runs EDGHSKP vital record processing and storage location management processing.

```
// EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 RMM SEARCHVOLUME VOLUME(*) OWNER(*) LIMIT(*) -
   LOCATION(atlname) DESTINATION(storename) -
   INTRANSIT(N) CLIST('RMM CHANGEVOLUME ',' EJECT(BULK)')
 EXEC EXEC.RMM.CLIST
/*
```

Figure 113. Automatically Ejecting Volumes from System-managed Libraries

The job step shown in Figure 113 runs the batch TMP and uses DFSMSrmm TSO subcommands to produce a list of volumes to be ejected and then uses the list to initiate the ejects. To eject stacked volumes, you can use the same DFSMSrmm TSO subcommands that you used for physical volumes to generate a list to be ejected, but you must use the list to enter the requests at the Library Manager console.

You can override automatic vital record specification location selection by using the RMM CHANGEVOLUME subcommand. See “Moving Volumes Manually” on page 123 for more information.

To select a suitable bin number for a volume moving to a storage location, DFSMSrmm uses the volume media name and the LOCDEF media names. If the media name is specifically defined on LOCDEF, DFSMSrmm looks for a bin number which is empty and uses the same media name. If the specific media name is not found, DFSMSrmm checks if '*' is defined on the LOCDEF command. If it is, then DFSMSrmm looks for a bin with a media name of '*'. If neither the specific media name nor '*' is found on the LOCDEF command, then the volume cannot be sent to that location, and DFSMSrmm issues message EDG2412E.

To determine if a volume move is required, DFSMSrmm uses vital record processing information. If a move to a shelf-managed storage location is required, DFSMSrmm selects an empty bin in the target location and assigns it to the volume based on media name and LOCDEF information.

If there are no empty bin numbers available in a storage location, DFSMSrmm issues message EDG2403E. Inventory management continues and DFSMSrmm does not change the status of the volumes destined for a storage location which has no empty bin numbers. The volumes are moved in subsequent inventory management runs after more empty bin numbers are added, or when existing bin numbers are freed by other volumes being moved out of the storage location. After you have corrected the problem, request storage location management processing again.

If you move volumes by set, DFSMSrmm sets the same required location for all the volumes in the set that are retained by vital record specifications. DFSMSrmm sets the location based on the priority of each volume's required location. See "Defining System Options: OPTION" on page 134 for information about implementing volume movement by set by using the DFSMSrmm EDGRMM*xx* parmlib OPTION command MOVEBY(SET) operand.

When you run storage location management processing without specifying a location, DFSMSrmm processes all volumes. You can request that DFSMSrmm only process volumes in specific locations when you run the EDGHSKP exec using the LOCATION parameter to specify from and to locations. See "EXEC Parameters for EDGHSKP" on page 283 for a detailed description. If you use the INSEQUENCE parameter, DFSMSrmm assigns volumes to bins in sequential volume serial number order and bin number order. If you use the REASSIGN parameter, DFSMSrmm reassigns volumes to bins during processing. To maximize reuse of bins, use both the INSEQUENCE and REASSIGN parameters and specify the DFSMSrmm parmlib OPTION command REUSEBIN(STARTMOVE) operand.

Running Expiration Processing

During expiration processing, DFSMSrmm performs the following functions:

- Identifies volumes not required for vital records that are ready to expire by checking the expiration date.
- Determines the type of release action, if any, needed for the volumes that have reached their expiration date.
- Manages the release actions for each volume before returning it to the scratch pool or removing it from the library.
- Uncatalogs all data sets for volumes returning to scratch status if you have specified UNCATALOG(Y or S) in parmlib member EDGRMMxx.
- | Updates the catalog entry to remove the volume for the part of the DFSMSrmm control data set on the volume returning to scratch for a multivolume data set. |
- Optionally, updates RACF tape security profiles for volumes returning to scratch status.
- Ensures all volumes in a set are not expired if any one volume does not expire when the EDGRMMxx parmlib option RETAINBY(SET) is in use.

JCL for Expiration Processing

To request expiration processing, specify the EXPROC parameter. You can submit a job with JCL similar to Figure 114.

```
//HSKP     EXEC PGM=EDGHSKP,
//     PARM='EXPROC'
//MESSAGE  DD DISP=SHR,DSN=HSKP.MESSAGES
```

Figure 114. Example of JCL for Expiration Processing

How Expiration Processing Works

DFSMSrmm checks the expiration date for volumes not retained by a vital record specification. If the expiration date has been reached, DFSMSrmm changes the volume status to 'pending release'. DFSMSrmm checks for any actions that should be taken before the volume can be released. DFSMSrmm defers most release actions for later processing. However, DFSMSrmm processes the 'notify owner' release action immediately if an electronic address is provided for the owner and the DFSMSrmm system option NOTIFY is in use.

When a data set is no longer retained by a vital record specification, DFSMSrmm releases the volume on which the data set resides only if no data set on the volume is retained by a vital record specification. If you use the DFSMSrmm EDGRMM*xx* parmlib OPTION command VRSEL(NEW) option and the RMM ADDVRS RELEASE(EXPIRYDATEIGNORE) operand, DFSMSrmm ignores the volume expiration date and uses information in a vital record specification to control retention.

DFSMSrmm does not immediately return a volume to scratch status or to its owner when a volume reaches its expiration date and is not retained by a vital record specification. You must run expiration processing two times to return a volume to scratch status or to its owner. The first run of expiration processing sets the volume status to pending release. The second run of expiration processing completes the return. DFSMSrmm does not change the volume status during the first run of expiration processing. DFSMSrmm marks the volume as pending release during the first run of expiration processing in case you want to reclaim the volume. Running expiration processing two times give you time to make changes to the volume status before the volume is released.

If you do not need to run expiration processing in two runs, specify the DFSMSrmm EDGRMM*xx* parmlib OPTION command VRSEL(NEW) option and the RMM ADDVRS RELEASE(SCRATCHIMMEDIATE) operand. This enables you to return volumes to scratch in a single run of expiration processing. Sometimes DFSMSrmm cannot make the return in a single run, for example there may be other release actions required. Also, when all catalogs are not fully shared or when TCDBs for partitioned tape libraries are not shared, you might need to run expiration processing more than one time to return volumes to scratch status.

See "Confirming Global Volume Movement" on page 309 for information about the global move confirmation processing that takes place during vital record processing.

If you retain volumes by set, DFSMSrmm retains all the volumes in the set if any volume in the set is retained. DFSMSrmm sets the retention date for all the volumes in the set to the highest retention date for the volumes in the set. See "Defining System Options: OPTION" on page 134 for information about implementing volume retention by set by using the DFSMSrmm EDGRMM*xx* parmlib OPTION command RETAINBY(SET) operand.

Returning Volumes to Scratch Status

DFSMSrmm returns volumes to scratch that are already in 'pending release' status,
| that can be returned to scratch on the running system, and require no other action than returning to scratch. These volumes are available to satisfy scratch mount processing or RMM TSO GETVOLUME subcommand requests.

| To determine if a volume can return to scratch on the running system, DFSMSrmm
| considers if the scratch action has been manually confirmed, the TCDB, and
| catalog sharing. If the scratch action has been manually confirmed, the return to

scratch can be processed on any system; otherwise, the following must be considered: The volume must be in the TCDB and the system-managed library must be defined. When catalogs are not fully shared, the volume must be returned to scratch on a system with access to the catalogs where the first file was created.

When DFSMSrmm returns a volume to scratch status, DFSMSrmm clears the following information from the control data set:
- Volume description.
- Jobname.
- Accounting information.
- Access list.
- Volume owner.
- Owner access.
- Software product details. DFSMSrmm removes the volume from the product's list of volumes.

DFSMSrmm updates the volume information to indicate that the volume is no longer retained by a vital record specification. When a volume has been released manually using the RMM DELETEVOLUME subcommand, but is still retained by a vital record specification, DFSMSrmm sets the retention date to the current date and updates the volume information to indicate that the volume is no longer retained by a vital record specification.

During inventory management expiration processing, DFSMSrmm calls the EDGUX200 installation exit. The exit is called every time a volume is identified as ready to return to scratch. See "Using the DFSMSrmm EDGUX200 Installation Exit" on page 248 for information on the EDGUX200 installation exit processing.

In addition, DFSMSrmm requests that all data sets recorded in the control data set for the volume are uncataloged. DFSMSrmm performs RACF processing as described in Table 29 on page 186.

Tracking Volumes with Permanent Errors

DFSMSrmm keeps track of permanent I/O errors for a volume. During each run of expiration processing, DFSMSrmm sets the release actions for volumes that are not pending release. During the run volumes that are marked with the return to scratch release action, that have permanent I/O errors will be marked for replacement. DFSMSrmm only identifies those volumes with the 'return to scratch' release action, not those with the 'return to owner' action.

Use the RMM SEARCHVOLUME subcommand to obtain information about volumes that might need replacement. You can use the extract data set or the DFSMSrmm ISPF dialog to check for volumes with temporary errors. See "JCL for Creating an Extract Data Set" on page 282 for information about using the extract data set to identify volumes with temporary errors.

Managing Open Data Sets

DFSMSrmm produces a report in the MESSAGE file as shown in Figure 115 on page 305 that lists tape data sets that might be open as well as messages describing expiration processing.

```
EDG2404W VOLUME 111020 FOR JOB STSGGWC1 IS OPEN - VOLUME HAS EXPIRATION DATE 16/03/1993
EDG2404W VOLUME 111023 FOR JOB STSGGWC1 IS OPEN - VOLUME HAS EXPIRATION DATE 16/03/1993
EDG2229I NUMBER OF VRS RECORDS READ IS 14
EDG2420I TOTAL VOLUMES READ                        =            1970
EDG2421I TOTAL VOLUMES UPDATED                     =             330
EDG2422I TOTAL VOLUMES, THIS RUN, KEPT FOR VRS     =              50
EDG2423I TOTAL VOLUMES, THIS RUN, ASSIGNED TO STORES  =           85
EDG2424I TOTAL VOLUMES, THIS RUN, SET PENDING RELEASE =          126
EDG2425I TOTAL VOLUMES RETURNED TO SCRATCH         =              55
EDG2429I MAIN INVENTORY MANAGEMENT UPDATES HAVE COMPLETED SUCCESSFULLY
EDG2307I INVENTORY MANAGEMENT TASK VRSEL    COMPLETED SUCCESSFULLY
EDG2307I INVENTORY MANAGEMENT TASK DSTORE   COMPLETED SUCCESSFULLY
EDG2307I INVENTORY MANAGEMENT TASK EXPROC   COMPLETED SUCCESSFULLY
EDG2307I INVENTORY MANAGEMENT TASK RPTEXT   COMPLETED SUCCESSFULLY
EDG6401I MASTER FILE BACKUP SUCCESSFUL
```

Figure 115. Expiration Processing Report

These might be valid cases, or jobs that are currently active. However, some data sets on the list might be produced by a job that failed so the data set was never closed. Review this list and make the necessary corrections. The expiration date in the message, shows when the volume will be automatically released. During vital record processing, DFSMSrmm matches open data sets to vital record specifications using the OPEN reserved data set name mask or to vital record specifications defined with data set name masks.

Running DFSMSrmm Catalog Synchronization

DFSMSrmm always tracks tape data set catalog activity independently of the CATSYSID parameter. When you enable catalog synchronization, DFSMSrmm synchronizes the DFSMSrmm control data set and available user catalogs. Enabling catalog synchronization is normally a one-time task. Once the catalog status is synchronized, DFSMSrmm continually tracks and updates the catalog status. You enable catalog synchronization by specifying the DFSMSrmm EDGRMMxx OPTION CATSYSID operand described in Defining System Options: OPTION “Defining System Options: OPTION” on page 134. When you specify the CATSYSID operand, DFSMSrmm uses the catalog search interface to obtain catalog information used for DFSMSrmm vital record processing. DFSMSrmm performs checking to determine that the catalog search interface catalog environment does not have any errors that might prevent successful processing. DFSMSrmm issues messages EDG2235E, EDG2236I, or EDG2237E when an error is encountered. If you do not specify the DFSMSrmm EDGRMMxx OPTION CATSYSID operand, DFSMSrmm vital record processing uses catalog locates to determine if data sets are still cataloged.

You must re-synchronize the DFSMSrmm control data set and user catalogs under these conditions:

- | You ran EDGHSKP with CATSYNCH and VERIFY, and differences in catalog status were found. DFSMSrmm clears the catalog synchronization date and time to force you to run CATSYNCH.
- DFSMSrmm is not active when catalog activity for tape data sets is taking place. DFSMSrmm issues messages EDG8200E and EDG8201E when DFSMSrmm cannot track catalog updates.
- A DFSMSrmm failure occurs during catalog processing. DFSMSrmm issues EDG8200E and EDG8201E when an error is detected in processing.
- | You connect or disconnect user catalogs, that contain entries for tape data sets, to the master catalog.

| When you use IDCAMS REPRO MERGECAT for catalog maintenance, you do not
| need to run a job to re-synchronize the DFSMSrmm control data set and catalogs.

Recommendation: Use EDGUTIL with the UPDATE parameter and with the SYSIN command CONTROL CATSYNCH(NO) parameter when you think the DFSMSrmm control data set and the catalogs are no longer synchronized. This prevents DFSMSrmm from relying on the catalog information in the control data set. Later you can re-synchronize the information or DFSMSrmm will synchronize the information automatically at inventory management time when the catalogs are fully shared.

You can automate the response to the DFSMSrmm messages EDG8200E and EDG8201E to force re-synchronization before any inventory management functions dependent on catalog information are run. Run EDGUTIL UPDATE with the SYSIN command CONTROL CATSYNCH(NO) parameter to set up this automation.

There are some limitations to using DFSMSrmm catalog synchronization. Some of the limitations include the following conditions:

- You must define data sets to DFSMSrmm before you create catalog entries for them.
- If either of the following applies to you, do not rely on DFSMSrmm for maintaining catalog synchronization. However, you can exploit the status tracking DFSMSrmm performs, by always specifying the EDGHSKP CATSYNCH option when running expiration processing or vital record processing functions.
  - If you use the DFSMSrmm TSO subcommands to add data sets to the DFSMSrmm control data set, you should do so before cataloging the data sets. If you have to add cataloged data sets you can run EDGHSKP with CATSYNCH to synchronize the catalogs with DFSMSrmm.
  - If you create cataloged tape data sets by writing to tape volumes but do not allocate the file using the catalog entry, DFSMSrmm cannot track the catalog status. DFSMSrmm relies on intercepting the catalog and uncatalog activity and the data set information being already recorded by DFSMSrmm at the time of the catalog activity.

Control how DFSMSrmm performs catalog synchronization by using the EDGRMM*xx* parmlib OPTION CATSYSID operand. If you specify the CATSYSID(*) operand, you must ensure that the catalog environment is shared. If you specify CATSYSID(*) without ensuring that catalogs are shared, you can lose data. If you specify the CATSYSID operand with system IDS, you cannot run DFSMSrmm inventory management until the DFSMSrmm control data set and user catalogs are synchronized.

If catalogs are not fully shared, and you specify the EDGRMMxx parmlib OPTION UNCATALOG(Y/S) operand, you must run expiration processing on each system. This ensures that DFSMSrmm uncatalogs data sets in the correct catalogs on return to scratch. You must run expiration processing on each system to avoid running low on scratch volumes because DFSMSrmm only returns volumes to scratch when processing in the correct catalog environment.

To ensure that DFSMSrmm processes the return to scratch, you must also specify the EDGRMMxx parmlib OPTION CATSYSID operand with the system IDs on which the volumes were created. DFSMSrmm uses the CATSYSID operand to determine the systems to which volumes can be returned to scratch status. DFSMSrmm checks the system creation ID for the first file on the volume with the list of IDs that are specified for the CATSYSID operand. If there is a match, DFSMSrmm processes the return to scratch.

DFSMSrmm Catalog Processing

When you run DFSMSrmm with user catalogs and the DFSMSrmm control data set unsynchronized, DFSMSrmm issues catalog locates as required to check if data sets are cataloged. Catalog locates use the standard catalog search to find if a data set is cataloged. Any data sets cataloged using JOBCAT or STEPCAT might not be found to be cataloged because standard catalog search might not find them.

When you run DFSMSrmm with user catalogs and the DFSMSrmm control data set synchronized, DFSMSrmm does not need to issue catalog locates to find if a data set is cataloged. The catalog status tracked by DFSMSrmm in the control data set is used to determine if a data set is cataloged. When DFSMSrmm tracks catalog status, it does so regardless of whether JOBCAT or STEPCAT is used. During catalog synchronization DFSMSrmm uses the Catalog Search Interface (CSI) to retrieve data set catalog information. CSI returns catalog information for all data sets in all catalogs that are in or connected to the master catalog. Because of this, DFSMSrmm can detect a data set is cataloged even if it cannot be found using the standard catalog search.

JCL for Catalog Synchronization

To request catalog synchronization, specify the CATSYNCH parameter. You can submit a job with JCL similar to Figure 116.

```
//HSKP     EXEC PGM=EDGHSKP,
//     PARM='CATSYNCH'
//MESSAGE  DD DSN=MESSAGE.FILE.NAME,DISP=SHR
//ACTIVITY DD DSN=ACTIVITY.FILE.NAME,DISP=SHR
```

Figure 116. Example of JCL for Catalog Synchronization Processing

Specify the CATSYNCH and the VERIFY parameter to perform a trial run of catalog synchronization. You can submit a job with JCL similar to Figure 117.

```
//HSKP     EXEC PGM=EDGHSKP,
//     PARM='CATSYNCH,VERIFY'
//MESSAGE  DD DSN=MESSAGE.FILE.NAME,DISP=SHR
//ACTIVITY DD DSN=ACTIVITY.FILE.NAME,DISP=SHR
```

Figure 117. Example of JCL for Trial Run Catalog Synchronization Processing

| IBM recommends that you run CATSYNCH with VERIFY to perform a trial run early
| in your implementation so that you can be sure your catalogs are free from errors
| before you get started.

Synchronizing the DFSMSrmm Control Data Set with User Catalogs in a Fully Shared Catalog Environment

You can use shared user catalogs whether or not the control data set and user catalogs are synchronized. Synchronize the DFSMSrmm control data set and user catalogs so that DFSMSrmm does not need to locate catalog information for every data set that is managed using catalog control. To synchronize the DFSMSrmm control data set with user catalogs:

1. Install the DFSMSrmm level of code that performs catalog tracking and catalog synchronization on all systems that are sharing the control data set. When you

have the correct level of DFSMSrmm code installed, DFSMSrmm dynamically records all catalog updates for tape data sets in the DFSMSrmm control data set.

2. Specify the DFSMSrmm EDGRMM*xx* parmlib OPTION CATSYSID(*). When the CATSYSID operand is specified in the EDGRMM*xx* parmlib member, you cannot run VRSEL or EXPROC processing unless the control data set is synchronized or CATSYNCH is specified in the same processing run. When running EDGHSKP expiration processing or vital record processing, if CATSYSID(*) is specified in the EDGRMM*xx* parmlib member and the control data set is not synchronized with the catalogs, DFSMSrmm initiates catalog synchronization even if you have not specified it.
3. Set up automatic synchronization by automating the responses to DFSMSrmm messages EDG8200E and EDG8201E. Run the EDGUTIL UPDATE with the SYSIN command CONTROL CATSYNCH(NO).
4. Run the DFSMSrmm EDGHSKP utility with the PARM=CATSYNCH parameter to synchronize all data set records in the control data set with the status from the catalog. Before you run CATSYNCH you should verify that your MESSAGE data set is large enough to contain all the messages issued during the catalog synchronization process. For example, when you run CATSYNCH for the first time, perhaps as part of DFSMSrmm implementation, or you run CATSYNCH with VERIFY you should expect messages for each data set that must be updated or the status is different between the catalogs and the DFSMSrmm control data set.

Synchronizing the DFSMSrmm Control Data Set with User Catalogs When Catalogs Are Not Fully Shared

When catalogs are not fully shared, you can have non-GDG and GDG data sets with the same name that are cataloged but in different, unshared catalogs. To retain all copies of all data sets, define a retention policy that uses DAYS with COUNT(99999) rather than CYCLES. Using CYCLES can produce unpredictable results for example, the premature expiration of data sets.

To ensure cycles are grouped together, use the jobname to further qualify the data sets, using a different jobname for each set of cataloged data sets. For example, use JOBNAME(xjzA) for data sets created on system A, and JOBNAME(xjzB) for data sets created on system B so that the cycle retention is based on the scope of the creation system.

The DFSMSrmm control data set and user catalogs must be synchronized for DFSMSrmm to support unshared user catalogs. To synchronize the DFSMSrmm control data set with user catalogs, follow these procedures:

1. Ensure that all systems that share the control data set are running the DFSMSrmm level of code that supports catalog tracking and catalog synchronization. When you have the correct level of DFSMSrmm code installed, DFSMSrmm dynamically records all catalog updates for tape data sets in the control data set.
2. Specify the DFSMSrmm EDGRMM*xx* parmlib OPTION CATSYSID operand with the system IDs that are common to the set of user catalogs on the system. Specify current IDs and previously used IDs if you are still retaining data sets from those systems.
3. Set up automatic synchronization by automating the responses to DFSMSrmm messages EDG8200E and EDG8201E running the EDGUTIL UPDATE with the SYSIN command CONTROL CATSYNCH(NO).

4. Run the DFSMSrmm EDGHSKP utility with the PARM=CATSYNCH parameter to partially synchronize all data set records in the control data set with the status from the accessible catalogs. Run CATSYNCH on other systems that share the control data set until all the catalogs have been synchronized. Before you run CATSYNCH you should verify that your MESSAGE data set is large enough to contain all the messages issued during the catalog synchronization process. For example, when you run CATSYNCH for the first time, perhaps as part of DFSMSrmm implementation, or you run CATSYNCH with VERIFY you should expect messages for each data set that must be updated or the status is different between the catalogs and the DFSMSrmm control data set.
5. Run EDGUTIL with PARM=UPDATE parameter and CONTROL CATSYNCH(YES) to set the 'last catalog synchronization date and time' in the DFSMSrmm control data set control record.
6. Run inventory management functions on any one system at any time. However, if you run expiration processing to return volumes to scratch status, DFSMSrmm only processes a subset of volumes.
7. You must run EDGHSKP expiration processing on multiple systems if you specify the EDGRMMxx parmlib OPTION UNCATALOG(Y or S). You should run expiration processing once for each set of user catalogs on a system with access to those user catalogs.

Confirming Global Volume Movement

Vital record processing determines the volume movements required and storage location management processing assigns the volume destinations. You must ensure that volume movements are completed and confirmed to DFSMSrmm.

You use the RMM subcommands and DFSMSrmm ISPF dialog to confirm that volume movements have taken place. You use the RMM CHANGEVOLUME * CONFIRMMOVE subcommand to change the status of a volume move from PENDING to CONFIRMED. You can perform confirmation for a single volume or for multiple volumes. Performing confirmation for multiple volumes is called global confirmation. In DFSMSrmm, confirmation means to issue a command to notify DFSMSrmm that a volume has moved to a location or that a release action defined for a volume has been completed. If you perform confirmation for a single volume, DFSMSrmm processes the confirmation immediately. If you issue a request for a global confirmation, DFSMSrmm processes the global confirmation during inventory management processing.

Global move confirmation is performed as part of DFSMSrmm inventory management. DFSMSrmm performs move confirmation for a volume, prior to making any other updates to a volume, as part of the following inventory management functions:
- Vital record processing
- Storage location management processing
- Expiration processing

If a volume move is outstanding, and it has been globally confirmed, DFSMSrmm performs confirmation of the move for that volume. DFSMSrmm confirms moves in progress prior to starting any new moves that might be set by the current inventory management run. If each of the inventory management functions is in a separate job step, then global move confirmation is performed once for each step. If the inventory management functions are run in a single job step, DFSMSrmm performs global move confirmation once.

The timing of the inventory management runs is very important. Here are some things to consider:

- If you requested global move confirmation and want to change it, you can undo the request prior to the next run of inventory management. You can use the RMM CHANGEVOLUME subcommand to undo global confirmation of volume moves. After the next run of inventory management, DFSMSrmm will have confirmed the volume moves making undoing the confirmation of the moves difficult.
- When you request an extract data set to be used for generating movement or inventory reports, consider the timing of inventory management functions. An extract data set produced after a run of vital record processing and storage location management processing, contains the information of new volume movement for movement reports. An extract data set produced after a global confirm move and a run of any of the listed inventory management functions, contains information about volumes after moves have been confirmed.
- To make scratch volumes available after a move and when you use global move confirmation, make sure you run expiration processing to complete the global move confirmation and return the moved volumes to scratch status.

Confirming Global Release Actions

If the RMM CHANGEVOLUME subcommand with * CONFIRMRELEASE operands is issued to confirm global release actions, for any volume with the action outstanding, DFSMSrmm marks the corresponding action complete. Global release action confirmation takes place during expiration processing just prior to when DFSMSrmm performs a check for outstanding global confirmation or if the volume expiration date has been reached.

When you use the NOTIFY release action, the NOTIFY release action must be confirmed in order for any other release action to be processed. DFSMSrmm can confirm this automatically by sending a notify message, or you can confirm it manually. See “OPTION Command Operands” on page 137 for more information about the NOTIFY parmlib OPTION.

If the ‘return to owner’ action is confirmed, the volume is deleted from the DFSMSrmm control data set. If the erase or initialize release actions are confirmed, DFSMSrmm deletes the data set information for those volumes with the actions outstanding. If erase is confirmed, DFSMSrmm sets the initialize action as outstanding so that the correct volume labels can be written to the erased volume.

If the RMM DELETEVOLUME subcommand with the RELEASE operand was issued, DFSMSrmm changes the volume status to ‘pending release’ during subcommand processing. Use expiration processing to complete the release process. These volumes are returned to scratch during the first EDGHSKP run, if no other release actions are required.

Backing Up the Control Data Set

You can use EDGHSKP or EDGBKUP described in Chapter 15, “Maintaining the Control Data Set,” on page 317 to back up the control data set and the journal. Choose the appropriate utility based on your needs. You can start the back up of the control data set or journal at any time. But you cannot start backup if backup is already in progress. You need to schedule backing up and clearing of the journal before it exceeds its threshold using the EDGHSKP utility.

- Use EDGHSKP to back up both the control data set and the journal when the DFSMSrmm subsystem is active by specifying the BACKUP DD statement and the JRNLBKUP DD statement.

  Use the EDGHSKP utility to back up the control data set and to clear the journal data set.
- Use EDGBKUP to back up the control data set and journal when the DFSMSrmm subsystem is not active.

  Use the EDGBKUP utility to back up the control data set and journal when DFSMSrmm is active. EDGBKUP does not clear the journal data set.

Schedule regular backups for both the control data set and journal. Backing up the DFSMSrmm control data set and the journal as the first step in inventory management is a way for you to create a restore point for the control data set. Schedule back ups of the journal when it reaches the threshold that you define. Use the DFSMSrmm parmlib OPTION command JOURNALFULL operand, described in "Defining System Options: OPTION" on page 134. Clearing out the journal data set regularly prevents the journal from becoming full, and thus reduces the risk of losing the updates to the control data set. You can automate the process of backing up the control data set and clearing the journal as described in "Steps for Automating Control Data Set Backup and Journal Clearing" on page 314.

You should plan to keep multiple backup copies, so that, should a backup fail, you still have a previous, successful backup from which to recover. Use the current journal with the most recent control data set and journal backups to forward recover to the point of failure.

You can request that DFSMSrmm back up the control data set and the journal, using the access method services (AMS) REPRO command or DFSMSdss DUMP command. When you use the access method services REPRO command or DFSMSdss without concurrent copy for backup, DFSMSrmm does not allow updates to the control data set during control data set backup. Backup using DFSMSdss enables output directly to tape. DFSMSrmm resets the journal data set and discards journal records if the back up completes successfully. It is not necessary to use DFSMSdss concurrent copy when you specify BACKUP(DSS).

When DFSMSrmm is performing an intrusive backup, some processing might wait until the backup completes or some command processing can be interrupted. DFSMSrmm fails RMM TSO subcommand requests that update the DFSMSrmm control data set. For example, if you start backup using the AMS REPRO backup method, command processing can be interrupted. DFSMSrmm functions such as the DFSMSrmm TSO SEARCH subcommands, that only read the control data set, continue uninterrupted.

Example: Issue the RMM ADDRACK subcommand to add 2000 new shelf locations.

```
RMM ADDRACK RK0000 COUNT(2000)
```

Example: If backup processing starts before the request completes, DFSMSrmm issues these messages.

```
EDG3212E REQUEST REJECTED - DFSMSrmm BACKUP
         CURRENTLY IN PROGRESS
EDG3017I THE ERROR OCCURRED WHILE ADDING RACK NUMBER RK0700
EDG3018I 700 RACK NUMBER(S) ADDED
```

Example: When backup completes, issue the ADDRACK subcommand to add the remainder of the shelves.

```
RMM ADDRACK RK0700 COUNT(1300)
```

When a tape is used during inventory management, DFSMSrmm updates the control data set. If a job that opens or closes a tape data set tries to update the control data set during AMS REPRO backup processing, DFSMSrmm waits for up to five minutes. If backup is still in progress, DFSMSrmm issues a write-to-operator message EDG4010D that prompts the operator to retry or cancel each job. If the operator specifies retry, DFSMSrmm retries the job five more times at minute intervals before again issuing a write-to-operator if backup processing has not been completed.

When you use the DFSMSdss DUMP command to request a back up, plan to keep backup copies for both the control data set and the journal. For any backup being restored, a journal backup is also required for forward recovery. For the latest control data set backup, both the latest journal backup and the current journal must be used for forward recovery.

To use DFSMSdss concurrent copy, you must authorize the DFSMSrmm batch inventory management backup job to the RACF profile STGADMIN.ADR.DUMP.CNCURRNT. See z/OS DFSMSdss Storage Administration Guide for information about protecting DFSMSdss keywords with RACF.

When you use DFSMSdss to back up the control data set and you use concurrent copy, DFSMSrmm resets the DFSMSrmm journal only if you also back up the journal data set. The journal and the journal backup are required to forward recover to the latest point in time.

When you use DFSMSdss concurrent copy for backup, DFSMSrmm continues to process commands and update the control data set. As an alternative to use of concurrent copy capable hardware, you can use snapshot capable hardware. When you request backup using DFSMSdss, DFSMSdss uses CC-compatible snapshot instead of concurrent copy.

JCL for Backing Up the Control Data Set and Journal

DFSMSrmm obtains the name of the control data set and journal from the running DFSMSrmm subsystem.

Rule: Do not specify the data set names in the JCL. If you do, the job fails.

To create a backup copy using EDGHSKP, specify the BACKUP parameter.

1. Submit a job to back up the control data set to tape with JCL as shown in Figure 118 on page 313.

```
//EDGHSKP EXEC PGM=EDGHSKP,PARM='BACKUP(DSS)'
//MESSAGE DD DISP=SHR,DSN=RMM.MESSAGE
//SYSPRINT DD SYSOUT=*
//BACKUP DD DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.CDS(+1),
// LABEL=(,SL)
//JRNLBKUP DD DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.JRNL(+1),
// DCB=(RECFM=VB,BLKSIZE=0,LRECL=9000),
// LABEL=(2,SL),VOL=REF=*.BACKUP
//DSSOPT DD *
CONCURRENT OPTIMIZE(1) VALIDATE
/*
```

Figure 118. Example of JCL for Backing Up the Control Data Set and the Journal to Tape

2. Specify the JRNLBKUP DD statement only if you want to back up the journal. The DSSOPT DD statement is optional and allows you to customize the DFSMSdss DUMP and DFSMSdss RESTORE commands. See “Customizing the DSSOPT DD Statement” on page 323 for information about changing the DSSOPT DD statement.
3. Specify BACKUP(DSS) when you want to use DFSMSdss to perform the back up. You do not need to use concurrent copy to specify BACKUP(DSS). Using DFSMSdss concurrent copy permits the update of the control data set during backup processing so that all tape activities can continue during backup processing. To use DFSMSdss concurrent copy, you must have a concurrent copy environment set up. If you specify BACKUP(DSS) without concurrent copy, you can still direct the output to tape. All other updates to the control data set will wait until the control data set backup completes. If you specify BACKUP(DSS) and you use concurrent copy, DFSMSrmm clears the journal only if you also back up the journal.

You can submit a job to back up the control data set to DASD with JCL as shown in Figure 119.

```
//BACKUP EXEC PGM=EDGHSKP,PARM='BACKUP(AMS)'
//SYSPRINT DD SYSOUT=*
//MESSAGE DD DSN=RMM.MESSAGE(0),
// DISP=SHR
//BACKUP DD DSN=RMM.BACKUP(+1),
// DISP=(,CATLG,DELETE),
// UNIT=SYSDA,
// AVGREC=U,SPACE=(4096,(1000,500),RLSE),
// LRECL=9000,BLKSIZE=0,RECFM=VB,
// DCB=RMM.GDGMODEL
//JRNLBKUP DD DSN=RMM.JRNLBKUP(+1),
// DISP=(,CATLG,DELETE),
// UNIT=SYSDA,
// AVGREC=U,SPACE=(4096,(1000,500),RLSE),
// LRECL=9000,BLKSIZE=0,RECFM=VB,
// DCB=RMM.GDGMODEL
```

Figure 119. Example of JCL for Backing Up the Control Data Set and Journal to DASD

Backing Up the Journal

You can backup and clear the journal without backing up the control data set. DFSMSrmm allows updates to the control data set during journal backup so that the impact of journal backup on other tasks is low. Use EDGHSKP when you want to backup and clear the journal. Use EDGBKUP when you only want to back up the journal and not clear it. Use the RMM LISTCONTROL TSO subcommand to find the time of the last journal back up.

When the journal reaches its threshold, schedule a backup of just the journal to minimize the time needed to clear the journal. When you restore the control data set, ensure that the correct journal backups are used for forward recovery.

JCL for Backing Up the Journal

DFSMSrmm obtains the name of the control data set and journal from the running DFSMSrmm subsystem.

Rule: Do not specify the data set names in the JCL. If you do, the job fails.

You can back up and clear the journal without taking a backup of the control data set. To back up the journal using EDGHSKP, specify the BACKUP parameter as shown in Figure 120.

```
//EDGBKP   EXEC PGM=EDGHSKP,PARM='BACKUP'
//SYSPRINT DD SYSOUT=*
//MESSAGE  DD DISP=SHR,DSN=RMM.MESSAGE
//JRNLBKUP DD DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.JRNL(+1),
//       LABEL=(1,SL)
```

Figure 120. Example of JCL for Backing Up and Clearing the Journal

Steps for Automating Control Data Set Backup and Journal Clearing

You need to ensure that the journal does not fill up because a full journal can impact tape usage on your system. You can run regular backups as part of inventory management in order to clear the journal. Automating control data set backup and clearing the journal help to ensure that the journal does not fill up.

To automate the clearing of the journal, follow these steps:

1. Specify a value for JOURNALFULL operand on the OPTION command in parmlib as described in “Defining System Options: OPTION” on page 134. If you do not specify a value, DFSMSrmm uses a value of 75%. Use only a single system to trigger backup using the journal threshold. If you have multiple systems and the journal threshold is reached, each system can start the backup procedure at the same time if you use the same threshold number. To avoid this situation if you have multiple systems, you can specify a different threshold on each system. For example, specify a threshold of 75% on the main system where DFSMSrmm is active and then specify thresholds of 80% and 85% for your other systems. You can also disable threshold processing on a system by specifying a zero value. Ensure that the systems you select have DFSMSrmm active and have a high chance of processing DFSMSrmm requests. A system that processes no or few DFSMSrmm requests is a bad choice for automatic backup because DFSMSrmm only checks the journal threshold when a request is processed.
2. Create a backup procedure in the system procedure library. The backup procedure that you write should use EDGHSKP to backup the control data set and journal because EDGHSKP also clears the journal. To automate the clearing of the journal without backing up the control data set, write your backup procedure to backup just the journal. See “Automating Backup” on page 382 for an example of a procedure that runs backup as part of inventory management. Alternatively, you might want to use the procedure to submit a batch job to perform the backup or to inform your job scheduler to submit the job. DFSMSrmm will not start the procedure if backup is already in progress. See “Event Triggered Tracking” on page 401 for information about using the IBM Tivoli Workload Scheduler for z/OS.

3. Specify the procedure name for the BACKUPPROC operand of the OPTION command in parmlib as described in “Defining System Options: OPTION” on page 134. DFSMSrmm starts this procedure when the journal threshold is reached. If you do not specify a procedure name, you cannot automate backup using DFSMSrmm. However, you could still use the EDG2107E message as a trigger for your site automation processes.

Return Codes for EDGHSKP

EDGHSKP can issue the return codes shown in Table 49.

Table 49. EDGHSKP Return Codes

Return Code	Explanation
0	All requested functions completed successfully.
4	DFSMSrmm encountered a minor error during processing. It issues a warning message and processing continues.
8	DFSMSrmm has stopped at least one requested function. Processing continues with the next requested function.
12	A severe error occurred during processing of one of the requested functions. DFSMSrmm stops the utility.
16	A severe error occurred during a required communication with the DFSMSrmm subsystem. DFSMSrmm stops the utility.

Chapter 15. Maintaining the Control Data Set

DFSMSrmm Samples Provided in SAMPLIB

- EDGJBKUP Sample JCL for Using the Backup Program
- EDGJUTIL Sample JCL for Initializing the Control Data Set
- EDGJMFAL Sample JCL for Allocating the Control Data Set
- EDGJNLAL Sample JCL for Allocating the Journal

Before you begin: DFSMSrmm provides you with the EDGHSKP utility, the EDGBKUP utility, and the EDGUTIL utility that you can use to maintain the control data set. EDGHSKP and EDGBKUP provide functions that you can use to back up the DFSMSrmm control data set. Review the following descriptions to determine which utility you should use for back up.

Use the DFSMSrmm utility EDGHSKP described in “Backing Up the Control Data Set” on page 310 to perform the following tasks:

- Back up the DFSMSrmm control data set and journal when DFSMSrmm is active.
- Clear the journal data set. Clear the journal only after back up is completed successfully to avoid losing changes that are made since the last backup. Without these latest changes, a forward recovery can only recover the control data set up to the last backup before the journal was cleared.

Use the DFSMSrmm utility EDGBKUP described in “Backing Up the Control Data Set” on page 324 to perform the following tasks:

- Back up the DFSMSrmm control data set and journal when DFSMSrmm is active.
- Back up or restore the DFSMSrmm control data set when DFSMSrmm is stopped or quiesced. You can use EDGBKUP independently of DFSMSrmm to back up, restore, and reorganize the DFSMSrmm control data set and to back up the journal.

Perform these tasks to maintain the DFSMSrmm control data set:

- Create the control data set control record as described in “Creating or Updating the Control Data Set Control Record” on page 341.
- Back up the control data set and the journal when DFSMSrmm is active using either EDGHSKP as described in “Backing Up the Control Data Set” on page 310 or EDGBKUP as described in “Backing Up the Control Data Set” on page 324.
- Back up the control data set and the journal when DFSMSrmm is inactive using EDGBKUP as described in “Backing Up the Control Data Set” on page 324.
- Back up the journal when DFSMSrmm is active or inactive using either EDGHSKP as described in “Backing Up the Journal” on page 313 or EDGBKUP as described in “Backing Up the Control Data Set” on page 324.
- Clear the journal by using EDGHSKP when DFSMSrmm is active as described in “JCL for Backing Up the Journal” on page 314
- Restore the control data set, and optionally forward recover when DFSMSrmm is stopped or quiesced using EDGBKUP as described in “Restoring the Control Data Set with Forward Recovery” on page 325.

- Restore the control data set by using non-DFSMSrmm products such as IDCAMS and DFSMSdss as described in “Using Non-DFSMSrmm Utilities to Restore the Control Data Set” on page 328.
- Forward recover the control data set by using EDGBKUP as described in “Forward Recovering the Control Data Set” on page 327.
- Reorganize the control data set using EDGBKUP as described in “Reorganizing the Control Data Set” on page 329.
- Move the control data set by using EDGHSKP as described in “Moving the Control Data Set and Journal to a Different Device” on page 332.
- Move the control data set by using non-DFSMSrmm utilities such as IDCAMS and products such as DFSMSdss as described in “Steps for Moving the Control Data Set using Non-DFSMSrmm Utilities” on page 336.
- Move the journal as described in “Moving the Journal using DFSMSrmm Utilities” on page 335.
- Recover from control data set update failures by using EDGBKUP as described in “Recovering from Control Data Set Update Failures” on page 331.
- Verify the contents of the control data set by using EDGUTIL as described in “Verifying the Contents of the Control Data Set” on page 343.
- Repair the control data set by using EDGUTIL as described in “Mending the Control Data Set” on page 345.
- Share the control data set as described in “Sharing the DFSMSrmm Control Data Set” on page 348.

This section contains information for using the DFSMSrmm EDGBKUP utility and the EDGUTIL utility:
- “Using EDGBKUP” on page 319
- “Using EDGUTIL for Tasks Such as Creating and Verifying the Control Data Set” on page 336

| DFSMSrmm Considerations when Client/Server Support is Enabled

| This topic describes utilities that provide restricted functions when run on a client or
| server system.

DFSMSrmm Utility	Considerations	Where to Find More Information
EDGUTIL	• When you run EDGUTIL on a client system, the utility can only process a DFSMSrmm control data set that is not in use by DFSMSrmm. Because there is no control data set for a client system, you cannot run with the active control data set. However, you could recover a backup copy of the DFSMSrmm control data set to the client system to run VERIFY(VOLCAT), VERIFY(SMSTAPE), and MEND(SMSTAPE). • When you run EDGUTIL on a server system, you cannot access the TCDB or library for tape libraries that you can only access from the client system. • When you run EDGUTIL on a client system and do not specify the MASTER DD, DFSMSrmm issues message EDG6101E and the utility ends with return code 12.	"Using EDGUTIL for Tasks Such as Creating and Verifying the Control Data Set" on page 336
EDGBKUP	• You cannot process an active DFSMSrmm control data set on a client system. All functions are available on a client system as long as you provide the DD statements for the control data set and journal. This enables you to use the BACKUP and RESTORE options independent of the DFSMSrmm subsystem. When you run EDGBKUP on a client system and do not specify the MASTER DD or JOURNAL DD, DFSMSrmm issues message EDG6101E and the utility ends with return code 12.	"Backing Up the Control Data Set" on page 310

Using EDGBKUP

This section describes the JCL, EXEC parameters, DD statements, and return codes associated with using the EDGBKUP utility. In addition, this section describes how to customize the DSSOPT DD statement.

JCL for EDGBKUP

This section describes the EXEC parameters for EDGBKUP and provides JCL examples to back up the control data set and journal, restore the control data set, and reorganize the control data set. Figure 121 on page 320 shows the JCL for EDGBKUP.

```
//EDGBKUP   EXEC PGM=EDGBKUP,PARM='BACKUP(DSS)'
//SYSPRINT DD SYSOUT=*
//MASTER    DD DISP=SHR,DSN=RMM.CDS
//JOURNAL  DD DISP=SHR,DSN=RMM.JOURNAL
//BACKUP    DD DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.CDS(+1),
//          LABEL=(,SL)
//JRNLBKUP DD DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.JRNL(+1),
//          LABEL=(2,SL),VOL=REF=*.BACKUP,LRECL=32756,
//          BLKSIZE=32760,RECFM=VB
```

Figure 121. JCL Example for EDGBKUP

EXEC Parameters for EDGBKUP

Figure 122 shows the EXEC parameters for EDGBKUP.

```
►►──┬─RESTORE──────────────────┬──────────────────────────────────────────►◄
    │          ┌─NREORG─┐      │
    └─BACKUP(──┼─DSS────┼──)──┘
               └─REORG──┘
```

Figure 122. EDGBKUP EXEC Parameters

BACKUP(DSS|NREORG|REORG)

Specify BACKUP to control the way DFSMSrmm backs up, and optionally reorganizes the DFSMSrmm control data set.

Restriction: You cannot RESTORE or REORG an active DFSMSrmm control data set. You must stop or quiesce DFSMSrmm or ensure that the control data set is not in use by DFSMSrmm.

DSS

Specify BACKUP(DSS) to use DFSMSdss to back up the control data set. DFSMSrmm uses DFSMSdss to back up the control data set and IDCAMS to back up the journal. To allow updates to the control data set during backup processing, set up the DFSMSdss concurrent copy environment or virtual concurrent copy environment. You must have the hardware and software required to establish a concurrent copy session or a virtual concurrent copy session.

Restriction: If you specify BACKUP(DSS) without establishing a concurrent copy session, DFSMSrmm performs back up processing but DFSMSrmm does not allow updates to the control data set until the control data set backup completes. If DFSMSrmm is active and you specify BACKUP(DSS), DFSMSrmm allows backup directly to tape when the journal is not full or locked.

DSSOPT is an optional DD statement that is for use with BACKUP(DSS). BACKUP allows you to customize the DFSMSdss options that DFSMSrmm uses. See “Customizing the DSSOPT DD Statement” on page 323 for additional information.

You can use the EDGBKUP utility or DFSMSdss to restore backups of the control data set that are created using DFSMSdss. If you use DFSMSdss to restore the backup, you must use the EDGBKUP utility to perform forward recovery to reset the control information in the restored control data set. DFSMSrmm issues message EDG0123D when you start DFSMSrmm and have not reset the control information in the control data set.

NREORG

Specify BACKUP(NREORG) to use IDCAMS to back up the control data set and to back up the journal data set.

NREORG is the default.

REORG

Specify BACKUP(REORG) to reorganize the control data set during backup processing. DFSMSrmm uses IDCAMS to back up the control data set, to optionally back up the journal, and to restore the control data set from the backup.

RESTORE

Specify RESTORE to restore the control data set from a backup copy. When you specify the BACKUP DD statement, DFSMSrmm restores the control data set. When you specify the JOURNAL DD statement, forward recovery is begun. You can specify both the BACKUP DD statement and the JOURNAL DD statement in the same job step to restore and forward recover the control data set from journal backups and the journal data set.

RESTORE allows you to customize the DFSMSdss options that DFSMSrmm uses. See “Customizing the DSSOPT DD Statement” on page 323 for additional information.

DD Statements for EDGBKUP

The data sets that are used by the EDGBKUP utility are described in Table 50.

Table 50. DFSMSrmm EDGBKUP Data Sets

DD Statement	Description
BACKUP	Contains the backup copy of the DFSMSrmm control data set. This data set is optional. When you run backup, specify the BACKUP DD statement, the JRNLBKUP DD statement, or both statements. You can back up directly to tape when you specify the BACKUP(DSS) parameter on the EXEC statement even when you have not set up DFSMSdss concurrent copy when DFSMSrmm is active.
DSSOPT	Contains DUMP or RESTORE command options used by DFSMSdss during processing. This data set is optional. See “Customizing the DSSOPT DD Statement” on page 323 for information about changing the commands.
JOURNAL	Identifies the DFSMSrmm journal to be used during backup processing. Identifies the journal backups and journal to be used during restore processing.
JRNLBKUP	Contains the backup copy of the DFSMSrmm journal. This data set is optional. When you run backup, specify the BACKUP DD statement, the JRNLBKUP DD statement, or both statements. DFSMSrmm uses IDCAMS to back up the journal when you specify the BACKUP(AMS) or BACKUP(DSS) parameter. You can back up the journal directly to tape when you specify the BACKUP(DSS) parameter when DFSMSrmm is active.
MASTER	Identifies the DFSMSrmm control data set. This data set is optional when you code the BACKUP on the EXEC statement if DFSMSrmm is active. This data set is required for RESTORE except when DFSMSdss is used for RESTORE. Only specify the MASTER DD if the target data set already exists.
SYSPRINT	Contains the utility program messages that IDCAMS and ADRDSSU issue when backing up the DFSMSrmm control data set. This data set can be a SYSOUT file.

Return Codes for EDGBKUP

EDGBKUP can issue the return codes shown in Table 51.

Table 51. EDGBKUP Return Codes

Return Code	Explanation
0	All requested functions completed successfully.
4	DFSMSrmm encountered a minor error during processing. It issues an informational message and continues processing.
8	DFSMSrmm encountered errors during the forward recovery or restore process. DFSMSrmm has updated the control record in the recovered control data set to show that recovery was not successful.
12	DFSMSrmm encountered a severe error during processing of one of the requested functions. DFSMSrmm stops the utility. DFSMSrmm has updated the control record in the recovered control data set to show that recovery was not successful.
16	DFSMSrmm encountered a severe error during a required communication with the DFSMSrmm subsystem. DFSMSrmm stops the utility.

Additional EDGBKUP Return Code Information

If DFSMSrmm encounters an error during restore processing, DFSMSrmm completes the restore and sets a return code of 8 under these conditions:

- DFSMSrmm finds that the previous update of the control data set failed and the error was not corrected.
- DFSMSrmm cannot match the control data set and the journal. Journal copies used as input to forward recovery are out of sequence with other journal copies.
- DFSMSrmm can match the journal and the control data set, but the last set of journal records is incomplete. DFSMSrmm issues a message to indicate that forward recovery is completed to the point of the last complete set of journal records.
- DFSMSrmm detects discrepancies in the journal records used to perform a forward recovery.

For either a return code of 8 or 12, you should decide whether the restored control data set is acceptable, then take corrective actions. You can try to recover again by using the correct control data set and journal data sets. If you do not have any other backups or journals from which to restore, you will have to accept the restored control data set.

DFSMSrmm sets values in the control data set control record to indicate that the recovery did not complete satisfactorily. Until these values are reset, DFSMSrmm always prompts the operator at DFSMSrmm startup time to determine if the control data set can be used or not. To clear the values, run EDGUTIL VERIFY(ALL) to validate the recovered control data set. If processing completes successfully, DFSMSrmm resets the control data set control record and accepts the control data set for use without prompting the operator at future startup times.

If EDGUTIL finds errors in the restored control data set, correct the errors and run EDGUTIL again. Start DFSMSrmm and use the restored control data set while you issue RMM TSO subcommands. Correct the errors in the control data set by using RMM TSO subcommands or the EDGUTIL utility with the MEND parameter as described in “Mending the Control Data Set” on page 345 to update the information in the control data set.

Customizing the DSSOPT DD Statement

You can use the DSSOPT DD statement to replace the DFSMSdss DUMP and RESTORE commands that are used by DFSMSrmm. You might want to customize the operands based on the media that you used for the backup and the resource that you have available.

Related Reading: Refer to *z/OS DFSMSdss Storage Administration Reference* for information about the DFSMSdss DUMP command and RESTORE command.

Figure 123 shows examples of the DFSMSdss DUMP command and the DFSMSdss RESTORE command that DFSMSrmm issues. You can replace the second line of the DUMP and RESTORE commands that are shown in Figure 123 with one or more DFSMSdss options. If you decide to change the commands, specify all the operands you want to have processed because DFSMSrmm uses your input in place of its own.

```
DUMP DS(INCLUDE(cds_name)) OUTDD(BACKUP) SHARE -
   COMPRESS CONCURRENT VALIDATE OPTIMIZE(1)

RESTORE DS(INCLUDE(**)) INDD(BACKUP) -
    REPLACE
```

Figure 123. DFSMSdss Commands that are Issued by DFSMSrmm

The DUMP command and RESTORE command operands that you can specify in the DSSOPT DD statement are controlled and validated by DFSMSdss, not by DFSMSrmm. If you specify an unsupported command operand, DFSMSdss fails the dump operation. The REPLACE keyword on the RESTORE command ensures that when you recover the DFSMSrmm control data set from a backup, any existing control data set is reused when possible or reallocated if necessary. If you are increasing the size of the control data set and have preallocated a larger control data set, specify the REPLACE keyword to restore to the preallocated data set.

The DSSOPT DD statement can be specified for both dump and restore operations. DFSMSrmm reads all the records and uses them to replace its default command operands beginning at the second line. You can include comments in the DSSOPT records by using DFSMSdss conventions.

Example: Ensure that DFSMSdss does not compress the data and that the tape hardware is used to compress the records. This example uses EDGHSKP to back up to tape. To use EDGBKUP, the backup data sets must be allocated on DASD.

```
//EDGBKUP  EXEC PGM=EDGHSKP,PARM='BACKUP(DSS)'
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.CDS(+1),
//         LABEL=(,SL)
//JRNLBKUP DD DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.JRNL(+1),
//         LABEL=(2,SL),VOL=REF=*.BACKUP
//DSSOPT   DD *
   CONCURRENT OPTIMIZE(4) VALIDATE
/*
```

Example: Rename the control data set during restore processing. The restored control data set is renamed when DFSMSdss renames each of the components of the backup copy of the original DFSMSrmm control data set. The example also shows how to restore the control data set and forward recover it to a volume different from the volume on which it was backed up. When the control data set is renamed during the restore, do not specify the MASTER DD statement unless it identifies the predefined target data set.

```
//EDGBKUP  EXEC PGM=EDGBKUP,PARM='RESTORE'
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=SHR,UNIT=TAPE,DSN=BACKUP.CDS(0)
//JOURNAL  DD DISP=SHR,DSN=BACKUP.JOURNAL(0)
//         DD DISP=SHR,DSN=RMM.JOURNAL
//NEWVOL   DD DISP=SHR,VOL=SER=MYVOLX
//DSSOPT   DD *
/*RESTORE TO NEW VOLUME,AND
RENAME THE CDS */
OUTDD(NEWVOL)RENAMEU((*.CDS,*.NEWCDS))
```

Backing Up the Control Data Set

Requirement: To use DFSMSdss for non-intrusive backup, set up the DFSMSdss concurrent copy environment or the virtual concurrent environment. You must have the hardware and software required to establish a concurrent copy session or a virtual concurrent copy session.

Use EDGBKUP to perform the following tasks:

- Back up the control data set to a non-VSAM data set or a VSAM cluster that is described in “Backing Up the DFSMSrmm Control Data Set and Journal.”
- Back up the journal described in “Backing Up the Journal” on page 313.
- Restore the control data set by returning a backup of the control data set and, optionally, forward recover it described in “Restoring the Control Data Set” on page 325.
- Reorganize the control data set described in “Reorganizing the Control Data Set” on page 329.
- Move the control data set and journal to different device types that are described in “Moving the Control Data Set and Journal to a Different Device” on page 332.

You can back up and restore the control data set using EDGBKUP with the DFSMSdss DUMP command and the RESTORE command or the access method services REPRO command. Use the DFSMSdss DUMP command with concurrent copy and the CONCURRENT option to perform a non-intrusive back up where DFSMSrmm allows updates to the DFSMSrmm control data set and journal. EDGBKUP links to IDCAMS or ADRDSSU to restore the control data set or back up the control data set. EDGBKUP links to IDCAMS to back up the journal data set.

You can see the commands and the messages that DFSMSrmm issues during processing in the SYSPRINT data set.

You cannot selectively copy, edit, or sort records in the DFSMSrmm journal data set or in backup copies because the records are in a format that only DFSMSrmm understands.

Backing Up the DFSMSrmm Control Data Set and Journal

When DFSMSrmm is active, you do not need to specify the MASTER and JOURNAL DD statements. EDGBKUP obtains the name of the control data set and the journal from the DFSMSrmm subsystem. When the DFSMSrmm subsystem is stopped, quiesced, or when you want to back up a control data set that is not in use by DFSMSrmm, specify the MASTER DD statement and the JOURNAL DD statement. You decide which data sets are backed up. Specify the BACKUP DD to request control data set backup. Specify the JRNLBKUP DD to request journal backup. You can specify either or both of the BACKUP and JRNLBKUP DD statements. If you do not back up the journal before it is cleared, forward recovery from previous control data set backups is limited. EDGBKUP does not reset the journal after backing up the control data set. Use EDGHSKP to reset the journal after backing up the control data set.

Example: Back up the control data set by using EDGBKUP.

```
//EDGBKP   EXEC PGM=EDGBKUP,PARM='BACKUP'
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=(,CATLG),UNIT=SYSDA,DSN=BACKUP.CDS(+1),
//         SPACE=(TRK,(ppp,sss),RLSE)
```

Example: Back up the journal set by using EDGBKUP.

```
//EDGBKP   EXEC PGM=EDGBKUP,PARM='BACKUP'
//SYSPRINT DD SYSOUT=*
//JRNLBKUP DD DISP=(,CATLG),UNIT=SYSDA,DSN=BACKUP.JRNL(+1),
//         SPACE=(TRK,(ppp,sss),RLSE)
```

DFSMSrmm checks that any previous update of the control data set has completed successfully before backing up the DFSMSrmm control data set. DFSMSrmm does not back up the control data set when the previous update of the control data set fails and you are not using BACKUP(REORG). DFSMSrmm issues an informational message and sets a return code of 12.

To store backup copies of the control data set in a storage location to prepare for disaster recovery, you can place the backups on tape and define vital record specifications to move the backups off-site. When BACKUP(DSS) is used, you can back up directly to tape. When you use BACKUP(NREORG) or BACKUP(REORG), use a DASD data set for the initial backup copy. Then copy the backup copy to a tape data set in a subsequent job step or job. For disaster recovery, you can restore directly from the tape data set with the DFSMSrmm subsystem stopped or quiesced as long as you use the EDGBKUP utility. For on-site recovery, use a DASD backup for faster recovery.

Restoring the Control Data Set

This section includes information about these topics:
- “Restoring the Control Data Set with Forward Recovery”
- “Restoring the Control Data Set without Forward Recovery” on page 327
- “Forward Recovering the Control Data Set” on page 327
- “Restoring the Control Data Set at a Recovery Site” on page 327
- “Using Non-DFSMSrmm Utilities to Restore the Control Data Set” on page 328

Restoring the Control Data Set with Forward Recovery

Figure 124 on page 326 shows the JCL for restoring the DFSMSrmm control data set and for forward recovering the DFSMSrmm control data set. You can restore the control data set directly from tape by using the EDGBKUP utility when DFSMSrmm is stopped or quiesced. You do not need to use the EDGRESET utility or the MODIFY command to reset DFSMSrmm. EDGBKUP uses either the DFSMSdss

RESTORE command or the AMS REPRO utility to restore the control data set based on the contents of the control data set backup.

```
//EDGBKUP  EXEC PGM=EDGBKUP,PARM='RESTORE'
//SYSPRINT DD   SYSOUT=*
//MASTER   DD   DISP=SHR,DSN=RMM.CDS
//*
//BACKUP   DD   DISP=SHR,DSN=BACKUP.CDS(0)
//*
//JOURNAL  DD   DISP=SHR,DSN=BACKUP.JRNL(0)
//         DD   DISP=SHR,DSN=RMM.JOURNAL
```

Figure 124. Restoring the Control Data Set with Forward Recovery

You must specify the MASTER DD statement to restore from an AMS REPRO backup and to forward recover an existing control data set.

The MASTER DD is optional when you restore the control data set from a DFSMSdss backup. In this way, you do not have to pre-allocate the control data set, and you can optionally restore to a control data set by using a different data set name. After the DFSMSdss restore is completed, DFSMSrmm dynamically allocates the MASTER file to the restored data set prior to starting forward recovery. If the MASTER DD statement is specified, the data set name must match the data set name that DFSMSdss restores.

You can restore the DFSMSrmm control data set and forward recover the restored control data set with journal records. Use the JOURNAL DD statement to concatenate multiple journal data sets. If you forward recover using multiple journal data sets, concatenate the journal data sets in the order in which changes were originally made. The first journal data set in the concatenation must match the control data set to be forward recovered. For control data set backups that are taken with AMS REPRO, DFSMSrmm considers the matching journal to be the one that contains records that are created after the journal was cleared. For control data set backups that are taken with DFSMSdss, DFSMSrmm considers the matching journal to be the journal in use at the time the backup is taken.

Example: The example shows how to restore from the second generation backup. Concatenate journal data sets for back up using AMS REPRO.

```
//BACKUP   DD      DISP=SHR,DSN=BACKUP.CDS(-2)
//JOURNAL  DD      DISP=SHR,DSN=BACKUP.JOURNAL(-1)
//         DD      DISP=SHR,DSN=BACKUP.JOURNAL(0)
//         DD      DISP=SHR,DSN=RMM.JOURNAL
```

Example: The example shows how to restore from the second generation backup. Concatenate journal data sets for back up using DFSMSdss.

```
//BACKUP   DD      DISP=SHR,DSN=BACKUP.CDS(-2)
//JOURNAL  DD      DISP=SHR,DSN=BACKUP.JOURNAL(-2)
//         DD      DISP=SHR,DSN=BACKUP.JOURNAL(-1)
//         DD      DISP=SHR,DSN=BACKUP.JOURNAL(0)
//         DD      DISP=SHR,DSN=RMM.JOURNAL
```

When backup is taken using AMS REPRO and you restore the control data set from the latest control data set backup, use the active journal to forward recover to the latest point in time.

When backup is taken using BACKUP(DSS) and you restore from the latest control data set backup, both the latest journal backup and the active journal must be used for forward recovery. For information about restoring the control data set at a recovery site, see “Restoring the Control Data Set at a Recovery Site.”

Restoring the Control Data Set without Forward Recovery

To restore the control data set without forward recovery, do not specify a JOURNAL DD statement. EDGBKUP checks the contents of the control data set backup and calls the correct utility to restore the control data set. The MASTER DD statement is optional when you have used DFSMSdss in the DFSMSrmm utilities to create the backup copy and you do not need to change the name of the DFSMSrmm control data set.

Example: To restore the control data set without forward recovery:

```
//RESTORE  EXEC PGM=EDGBKUP,PARM='RESTORE'
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=SHR,DSN=RMM.BACKUP.CDS
//MASTER   DD DISP=SHR,DSN=RMM.CDS
```

Forward Recovering the Control Data Set

You can forward recover the control data set after you have restored the control data set in a previous step.

Example: To forward recover the control data set by using EDGBKUP:

```
//FWDRECVR EXEC PGM=EDGBKUP,PARM='RESTORE'
//SYSPRINT DD SYSOUT=*
//MASTER   DD DISP=SHR,DSN=RMM.CDS
//JOURNAL  DD DISP=SHR,DSN=RMM.JOURNAL
```

When you do not provide the BACKUP DD statement, DFSMSrmm does not restore the control data set. DFSMSrmm assumes that the data set identified by the MASTER DD statement identifies the correct control data set, and EDGBKUP uses data from the journal to forward recover the control data set.

You can use the JOURNAL DD statement to concatenate multiple journal data sets. See “Restoring the Control Data Set with Forward Recovery” on page 325 for information about concatenating journal data for recovery. You can forward recover a restored or unrestored control data set. For example, if automatic forward recovery fails, you could attempt forward recovery by using the active journal. If you have already restored the control data set but have not used journal data for forward recovery, run forward recovery as a second step. The DFSMSrmm subsystem must be stopped or quiesced during forward recovery of the active control data set.

Restoring the Control Data Set at a Recovery Site

Before you start the DFRMM procedure, use the EDGBKUP utility to restore from the latest control data set backup. You can restore from either tape or DASD. When you restore from tape, even though the DFRMM procedure is not started, DFSMSrmm verifies that you are restoring a backup of the DFSMSrmm control data set and allows you to use EDGBKUP with DFSMSrmm inactive. Any backup of the control data set is a backup consistent with the time that you started the backup. You can restore that backup copy to your selected point in time, or can optionally forward recover to the end of the DFSMSrmm backup processing or some later time as long as you have the journal backups and journal data set. Usually you will not use forward recovery. To restore the control data set at a recovery site, follow these steps:

1. If you used DFSMSrmm to create the backup by using IDCAMS REPRO, allocate a new control data set. You can use the DFSMSrmm-supplied sample job EDGJMFAL to allocate a new control data set.
2. Run EDGBKUP with the RESTORE parameter to restore the control data set from a backup copy. The MASTER DD statement is optional if you used DFSMSrmm to create the backup with DFSMSdss.

```
//RESTORE  EXEC PGM=EDGBKUP,PARM='RESTORE'
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=SHR,DSN=RMM.BACKUP.CDS
//MASTER   DD DISP=SHR,DSN=RMM.CDS
```

3. If you want to forward recover to a later point in time, run EDGBKUP with the RESTORE parameter and include the JOURNAL DD statement to identify the journals to be applied.

```
//RESTORE  EXEC PGM=EDGBKUP,PARM='RESTORE'
//SYSPRINT DD SYSOUT=*
//MASTER   DD DISP=SHR,DSN=RMM.CDS
//JOURNAL  DD DISP=SHR,DSN=RMM.JOURNAL.BACKUP(0)
```

   This step can be combined with Step 2 to restore the control data set and journal at the same time:

```
//RESTORE  EXEC PGM=EDGBKUP,PARM='RESTORE'
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=SHR,DSN=RMM.BACKUP.CDS
//MASTER   DD DISP=SHR,DSN=RMM.CDS
//JOURNAL  DD DISP=SHR,DSN=RMM.JOURNAL.BACKUP(0)
```

   The MASTER DD statement is optional if you used DFSMSrmm to create the backup with DFSMSdss.
4. Allocate an empty journal data set. DFSMSrmm provides sample EDGJNLAL that you can use to allocate the journal.
5. Start DFRMM with a DFSMSrmm EDGRMM*xx* parmlib member that names the restored control data set and the empty journal.

Using Non-DFSMSrmm Utilities to Restore the Control Data Set

Recommendation: Use the DFSMSrmm EDGBKUP utility to restore the control data set to get the benefits that the DFSMSrmm utility provides and to avoid replying to operator messages if DFSMSrmm is not active.

You can use IDCAMS REPRO or DFSMSdss RESTORE to recover the DFSMSrmm control data set from a backup copy taken using those utilities. This section contains examples for you to use IDCAMS and DFSMSdss.

If you use IDCAMS or DFSMSdss to restore the control data set from a backup, you must forward recover the control data set before it can be used by DFSMSrmm. If you do not perform forward recovery and start DFSMSrmm, you will have to reply to message EDG0123D. To avoid this, forward recover the control data set by using the EDGBKUP utility. You can use a dummy journal if you do not have an existing journal or journal backups. When forward recovery is completed, you are ready to use the control data set with DFSMSrmm.

Example: Restore the control data set by using IDCAMS.

```
//RESTORE  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=SHR,DSN=RMM.BACKUP.CDS
//MASTER   DD DISP=SHR,DSN=RMM.CDS
//SYSIN    DD *
 REPRO INFILE(BACKUP) OUTFILE(MASTER)
/*
```

Example: Restore the control data set using DFSMSdss.

```
//RESTORE  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=SHR,DSN=RMM.BACKUP.CDS
//SYSIN    DD *
 RESTORE DS(INCLUDE(**))INDD(BACKUP) REPLACE
/*
```

Example: Forward recover the control data set using a dummy journal.

```
//FWDRECVR EXEC PGM=EDGBKUP,PARM='RESTORE'
//SYSPRINT DD SYSOUT=*
//MASTER DD DISP=SHR,DSN=RMM.CDS
//JOURNAL DD DUMMY.
```

Reorganizing the Control Data Set

Use EDGBKUP to reorganize the control data set. The DFSMSrmm subsystem must be stopped or quiesced when you reorganize the active control data set. DFSMSrmm reorganizes the control data set by first backing up the control data set and optionally the journal. Then DFSMSrmm restores the control data set from the backup control data set.

Do not use the BACKUP(DSS) parameter to reorganize the control data set. If you use BACKUP(DSS) to regularly back up for recovery purposes, use the EDGBKUP utility with the BACKUP(REORG) parameter with care. Although DFSMSrmm can determine which utility to use for recovery from any backup taken using an DFSMSrmm utility, any backup taken during BACKUP(REORG) might impact the cycles that have been maintained of regular backup copies.

Example: Reorganize the control data set without backing up the journal.

```
//REORG     EXEC PGM=EDGBKUP,PARM='BACKUP(REORG)'
//SYSPRINT DD   SYSOUT=*
//BACKUP   DD   DISP=(,CATLG),DSN=RMM.BACKUP.CDS(+1),UNIT=SYSDA,
//       SPACE=(CYL,(ppp,sss),,RLSE)
//MASTER   DD   DISP=OLD,DSN=RMM.CDS
```

Example: Reorganize the control data set and back up the journal. DFSMSrmm backs up the journal before attempting to restore the control data set because the JRNLBKUP DD and the JOURNAL DD statement are specified in this example.

```
//REORG    EXEC PGM=EDGBKUP,PARM='BACKUP(REORG)'
//SYSPRINT DD   SYSOUT=*
//BACKUP   DD   DISP=(,CATLG),DSN=RMM.BACKUP.CDS(+1),UNIT=SYSDA,
//              SPACE=(CYL,(ppp,sss),,RLSE)
//MASTER   DD   DISP=OLD,DSN=RMM.CDS
//JRNLBKUP DD   DISP=(,CATLG),DSN=RMM.BACKUP.JOURNAL(+1),UNIT=SYSDA,
//              SPACE=(CYL,(ppp,sss),,RLSE)
//JOURNAL  DD   DISP=OLD,DSN=RMM.JOURNAL
```

Monitoring the Space Used by the Control Data Set

See “Step 12: Creating the DFSMSrmm Control Data Set” on page 36 for information on creating the DFSMSrmm control data set, including information on how to calculate DASD space for the control data set, placement of the control data set, and allocating space for the control data set. We recommend that you allocate the DFSMSrmm control data set with enough secondary space so that the DFSMSrmm control data set can grow as you add new resources such as volumes and data sets. Also, allow for enough planned free space in the DFSMSrmm control data set to allow for growth and any known new-volume requirements. However, at some time, the allocated space in the DFSMSrmm control data set will fill up and new extents are allocated and used as long as there is free DASD space available. The DFSMSrmm control data set can grow in size within the limits imposed by VSAM and DFSMS.

Use the LISTCAT output from IDCAMS to check the details of the DFSMSrmm control data set and the RMM LISTCONTROL CNTL subcommand to list the calculated percentage used for the DFSMSrmm control data set. DFSMSrmm calculates the percentage of the DFSMSrmm control data set used by checking the High Used RBA and High Allocated RBA that the LISTCAT output shows. This percentage does not take into account any available embedded free space, nor does it take into account the number of used extents or the additional free space that may be available to extend the DFSMSrmm control data set.

Decide how you will monitor and report on the DFSMSrmm control data set space used. By using the information available to you, including your predicted tape usage, decide if the DFSMSrmm control data set is large enough or needs to be reallocated. Decide on your own thresholds, and check regularly to ensure they are used to trigger corrective action. Although you can reorganize the DFSMSrmm control data set to recover free space, do not do this regularly. Ensure your DFSMSrmm control data set is large enough and can grow as required to avoid the need to reorganize it regularly.

Changing the Size of the Control Data Set And Journal

To change the size of the DFSMSrmm control data set or journal (either to make the data sets larger or smaller), use the procedures documented in this chapter. The basic procedure is to use backup and recovery, but the method selected depends on whether you have a new volume available for the new control data set. If you use the existing volume and must delete the existing data set to allocate one of a new size, follow the procedure documented in “Backing Up the DFSMSrmm Control Data Set and Journal” on page 324 and “Restoring the Control Data Set with Forward Recovery” on page 325. If you use a new volume, follow the procedure documented in “Moving the Control Data Set and Journal to a Different Device” on page 332. When you allocate the new data sets, make them the required size by either increasing or decreasing the size based on your requirements.

|

Recovering from Control Data Set Update Failures

Information about a volume in the DFSMSrmm control data set is stored in several records. A change in a volume's status can require updating several records. Since an update of multiple records takes a certain amount of time, it is possible that external events, such as a power outage, could interrupt the updating process. An interruption can leave some records that reflect the volume's old status and some its new and the control data set is then considered corrupt. It might be impossible to subsequently change the volume's status. This situation is referred to as a *multi-record update failure*.

DFSMSrmm detects a multi-record update failure at DFSMSrmm address-space start-up time when the control data set is not shared between systems. When a control data set is shared between multiple systems, the multi-record update failure can be detected at any time. A hardware failure on any of the systems could be detected by another system at the next I/O to the control data set. In this case, a SYSTEM RESET of the failing system releases the control data set reserve, and the other systems sharing that control data set, find they had experienced a multi-record update failure.

Recovery Processing

The first request for I/O to the control data set, following a multi-record update failure, detects the failure, and recovery processing begins. Other attempts to access the control data set are queued behind the current request until either manual recovery is required or automatic recovery is successful.

When a multi-record update failure is first detected, DFSMSrmm attempts automatic recovery processing which requires no operator intervention. If a journal data set matching the corrupt control data set is found, DFSMSrmm issues message EDG2111I to notify the operator that automatic recovery processing is starting. DFSMSrmm issues message EDG2115I if automatic recovery is not possible and manual recovery is required. Manual recovery might be needed under these conditions:

- The journal data set is not defined in the initialization parameters
- The journal and control data sets do not match
- The journal data set has been disabled in response to message EDG2103D
- The journal data set update has been ignored in response to message EDG2103D

Manual recovery requires operator intervention. See *z/OS DFSMSrmm Guide and Reference* for operator procedures for responding to messages DFSMSrmm issues during recovery processing.

Recommendation: Use EDGHSKP with the CATSYNCH parameter to synchronize catalogs after recovery.

Handling I/O Requests Following a Failure

After recovery is completed, I/O requests for any tape processing activity and DFSMShsm tape volume release activity are retried.

Automatic recovery

Once automatic recovery is successful the first request continues as normal, and the queued requests are processed.

Manual recovery

When manual recovery is required, each subsystem request that results in I/O to the control data set fails. For requests that can be retried, DFSMSrmm issues WTOR EDG4001D or EDG8008D.

When you perform manual recovery because the control data set is no longer valid, restore the control data set using the most recent backup copy of the control data set. Forward recover the control data set to the point of failure by using the latest control data set backup and the applicable journal backups that are concatenated with the active journal. See “Restoring the Control Data Set with Forward Recovery” on page 325 for information. If manual recovery is required because the control data set is full or for some other reason where the control data set information is valid except for the most recent failed record updates, use the current control data set for recovery rather than a backup copy. When you use the current control data set for recovery, you do not need to perform forward recovery because DFSMSrmm restart performs automatic forward recovery. Also do not run the EDGUTIL utility against the control data set before you resume DFSMSrmm operation.

Use one of the following methods to perform manual recovery by using the current control data set.

- Run the EDGBKUP utility with PARM=’RESTORE’ to use the active journal to forward recover the current DFSMSrmm control data set. If processing is successful, the control data set information is correct. See “Forward Recovering the Control Data Set” on page 327.
- Run the EDGBKUP utility with PARM=’BACKUP(REORG)’ to reorganize the current control data set and reclaim enough free space to enable processing to continue. See “Reorganizing the Control Data Set” on page 329. To correct the control data set information, refresh the DFSMSrmm started task, and attempt automatic forward recovery.
- Use IDCAMS REPRO to copy the contents of the current control data set to a new, larger control data set. Create a new EDGRMM*xx* parmlib member to define data set name for the new control data set. To correct the control data set information, refresh the DFSMSrmm started task, and attempt automatic forward recovery.
- Run EDGBKUP with PARM=’RESTORE’ to use the latest control data set backup, and the applicable journal backups that are concatenated with the active journal to forward recover. If processing is successful, the control data set information is correct. See “Restoring the Control Data Set with Forward Recovery” on page 325.

Manual recovery requires operator intervention. See *z/OS DFSMSrmm Guide and Reference* for operator procedures for information about stopping, quiescing, and restarting the DFSMSrmm subsystem. After manual recovery completes, the operator can reply ’RETRY’ or ’CANCEL’ to the EDG4001D and EDG8008D WTORs. The following requests fail and must be rerun or reissued after manual recovery:

- Inventory management in progress
- DFSMSrmm utilities EDGINERS and EDGUTIL
- RMM TSO subcommands

Moving the Control Data Set and Journal to a Different Device

Before You Begin: See “Step 18: Starting DFSMSrmm” on page 47 for information about the stopping or quiescing of DFSMSrmm to allow processing for restoring or reorganizing the control data set.

To move the control data set or journal to a different device, use DFSMSrmm utilities. For other DFSMSrmm data sets, such as the extract data set, use the existing storage management techniques that you are familiar with to move them.

The following topics provide examples for moving the control data set and journal data set that use both DFSMSrmm utilities and non-DFSMSrmm techniques. If you want to move the control data set and not the journal, modify the following examples as follows:

1. Do not allocate a new journal data set.
2. Only alter the control data set name. Do not alter the journal name or change the journal name in the DFSMSrmm EDGRMM*xx* parmlib.

Do not change the restore step to ensure that the restore process includes forward recovery from the journal records.

Steps for Moving the Control Data Set and Journal Using the DFSMSrmm EDGHSKP Utility with the PARM='BACKUP' Parameter

Perform the following steps to move your control data set and journal to a different device by using the EDGHSKP utility with the PARM='BACKUP' parameter.

1. Allocate a new control data set and journal.
2. Back up the control data set and the journal.

   Example: With DFSMSrmm active, run EDGHSKP,PARM='BACKUP' ' to back up the control data set and the journal and to clear the current journal.

   ```
   // EXEC PGM=EDGHSKP,PARM='BACKUP'
   //MESSAGE  DD DISP=SHR,DSN=messages
   //SYSPRINT DD SYSOUT=*
   //BACKUP   DD DISP=(,CATLG),DSN=cds backup(+1),UNIT=SYSDA
   //JRNLBKUP DD DISP=(,CATLG),DSN= journal backup(+1),UNIT=SYSDA
   ```

   This clears the current journal data set. Some records might be written to the journal if any updates are made to the control data set before the DFSMSrmm procedure is stopped. This is not a problem because the restore operation will use them in forward recovery.
3. Stop or quiesce the DFSMSrmm procedure to prevent any further updates to the control data set during recovery.
4. Restore the backup of the control data set to the new control data set allocated in step 1.

   Example: This JCL example puts the latest DFSMSrmm information into the new control data set and uses the old journal to forward recover the control data set backup to the point where the you stopped the DFSMSrmm procedure.

   ```
    //EXEC PGM=EDGBKUP,PARM='RESTORE'
   //SYSPRINT DD SYSOUT=*
   //BACKUP DD DISP=SHR,DSN=cds backup(0)
   //MASTER DD DISP=SHR,DSN=new control data set
   //JOURNAL DD DISP=SHR,DSN=old journal
   ```

5. Implement the new data sets by using one of these techniques:
   - Use IDCAMS ALTER command to rename the new control data set and new journal, after renaming the old data sets, or
   - Create a new EDGRMM*xx* parmlib member with the new names, or
   - Update the current parmlib member to include the names of the new control data set and journal.

6. If you stopped DFSMSrmm, start the DFSMSrmm procedure, using the updated parmlib member or the new parmlib member. If you quiesced DFSMSrmm, use the MODIFY command to specify the parmlib member suffix to be used.

   If you are keeping multiple control data set and journal backups for error recovery situations, perform step 2 on page 333 again to backup the control data set and journal.

   Recommendation: Back up the new data sets now to avoid keeping the old journal for recovery. As your backup copies are created in the future, your requirement for the old journal will be eliminated.

You are done when you have successfully moved the control data set and journal.

Steps for Moving the Control Data Set and Journal Using DFSMSrmm Utility EDGHSKP Utility with the PARM='BACKUP(DSS)' Parameter

Before you begin: Use the BACKUP(DSS) option on a concurrent copy capable device to enable DFSMSrmm to continue to process requests while the backup is taken.

Perform the following steps to move your control data set and journal to a different device.

1. With DFSMSrmm active, back up both the control data set and the journal.

   Example: Run EDGHSKP,PARM='BACKUP(DSS)' to back up the control data set and the journal and to clear the current journal. Use the BACKUP(DSS) option on a concurrent copy capable device to enable DFSMSrmm to continue processing requests while the backup is taken. DFSMSrmm might write some records to the journal if any updates are made to the control data set before the DFSMSrmm procedure is stopped. This is not a problem as the restore operation will use them in forward recovery.

```
//EXEC PGM=EDGHSKP,PARM='BACKUP (DSS)'
//MESSAGE DD DISP=SHR,DSN=messages
//SYSPRINT DD SYSOUT=*
//BACKUP DD DISP=(,CATLG),DSN=cds backup(+1),UNIT=SYSDA
//JRNLBKUP DD DISP=(,CATLG),DSN=journal backup(+1),UNIT=SYSDA
```

2. Stop or quiesce the DFSMSrmm procedure to prevent any further updates to the control data set during recovery.
3. Restore the backup of the control data set to the new control data set. If the BACKUP(DSS) option is used, you can use the DSSOPT DD statement to specify the new control data set data set name. This puts the latest DFSMSrmm information into the new control data set and uses the old journal to forward recover the control data set backup to the point when the you stopped the DFSMSrmm procedure.

   Example: This JCL example restores the back up to a different device using a new data set name.

```
//MOVECDS EXEC PGM=EDGBKUP,PARM='RESTORE'
//SYSPRINT DD SYSOUT=*
//DSSOPT DD *
   RENAMEU(*.CDS,*.NEWCDS) OUTDYNAM(SHRPK2) NULLSTORCLAS BYPASSACS(*)
//BACKUP DD DISP=SHR,DSN=cds_backup(0)
//JOURNAL DD DISP=SHR,DSN=journal_backup(0)
//        DD DISP=SHR,DSN=old_journal
```

4. Implement the new data sets by using one of these techniques:

   - Use IDCAMS ALTER command to rename the new control data set and journal after you rename the old data sets.
   - Create a new EDGRMM*xx* parmlib member with the new journal names and control data set names.
   - Update the current parmlib member to include the names of the new control data set and journal.
5. If you stopped DFSMSrmm, start the DFSMSrmm procedure, using the updated parmlib member or the new parmlib member. If you quiesced DFSMSrmm, use the MODIFY command to specify the parmlib member suffix to be used.

   If you are keeping multiple control data set and journal backups for error recovery situations, perform step 1 on page 334 again to backup the control data set and journal.

   Recommendation: Back up the new data sets now to avoid the requirement to keep the old journal for recovery. As your backup copies are created in the future, your requirement for the old journal will be eliminated.

You are done when you have successfully moved the control data set and journal.

Moving the Journal using DFSMSrmm Utilities

The journal cannot, strictly speaking, be moved. You move it by allocating a new journal data set and later deleting the old one.

To move your journal to a different device, follow this procedure:

1. Allocate a new journal data set.
2. Stop or quiesce the DFSMSrmm procedure to prevent any further updates to the control data set and journal.
3. With DFSMSrmm stopped or quiesced, run EDGBKUP,PARM='BACKUP' to backup both the control data set and journal as shown in Figure 125.

```
// EXEC PGM=EDGBKUP,PARM='BACKUP'
//SYSPRINT DD SYSOUT=*
//MASTER   DD DISP=SHR,DSN=cds name
//JOURNAL  DD DISP=SHR,DSN=journal name
//BACKUP   DD DISP=(,CATLG),DSN=cds backup(+1),UNIT=SYSDA
//JRNLBKUP DD DISP=(,CATLG),DSN= journal backup(+1),UNIT=SYSDA
```

Figure 125. JCL Example for Backing Up the Control Data Set and Journal

   This provides you with a valid point-in-time backup of the control data set and the old journal data set. After this step, if you need to recover the control data set, you can use this control data set backup, and the new journal data set.Optionally you can back up just the journal by removing the BACKUP DD statement. Because you are only moving the journal, you only need to back up the journal.
4. Implement the journal by using one of these techniques:
   - Use IDCAMS ALTER command to rename the new journal, having first renamed the old data set, or
   - Create a new EDGRMM*xx* parmlib member, or
   - Update the current parmlib member to include the name of the new journal.
5. If DFSMSrmm was stopped, start the DFSMSrmm procedure, using the current, updated, or the new parmlib member. If DFSMSrmm was quiesced, use the MODIFY command to specify the parmlib member suffix to be used. Doing so, uses the unmoved control data set and the newly allocated journal.

6. Delete the old journal.

Steps for Moving the Control Data Set using Non-DFSMSrmm Utilities

Perform the following steps to move your control data set to a different device by using non-DFSMSrmm utilities such as AMS REPRO and EXPORT/IMPORT, or DFSMSdss ADRDSSU.

1. Allocate a new control data set only if you are planning to use AMS REPRO.
2. Stop or quiesce the DFSMSrmm procedure to prevent any further updates to the control data set during recovery.
3. Copy the old control data set to the new control data set by using one of the following utilities:
   - AMS REPRO.
   - AMS EXPORT followed by IMPORT.
   - DFSMSdss ADRDSSU utility.

     Example: Copy the old control data set to the new control data set by using the DFSMSdss ADRDSSU utility.

     ```
     //COPYCDS  EXEC PGM=ADRDSSU,REGION=8M
     //SYSPRINT DD SYSOUT=*
     //SYSIN    DD *
             COPY -
             LOGINDYNAM (SHRPK3) -
             DS(INC(RMM.CDS)) -
             OUTDYNAM(SHRPK2) -
             SPHERE -
             RENAMEU(*.CDS,*.NEWCDS)
     ```
4. Implement the data sets by using one of these techniques:
   - Use IDCAMS ALTER command to rename the new control data set, having first renamed the old data set, or
   - Create a new EDGRMM*xx* parmlib member, or
   - Update the current parmlib member to include the name of the new control data set.
5. If you stopped DFSMSrmm, start the DFSMSrmm procedure, using the updated parmlib member or the new parmlib member. If you quiesced DFSMSrmm, use the MODIFY command to specify the parmlib member suffix to be used. This will use the moved control data set and the unmoved journal.

Using EDGUTIL for Tasks Such as Creating and Verifying the Control Data Set

Use EDGUTIL to perform these tasks:

- Create the control data set control record in an empty VSAM data set described in “Creating or Updating the Control Data Set Control Record” on page 341.
- Update an existing control data set control record described in “Creating or Updating the Control Data Set Control Record” on page 341.
- Verify control data set information to diagnose errors in the control data set described in “Verifying the Contents of the Control Data Set” on page 343.
- Check that control data set information about a volume’s status and the library in which it resides are consistent with the TCDB information, and optionally the library manager database information described in “Verifying the Control Data Set and Tape Configuration Database” on page 345.

- Synchronize the TCDB information and library manager information for system-managed volumes with information in the DFSMSrmm control data set that is described in “Synchronizing the Contents of the Control Data Set” on page 345
- Detect consistency errors in control data set information created during conversion activities and fix the errors that were detected. Run the mend function against a VSAM copy of the control data set as the first step in fixing a production control data set. The mend function should only be used with guidance from IBM Support Center to determine the underlying cause of any errors in the control data set. See “Mending the Control Data Set” on page 345 for restrictions that you should be aware of when you use the mend function.
- Enable selected functions:
  – “Setting up DFSMSrmm Stacked Volume Support” on page 346
  – “Enabling Extended Bin Support” on page 347

JCL for EDGUTIL

This section provides JCL for creating the DFSMSrmm control data set, verifying the contents of the control data set, updating the control data set, and mending the control data set. The MASTER DD statement is required.

JCL for Creating the Control Data Set

The JCL shown in Figure 126 can be used to create a control data set.

```
//EDGUTIL EXEC PGM=EDGUTIL,PARM='CREATE'
//SYSPRINT DD SYSOUT=*
//MASTER   DD DSN=RMM.CONTROL.DSET,DISP=SHR
//SYSIN    DD *
CONTROL
/*
```

Figure 126. Creating the Control Data Set

JCL for Updating the Control Data Set

Use the sample JCL in Figure 127 to update the control data set.

```
//UTIL     EXEC     PGM=EDGUTIL,PARM='UPDATE'
//SYSPRINT DD       SYSOUT=*
//MASTER   DD       DISP=SHR,DSN=RMM.CONTROL.DSET
//SYSIN    DD       *
CONTROL  RACKFREE(1234)
/*
```

Figure 127. Updating the Control Data Set

JCL for Verifying the Contents of the Control Data Set

Use the sample JCL in Figure 128 to verify the contents of the control data set.

```
//UTIL     EXEC     PGM=EDGUTIL,PARM='VERIFY(ALL)'
//SYSPRINT DD       utility message data set
//MASTER   DD       DSN=RMM.CONTROL.DSET,DISP=SHR
//SYSIN    DD       DUMMY
```

Figure 128. Verifying the Contents of the Control Data Set

JCL for Mending the Control Data Set

Use the sample JCL in Figure 129 to mend the control data set.

```
//UTIL      EXEC PGM=EDGUTIL,PARM='MEND'
//SYSPRINT  DD   SYSOUT=*
//MASTER    DD   DSN=RMM.CONTROL.DSET,DISP=SHR
//SYSIN     DD   DUMMY
/*
```

Figure 129. Mending the Control Data Set

EXEC Parameters for EDGUTIL

Figure 130 shows the EXEC parameters for EDGUTIL.

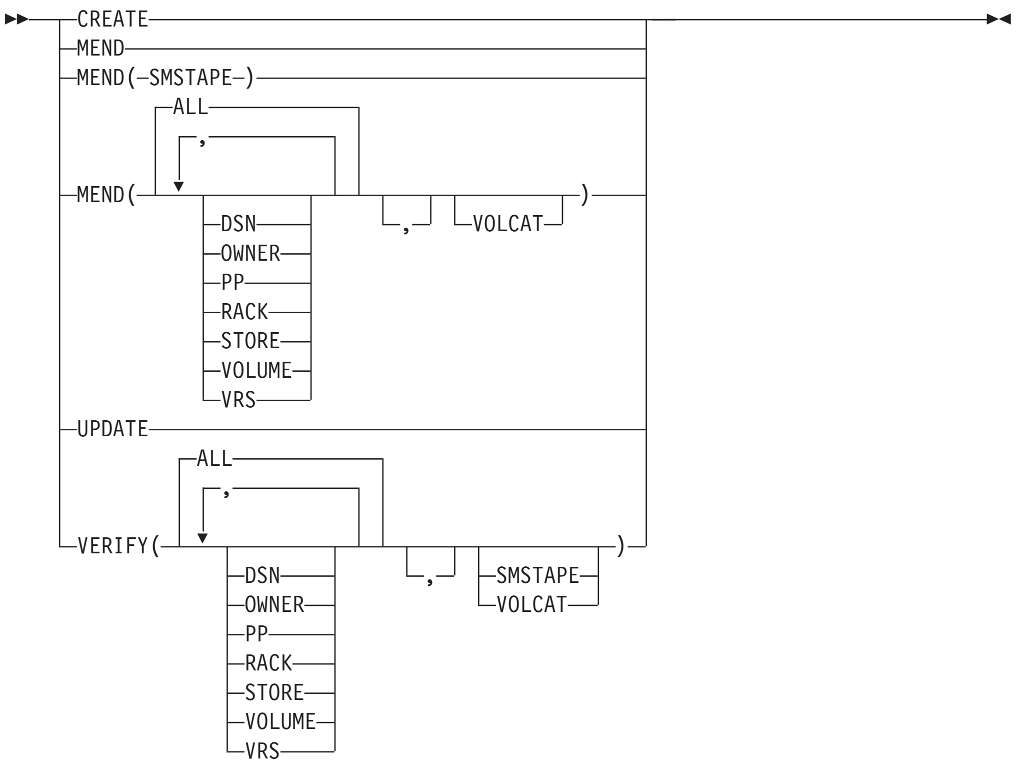


Figure 130. EDGUTIL EXEC Parameters

CREATE
Use CREATE to create a new control data set control record.

MEND
Use MEND to detect and fix errors in your control data set. If you have system-managed volumes, DFSMSrmm also uses information from the TCDB and the library manager database. The errors you encounter might have been created during conversion activities or as a result of system failures. Use the MEND function only with guidance from the IBM Support Center or to update the control data set once stacked volumes support is enabled. Run MEND on an unused control data set or with DFSMSrmm inactive. See “Mending the Control Data Set” on page 345 for more information.

MEND(SMSTAPE)
Use MEND(SMSTAPE) to update the TCDB and the library manager database using information from the DFSMSrmm control data set. Use MEND to update

the DFSMSrmm control data set based on information from the TCDB and the library manager database. Before running MEND(SMSTAPE), you should first use the VERIFY(SMSTAPE) option to find information that is not the same in the DFSMSrmm control data set, TCDB, and the library manager database. After running VERIFY(SMSTAPE) and before running MEND(SMSTAPE), you can update the control data set using DFSMSrmm TSO subcommands to correct DFSMSrmm information.

MEND(ALL,DSN,OWNER,PP,RACK,STORE,VOLUME,VRS,VOLCAT)
Use MEND to correct errors in the control data set. You can correct all the control data set information or select specific types of information.

The values you can specify on MEND are:

ALL
DFSMSrmm uses control data set information only and corrects all control data set information at once. No processing of the TCDB or library manager is performed.

DSN
DFSMSrmm corrects data set information based on comparing data set information to volume information.

OWNER
DFSMSrmm corrects owner information based on comparing owner information to volume information.

PP
DFSMSrmm corrects product information based on comparing software product information to volume and library shelf location information.

RACK
DFSMSrmm corrects rack number information based on comparing library shelf location information to volume information.

STORE
DFSMSrmm corrects bin number information based on comparing storage location shelf information to volume information.

VOLCAT
DFSMSrmm compares volume status and library name information in its control data set with the same information in the TCDB. If the information is different, DFSMSrmm corrects the control data set information.

VOLUME
DFSMSrmm corrects volume information based on comparing information about data sets, software products, owners, and shelf locations in the library and storage locations.

VRS
When correcting vital record specification errors, DFSMSrmm validates the next vital record specification information to see if a name vital record specification exists. If DFSMSrmm does not find a next vital record specification, DFSMSrmm fixes the information about the next vital record specification.

UPDATE
Use UPDATE to:

- Update an existing control data set control record.
- Mark the DFSMSrmm control data set as synchronized or not synchronized with the user catalogs so the control data set is synchronized at a later time.

- Enable extended bin support.

VERIFY(ALL,DSN,OWNER,PP,RACK,SMSTAPE,STORE,VOLUME,VRS,VOLCAT)
Use VERIFY to verify the information in the control data set and identify errors. You can verify all the information in the control data set at once or select specific values to verify individual pieces of information. If stacked volume support is enabled, DFSMSrmm checks the consistency of stacked volumes and the volumes in the stacked volumes.

To correct inconsistencies found during VERIFY processing, DFSMSrmm TSO subcommands to correct the inconsistencies. Use MEND(SMSTAPE) to drive changes to the TCDB and library manager database from the DFSMSrmm control data set or use access method services commands to correct errors in the TCDB.

The values you can specify on VERIFY are:

ALL
DFSMSrmm verifies all information at once, except for VOLCAT, which does consistency checking against the TCDB. ALL is the default.

DSN
DFSMSrmm validates data set information and compares data set information to volume information.

OWNER
DFSMSrmm validates owner information and compares owner information to volume information.

PP
DFSMSrmm validates product information and compares software product information to volume and library shelf location information.

RACK
DFSMSrmm validates rack number information and compares library shelf location information to volume information.

SMSTAPE
DFSMSrmm performs extra processing with the TCDB and library manager database when SMSTAPE is specified. DFSMSrmm scans both the DFSMSrmm control data set and the TCDB sequentially to find DFSMSrmm volumes that are not in the TCDB and TCDB volumes that are not in the DFSMSrmm control data set. DFSMSrmm also checks any volume that is found to be system-managed, either by definition to DFSMSrmm or retrieved from the TCDB, against the library manager database for an IBM automated tape library.

STORE
DFSMSrmm validates bin number information and compares storage location shelf information to volume information.

VOLCAT
DFSMSrmm compares volume status and library name information in its control data set with the same information in the TCDB. If the information is different, DFSMSrmm issues an information message, but sets a minimum return code of 0.

VOLUME
DFSMSrmm compares volume information to information about data sets, software products, owners, and shelf locations in the library and storage locations.

VRS

For vital record specification checking, DFSMSrmm validates the next vital record specification information to see if a name vital record specification exists. If DFSMSrmm does not find a next vital record specification, it issues an information message, but sets a minimum return code of 0.

Creating or Updating the Control Data Set Control Record

Create the control data set control record the first time you run EDGUTIL. This normally occurs during DFSMSrmm implementation or conversion to DFSMSrmm. The control data set control record contains information about the number of shelf locations in the library and storage locations. To create or update the control record, the user of EDGUTIL must have UPDATE or higher RACF access to the DFSMSrmm control data set.

Once you have created a control record, defined shelf locations to your installation, and have begun managing these shelves with DFSMSrmm, use the RMM ADDRACK subcommand to add shelf locations to DFSMSrmm. Do not use EDGUTIL to change the number of shelf locations.

You should only need to update the control record to correct rack or bin counts. DFSMSrmm lets you update the control data set and the control record only if inventory management and backup, restore and reorganize are not in progress.

As part of CREATE processing or UPDATE processing, EDGUTIL ensures that DFSMSrmm is not active and opens the control data set for load processing. The information provided on the CONTROL command in SYSIN is used to build a control record which is written to the control data set.

The SYSIN command in Figure 131 is required to create or update the control data set control record.

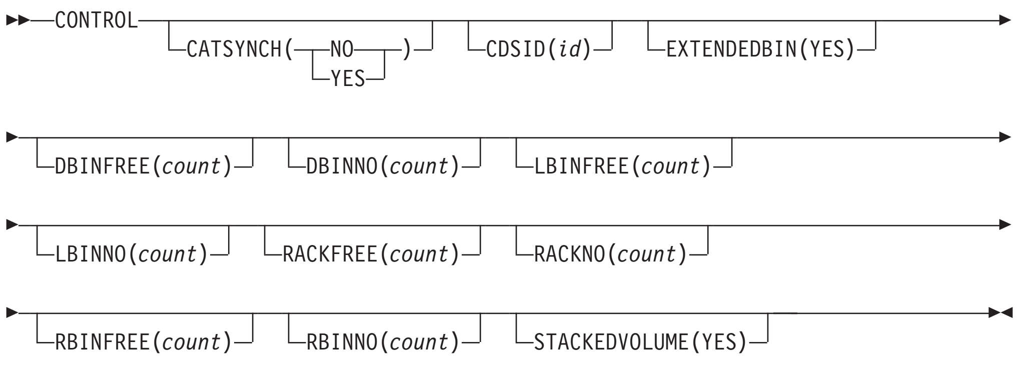


Figure 131. EDGUTIL SYSIN Commands

CATSYNCH(NO|YES)

Specifies whether or not the DFSMSrmm control data set and system catalogs are synchronized or not. When the DFSMSrmm control data set is not synchronized with the system catalogs, DFSMSrmm always retrieves catalog information to determine the catalog status of a data set.

When you indicate that catalogs are fully shared with the EDGRMM*xx* parmlib OPTION CATSYSID(*) as described in “Defining System Options: OPTION” on page 134, DFSMSrmm automatically marks the DFSMSrmm control data set as synchronized when you run the EDGHSKP utility with the CATSYNCH parameter as described in “EXEC Parameters for EDGHSKP” on page 283. You must use CATSYNCH(YES) to indicate to DFSMSrmm that catalogs are synchronized when you use DFSMSrmm with unshared catalogs. Do not specify CATSYNCH(YES) if you have not run the EDGHSKP utility with the CATSYNCH parameter on each system to synchronize the DFSMSrmm control data set with the user catalogs.

CATSYNCH(YES) sets the last synchronization date and time to the current date and time. CATSYNCH(NO) clears the last synchronization date and time. Specify CATSYNCH(NO) to force synchronization of the DFSMSrmm control data set and user catalogs the next time inventory management is run. DFSMSrmm cannot track catalog updates when the DFSMSrmm subsystem is stopped and issues messages when the updates cannot be made.

CDSID(*nnnnnn*)
Specifies one-to-eight alphanumeric characters that identify the control data set by name. There is no default.

At DFSMSrmm startup time, DFSMSrmm matches this CDSID value with the CDSID operand in parmlib member EDGRMM*xx*. The CDSID value in EDGUTIL will set or change the control data set ID. EDGUTIL does not validate the CDSID in the control data set control record; it simply sets the new value into the control record.

If you do not specify an ID for the control data set, you can set it the first time DFSMSrmm is started with this control data set by setting the CDSID operand in EDGRMM*xx*.

Do not change the CDSID value once you have set it, as changing it can affect inventory management. You can change the value only when no data sets exist that were created on the system or when the WHILECATALOG option for vital record specifications is not in use.

CONTROL
Specifies to update or create the control record.

DBINFREE(*nnnnnn*)
Specifies the number of empty bin numbers in the DISTANT storage location in a range from 0 to 999999.

DBINNO(*nnnnnn*)
Specifies the number of bin numbers in the DISTANT storage location in a range from 0 to 999999.

EXTENDEDBIN(YES)
Enables DFSMSrmm extended bin support, which allows the reuse of bins at the start of a move.

When extended bin support is enabled, DFSMSrmm records additional information in the volume and bin record while a volume is moving from or to a bin-managed storage location.

Do not enable extended bin support until you have made sure that the same level of code has been installed for all the DFSMSrmm systems that share a control data set.

Once it is enabled, extended bin support cannot be disabled.

Extended bin support must be enabled, if you want to use DFSMSrmm parmlib OPTION command REUSEBIN(STARTMOVE) operand to reuse bins when a volume moves from a bin.

LBINFREE(*nnnnnn*)
: Specifies the number of empty bin numbers in the LOCAL storage location in a range from 0 to 999999.

LBINNO(*nnnnnn*)
: Specifies the number of bin numbers in the LOCAL storage location in a range from 0 to 999999.

RACKFREE(*nnnnnn*)
: Specifies the number of empty rack numbers in the library for location SHELF and for system-managed libraries in a range from 0 to 2147483647.

RACKNO(*nnnnnn*)
: Specifies the number of rack numbers in the library for location SHELF and for system-managed libraries in a range from 0 to 2147483647.

RBINFREE(*nnnnnn*)
: Specifies the number of empty bin numbers in the REMOTE storage location in a range from 0 to 999999.

RBINNO(*nnnnnn*)
: Specifies the number of bin numbers in the REMOTE storage location in a range from 0 to 999999.

STACKEDVOLUME(YES)
: Enables DFSMSrmm stacked volume support. Prior to enabling stacked volume support, ensure all systems using the control data set are at the same release level.

When stacked volumes have been defined to DFSMSrmm but you have not enabled stacked volume support, DFSMSrmm manages the movement of the volumes that are in containers using the individual volume location. Volume movement is limited to non-shelf-managed storage locations. When you enable stacked volume support, DFSMSrmm uses the stacked volume records to manage the movement of the volumes contained in the stacked volumes. See “Setting up DFSMSrmm Stacked Volume Support” on page 346.

You cannot remove stacked volume support after it is enabled.

Verifying the Contents of the Control Data Set

Specify VERIFY on the EXEC parameter of EDGUTIL to verify the contents of the control data set. For VERIFY processing, EDGUTIL reads sequentially through the different record types in the control data set. The record types are identified by the VERIFY options you specify. For each record DFSMSrmm validates key fields and checks information with related records in the control data set. DFSMSrmm issues an informational message to the SYSPRINT file for each discrepancy that is identified. You can verify all the information in the control data set at once, or select specific values to verify individual pieces of information.

For example, if you specify the STORE value as shown in Figure 132, DFSMSrmm reads all the storage location shelf information in the control data set. DFSMSrmm then reads volume information for only those volumes in the storage locations and verifies that the volumes include the correct location information.

```
//UTIL     EXEC PGM=EDGUTIL,PARM='VERIFY(STORE)'
//SYSPRINT DD SYSOUT=*
//MASTER   DD DISP=SHR,DSN=RMM.CONTROL.DSET
```

Figure 132. Example of JCL for VERIFY(STORE)

When you specify VERIFY(ALL), and this completes successfully, DFSMSrmm resets the error indicator that is set when the control data set recovery processing was not successful. For VERIFY(VOLCAT), DFSMSrmm compares TCDB information with information in the DFSMSrmm control data set. For VERIFY(SMSTAPE), DFSMSrmm also retrieves the library manager information for each system-managed volume and compares the information to the TCDB and DFSMSrmm information. The function uses the DFSMSrmm control data set as the master and makes changes to the TCDB and library manager database such as status and storage group information. Use EDGUTIL MEND(SMSTAPE) to synchronize the TCDB and library manager database from DFSMSrmm. When you specify MEND, without SMSTAPE, EDGUTIL checks that DFSMSrmm is not active or that the DFSMSrmm control data set is not in use. MEND processing is performed the same way as VERIFY(ALL) and VERIFY(VOLCAT) processing. MEND processing cannot fix all discrepancies but those that can be fixed automatically are corrected by updating the control data set.

During VERIFY, EDGUTIL issues messages to indicate what stage of processing has been reached. These messages also go in the SYSPRINT file. An example of the SYSPRINT message file can be seen in Figure 133.

```
EDG6433I STARTING VERIFICATION OF RACK     RECORDS
EDG6433I STARTING VERIFICATION OF VOLUME   RECORDS
EDG6433I STARTING VERIFICATION OF DATA SET RECORDS
EDG6433I STARTING VERIFICATION OF OWNER    RECORDS
EDG6433I STARTING VERIFICATION OF PRODUCT  RECORDS
EDG6434I NO PRODUCT      RECORDS IN CONTROL DATA SET
EDG6433I STARTING VERIFICATION OF STORE    RECORDS
EDG6434I NO EMPTY BIN    RECORDS IN CONTROL DATA SET
EDG6434I NO INUSE BIN    RECORDS IN CONTROL DATA SET
EDG6433I STARTING VERIFICATION OF VRS      RECORDS
EDG6417I CONTROL DATA SET VERIFY SUCCESSFUL
EDG6901I UTILITY EDGUTIL COMPLETED WITH RETURN CODE 0
```

Figure 133. Sample EDGUTIL SYSPRINT Output

You can correct errors that are found in the control data set by using one of the following methods:

- Restore the control data set to a level where the errors are not present. See “Restoring the Control Data Set” on page 325 for additional information.
- Correct information in the control data set by using the RMM TSO ADD, CHANGE, DELETE, and LIST subcommands. See*z/OS DFSMSrmm Guide and Reference* for additional information.
- Mend the control data set by specifying MEND on the EXEC parameter of EDGUTIL. See“Mending the Control Data Set” on page 345 for additional information.
- Synchronize the TCDB and the library manager database from the DFSMSrmm control data set by specifying MEND(SMSTAPE) on the EDGUTIL EXEC parameter. See“Synchronizing the Contents of the Control Data Set” on page 345 for additional information.
- Synchronize the DFSMSrmm control data set from the TCDB by specifying MEND(VOLCAT) on the EDGUTIL EXEC parameter. See“Synchronizing the Contents of the Control Data Set” on page 345 for additional information.

While verify is running in parallel with DFSMSrmm active, the DFSMSrmm control data set can be updated by other processing, and as a result, verify can report inconsistencies for resources that are updated during the verify processing. To avoid rerunning verify to clean up these inconsistencies; run EDGUTIL when there is little activity that updates the control data set.

DFSMSrmm processes each resource in turn, dependent on the VERIFY parameter, and validates related information as follows:

- For every shelf location for a scratch volume, the associated volume must have the correct shelf location and media name and must be recorded as a scratch volume.
- For every data set, the corresponding volume must be defined and the next and previous data set names in sequence must also be correctly defined.
- For every defined owner, the corresponding volumes are checked for correct information.

Verifying the Control Data Set and Tape Configuration Database

You can use EDGUTIL VERIFY(VOLCAT) to check the consistency of information in the control data set and the tape configuration database (TCDB). EDGUTIL checks volume status and the library where the volume resides. You can use EDGUTIL VERIFY(SMSTAPE) to include the library manager database in the consistency checking. You can use MEND(SMSTAPE) to update the TCDB and library manager database with information from the DFSMSrmm control data set.

Figure 134 shows JCL for verifying the control data set and tape configuration database:

```
//UTIL     EXEC     PGM=EDGUTIL,PARM='VERIFY(VOLCAT)'
//SYSPRINT DD       utility message data set
//MASTER   DD       control data set
```

Figure 134. Sample JCL for Verifying the Control Data Set and the TCDB

Synchronizing the Contents of the Control Data Set

You can synchronize the contents of the DFSMSrmm control data set, TCDB, and the library manager database by using the EDGUTIL utility. First, use the verify function to check for inconsistencies in the contents of the DFSMSrmm control data set with the TCDB and the Library Then use EDGUTIL to automatically fix the inconsistencies found between the control data set and the TCDB and Library Manager.

You can use EDGUTIL VERIFY(SMSTAPE) to check the synchronization of the DFSMSrmm control data set with the TCDB and Library Manager database. Specify EDGUTIL MEND(SMSTAPE) to synchronize the TCDB and the library manager database from the DFSMSrmm control data set. You can run the function against an active control data set because the function does not update the control data set. The function uses the DFSMSrmm control data set as the master and makes changes to the TCDB and library manager database such as status and storage group information.

Specify EDGUTIL MEND(VOLCAT) with DFSMSrmm inactive to update the DFSMSrmm control data set from the TCDB with information such as location, media type, storage group, and intransit status.

Mending the Control Data Set

Recommendations:

- Always run the MEND function on a VSAM copy of the control data set first. Taking a back up of the control data set is essential because the control data set is unusable if MEND processing fails for any reason.
- Do not run the MEND function when DFSMSrmm is running on the same system and is using a control data set with the same name as the control data set that the MEND function is expected to correct.
- Do not enable stacked volume support until all systems using the control data set are on a supporting release level.

You can use EDGUTIL VERIFY as described in “Verifying the Contents of the Control Data Set” on page 343 to find most control data set errors. Obtain guidance from the IBM Support Center to use EDGUTIL MEND to fix the errors found in the DFSMSrmm control data set. When you use the EDGUTIL MEND function to fix errors for system-managed tape volumes, MEND updates the DFSMSrmm control data set from information from the TCDB and the library manager database.

You can also use the MEND function to enable stacked volume support as described in “Setting up DFSMSrmm Stacked Volume Support.” MEND processing checks to see if you have enabled stacked volume support with the EDGUTIL UPDATE option STACKEDVOLUME(YES). DFSMSrmm then creates stacked volume information from the existing volume ’In container’ values. DFSMSrmm marks the control data ready for stacked volume support if processing is successful.

During MEND processing, DFSMSrmm creates stacked volumes in the DFSMSrmm control data set that are marked as media type HPCT and recording format 128TRACK. DFSMSrmm obtains the media name from pool definitions defined using the DFSMSrmm parmlib member VLPOOL command. Start DFSMSrmm at least once with a parmlib member that contains the VLPOOL commands, before you run EDGUTIL, to ensure that EDGUTIL can use the VLPOOL information. DFSMSrmm marks the stacked volumes as ’Created during MEND’ and obtains location information from the last volume in sequence that is found to be in the container.

After MEND processing completes, you might need to add or change some volume information. Use the RMM CHANGEVOLUME subcommand to set or change any information.

Setting up DFSMSrmm Stacked Volume Support

To enable stacked volume support, perform the following tasks:

1. Update all systems sharing a control data set to the level of code that contains stacked volume support.
2. Correct any information about stacked volumes that you have defined to DFSMSrmm before running the DFSMSrmm EDGUTIL utility. EDGUTIL changes the volume type to stacked but does not set the correct location information for the volume. You can set the correct volume type and location information prior to running EDGUTIL MEND using the RMM CHANGEVOLUME subcommand. Figure 135 on page 347 shows an example of how you can use the RMM SEARCHVOLUME subcommand to build a CLIST that can be used to change volume information. Specify operands that are based on the way that you have defined volumes to DFSMSrmm. Then run the CLIST produced by the command to make changes to the volumes.

```
RMM SEARCHVOLUME VOLUME(ST*) OWNER(*) LIMIT(*) -
    CLIST('RMM CHANGEVOLUME ','TYPE(STACKED) LOCATION(vts_name) NORACK')
```

Figure 135. Changing Volume Type and Volume Location

3. Run the EDGUTIL utility with UPDATE with the STACKEDVOLUME(YES) operand on the CONTROL statement of the SYSIN file to enable stacked volume support.
4. To check the stacked volume information, you can use EDGUTIL with VERIFY(VOLUME) to check whether the container information is correct. Use the RMM LISTCONTROL CNTL subcommand to display the status of support. DFSMSrmm marks the support status as MIXED if there is any container information in the control data set volume records. If the support status shows MIXED, you can run EDGUTIL MEND as described in “Mending the Control Data Set” on page 345 to make the container information consistent. During MEND processing, DFSMSrmm creates the necessary stacked volumes if you have not previously defined them using the DFSMSrmm subcommands.
5. Run EDGHSKP storage location management processing to clean up location and bin number information in volumes that are in a container.

Enabling Extended Bin Support

To enable extended bin support, create or update the control data set control record using the EDGUTIL utility with the EXTENDEDBIN(YES) option. See “Creating or Updating the Control Data Set Control Record” on page 341 for a detailed description of the EXTENDEDBIN parameter.

When extended bin support is enabled, DFSMSrmm records additional volume information and bin information to keep track of the volume’s location when a volume is moving from a bin-managed storage location or to a bin-managed storage location.

DFSMSrmm keeps track of the following information for a volume: Destination bin number, Destination bin media name for a volume, current bin number, current bin media name, Old bin number, and Old bin media name When a volume starts moving to a bin-managed storage location, DFSMSrmm updates the destination bin fields with the bin number and the bin media name. When the move has been confirmed, DFSMSrmm updates the current bin fields with the destination bin fields.

If a volume moves from a bin-managed storage location, DFSMSrmm does not change the current bin fields until the move has been confirmed. DFSMSrmm changes the old bin number to the current bin number and clears the current bin number when a move is confirmed. When extended bin support is not enabled and a volume is moving, DFSMSrmm shows the source bin in the old bin fields and the target bin in the current bin fields.

DFSMSrmm keeps track of additional volume information in the bin record when extended bin support is enabled: Moving-in volume, Moving-out volume, and Old volume. When extended bin support is enabled, a volume, for which a move to the bin has been started, is shown as the ’moving-in volume’ in the bin record. When the volume move is confirmed, the volume is shown as the current volume in the bin record. A volume, for which a move from the bin has been started, is shown as the ’moving-out volume’ in the bin record. When the volume move out of the bin is confirmed, the volume is shown as the ’old volume’. When extended bin support is not enabled, DFSMSrmm shows a volume as the 'current volume' in the bin record from the time the move to the bin has been started until the time that the move from this bin has been confirmed.

Before you enable extended bin support, perform the following steps:

1. Complete all outstanding volume moves from and to bin-managed storage locations.
2. Run inventory management vital record processing or inventory management expiration processing to complete the confirmation of the volume moves.
3. Review user-written programs, REXX EXECs, and reports that contain information about bins. You might need to modify the programs and reports to incorporate information provided with extended bin support. In an RMMplex, your system-managed libraries must be connected to at least one system which either runs z/OS Version 1 Release 3 or higher or has APAR OW49863 installed.

Return Codes for EDGUTIL

EDGUTIL issues these return codes shown in Table 52.

Table 52. EDGUTIL Return Codes

Return Code	Explanation
0	All requested functions completed successfully.
4	DFSMSrmm encountered a minor error during processing. It issues a warning message and continues processing.
8	DFSMSrmm has encountered an error opening the control data set.
12	DFSMSrmm encountered a severe error during processing of one of the requested functions. DFSMSrmm stops the utility.
16	DFSMSrmm encountered a severe error during a required communication with the DFSMSrmm subsystem. DFSMSrmm stops the utility.

Sharing the DFSMSrmm Control Data Set

When you share a DFSMSrmm control data set where at least one system does not support system-managed tape libraries, you must consider the following conditions:

- Data Facility Removable Media Manager for MVS/DFP Version 3 (DFRMM) Program Offering provides support for non-system-managed tape libraries. When you use DFRMM and DFSMSrmm together, or multiple DFSMSrmm systems, they can share the same control data set. When both DFRMM and DFSMSrmm share the control data set, you can use the DFRMM ISPF dialog and RMM TSO subcommands to display all information that has been recorded in the control data set. There are some restrictions on using the RMM TSO subcommands from DFRMM, and from DFSMSrmm on a non-system-managed tape system, to add and change information in the control data set.
- DFRMM has no knowledge of volume residency in an IBM TotalStorage Enterprise Automated Tape Library (3494)r or an IBM TotalStorage Enterprise Automated Tape Library (3495), other than that which is provided by DFSMSrmm on a DFSMS system using a shared control data set.

Running DFSMSrmm Inventory Management When Sharing the Control Data Set

Use the EDGHSKP utility to run inventory management activities that include: vital record processing, expiration processing, storage location management processing, backing up the control data set and journal, and creating an extract data set. See Chapter 14, "Performing Inventory Management," on page 275 for more information.

When you are using DFRMM or have at least one non-system-managed tape environment, you should perform inventory management using DFSMSrmm in a system-managed tape environment. DFSMSrmm ensures that the TCDB is updated with the correct volume status as volumes are returned to scratch, and that volume movement is automatically confirmed to IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495).

Running EDGINERS When Sharing the Control Data Set

If you are using DFRMM or have at least one non-system-managed tape environment, run EDGINERS with DFSMSrmm in a system-managed tape environment so that you can take advantage of the dynamic allocation capability of EDGINERS.

Defining Volume Information When Sharing the Control Data Set

When you use DFRMM or have at least one non-system-managed tape environment, issue RMM TSO ADDVOLUME, DELETEVOLUME, and CHANGEVOLUME EJECT subcommands using DFSMSrmm in the system-managed tape environment so the TCDB is automatically updated.

Confirming Volume Movement When Sharing the Control Data Set

Run inventory management on the DFSMSrmm that is running in the system-managed tape environment when DFRMM and DFSMSrmm are sharing a control data set. Use the RMM CHANGEVOLUME subcommand with the following operands to confirm that pending volume movement has taken place:

```
RMM CHANGEVOLUME * CMOVE(from_location,to_location)
```

You can specify a system-managed library name as the *from_location* or *to_location* name when using either DFRMM or DFSMSrmm even if system-managed tape is not in use.

When you run inventory management on the DFSMSrmm system, the volume information in the control data set is updated to confirm that any volumes pending movement have been moved. If you run inventory management on the DFRMM system, your request does not fail but volume information is not updated.

Returning Volumes to Scratch When Sharing the Control Data Set

When volumes that reside in an IBM TotalStorage Enterprise Automated Tape Library (3494)r or IBM TotalStorage Enterprise Automated Tape Library (3495) are returned to scratch by DFSMSrmm, information needs to be updated in the TCDB.

When you have some systems without system-managed tape active, ensure that inventory management runs on a system with system-managed tape active so that system-managed volumes returning to scratch are updated in the TCDB automatically during inventory management expiration processing.

Chapter 16. Initializing and Erasing Tape Volumes

DFSMSrmm Samples Provided in SAMPLIB

- EDGJINER Sample JCL for Using the EDGINERS Utility for Initializing and Erasing Tapes
- EDGLABEL Sample Started Procedure for Initializing and Erasing Tapes

Use EDGINERS to initialize and erase volumes. Run EDGINERS regularly as part of your inventory management processing.

EDGINERS writes BCD labels on 7-track tape volumes and ASCII (ISO/ANSI format) labels on tape cartridges or 9-track tape volumes. EDGINERS writes 7-track tape labels in even parity (translator on, converter off). You can label tape cartridges, 7-track tape volumes, or 9-track tape volumes with EDGINERS.

EDGINERS provides support for ISO/ANSI version 3 VOL1 and HDR1 labels and for ISO/ANSI version 4 VOL1 and HDR1 labels.

If you are authorized to change the label type, you can relabel tape volumes by specifying the label type in your JCL. This is done at the time of volume use to avoid a separate mount of the volume. When you change the volume label type at the time of use, you do not need to use either EDGINERS or the INIT action. DFSMSrmm allows the tape volume label to be created or overwritten at any time, regardless of the status of the volume, if the user has the required access to a security resource. See Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171 for more information about volume labels.

See *z/OS DFSMSrmm Guide and Reference* for operator procedures that describe operator tasks like responding to initialization messages, tape mount messages, and using the LABEL procedure to request EDGINERS processing. See “Using the LABEL Procedure” on page 382 for a description of the EDGLABEL procedure provided by DFSMSrmm.

Before initializing or erasing a volume, DFSMSrmm ensures that the correct volume is mounted by reading the volume label. For DFSMSrmm-defined volumes, it also ensures that the requested action is actually required.

When DFSMSrmm reads an existing volume label, the request might fail because the existing volume requires formatting of the servo tracks. If DFSMSrmm detects that a mounted volume has a servo track formatting error, DFSMSrmm issues message EDG6658I and fails the request to initialize or erase the volume.

Note: The recording technology and media associated with IBM new tape architecture products, for example, 3590, uses servo track. See *IBM 3590 High Performance Tape Subsystem Introduction and Planning Guide, GA32-0330* for details.

DFSMSrmm erases volumes using the hardware security erase feature when it is available. When erasing volumes, DFSMSrmm also reinitializes them so that the correct volume labels are written and the volumes are ready for reuse. If the hardware security erase feature is not available, DFSMSrmm overwrites volumes with a bit pattern of hex FF.

Replacing IEHINITT with EDGINERS

You can use the DFSMSrmm EDGINERS utility or the IEHINITT utility to initialize and erase tape volumes.

Use the IEHINITT utility to initialize tapes you do not want to be defined to DFSMSrmm. Use the RMM CHANGEVOLUME subcommand to inform DFSMSrmm that the volume has been initialized. If you use tapes that are initialized using a utility other than EDGINERS and do not inform DFSMSrmm, DFSMSrmm can issue message EDG4026I at OPEN time.

EDGINERS, if you decide to use it, performs the following tasks:

- Reads and validates volume labels.
- Maintains RACF profiles.
- Uses and updates information in the DFSMSrmm control data set.
- Provides a facility to erase tapes.
- Defines volumes you initialize or erase that are not yet defined in the control data set.
- Ends processing when two consecutive errors are detected on the same volume to prevent subsequent volumes from being used incorrectly when a cartridge loader is in use. For example, if an incorrect volume label is read, the volume is demounted and a new mount request issued. Processing is dependent on the operator response to DFSMSrmm message EDG6663D and the WRONGLABEL processing described in “EXEC Parameters for EDGINERS” on page 354.
- Issues a WTOR to the operator when a mount request is issued. A reply to the WTOR is not always required as processing continues as soon as the volume is mounted. The WTOR is issued to allow the operator to skip a volume if the volume cannot be mounted for some reason.
- Bypasses any IOS000I messages for an ’NCA’ error (tape not capable message) when reading the label on a volume that has not been initialized.

For information about preventing or limiting the use of IEHINITT, see Chapter 9, “Authorizing DFSMSrmm Users and Ensuring Security,” on page 171.

Using EDGINERS

You can use EDGINERS in either automatic processing mode and manual processing mode. Initialize and erase actions that are defined in the control data set drive automatic processing. With automatic processing, volumes are initialized and erased without operator intervention. SYSIN within JCL or operator commands drive manual processing. You might set up a job that initializes new volumes using automatic processing to minimize librarian intervention. Use manual processing for initializing volumes when you want to change a known, existing volume serial number to another volume serial number.

Initializing and Erasing Volumes Automatically

To initialize and erase volumes without operator intervention, you can set up automatic processing by specifying any of these EDGINERS EXEC parameters: COUNT, INITIALIZE, ERASE, LOCATION, MEDIANAME, MEDIATYPE, POOL, or RECORDINGFORMAT. When you request automatic processing, DFSMSrmm does not process any SYSIN commands you specify.

During DFSMSrmm expiration processing, DFSMSrmm records the volumes that need to be initialized or erased when they are released or returned to scratch. If you use the EXEC parameters in your JCL to set up automatic processing, DFSMSrmm initializes or erases these volumes without operator intervention. EDGINERS issues write-to-operator messages and MSGDISP requests to the operator and the drive to get a volume mounted and demounted. If operators are unable to mount a volume, DFSMSrmm allows them to skip processing the current requested volume.

To initialize scratch volumes in a non-system-managed library that you are adding to DFSMSrmm, use the RMM ADDVOLUME subcommand with the INIT(Y) operand to mark the volumes you want initialized before they are available as scratch. Specify the INITIALIZE EXEC parameter in your EDGINERS JCL. DFSMSrmm initializes all the volumes marked as requiring initialization. See “Initializing Scratch Volumes in System-Managed Libraries” on page 92 for information on initialization for volumes in a system-managed library.

Initializing and Erasing Volumes Manually

Recommendation: Use manual processing with automatic cartridge loaders and volumes with old labels. DFSMSrmm performs manual processing that requires operator intervention when you do not specify any EXEC parameters that select automatic processing. These parameters are described in “Initializing and Erasing Volumes Automatically” on page 352. Specify commands in the SYSIN file or reply to operator messages to when you want operator intervention.

The only required SYSIN command operand is the volume serial number. When you specify the volume serial number, DFSMSrmm gets the rest of the information about the volume from the control data set.

If you supply MEDIANAME, POOL, or RACK SYSIN command operands that do not match the existing volume entry, DFSMSrmm issues an error message and stops processing the current volume.

Initializing and Erasing Volumes Using Multiple Tape Drives

You can use multiple tape drives when initializing or erasing volumes by running one copy of EDGINERS for each tape drive that you wish to use. To run multiple copies of EDGINERS, submit multiple jobs or start multiple procedures. You can specify the same parameters or different parameters for each EDGINERS job you run. For example, you can split volumes between jobs by specifying the MEDIANAME operand, POOL operand, or the LOCATION operand on separate runs of EDGINERS to initialize different volumes. If you use the same parameters, each EDGINERS job you run shares the volumes that are to be initialized. During EDGINERS processing, DFSMSrmm serializes the use of the volume by using a SYSTEMS ENQ. Serializing the use of the volume ensures that no other invocation of EDGINERS attempts to initialize the volume. EDGINERS skips those volumes already processed or that are serialized. When you run multiple copies of EDGINERS with the same parameters, specify the BATCH(0) parameter to ensure that EDGINERS processing continues until all volumes are initialized or erased. Refer to“EXEC Parameters for EDGINERS” on page 354 for information about the EDGINERS BATCH parameter and the EDGINERS COUNT parameter. Specify the COUNT(*x*) parameter if you are using the VERIFY parameter or if you are using cartridge loaders to control the batch size.

JCL for EDGINERS

Figure 136 shows sample JCL for automatic processing.

```
//INIT     EXEC PGM=EDGINERS,
//              PARM='MEDIANAME(3480),VERIFY'
//SYSPRINT DD   program message data set
//TAPE     DD   UNIT=(TAPE,,DEFER)
```

Figure 136. JCL for EDGINERS Automatic Processing

Figure 137 shows sample JCL for manual processing.

```
//INIT     EXEC PGM=EDGINERS
//SYSPRINT DD   program message data set
//TAPE     DD   UNIT=(TAPE,,DEFER)
//SYSIN    DD   optional control command input
```

Figure 137. JCL for EDGINERS Manual Processing

The TAPE DD statement is only required if any of the volumes to be processed are not in a system-managed tape library.

Figure 138 shows sample JCL for initializing volumes with ISO/ANSI version 4 tape labels. The EXEC JCL statement can be overridden by control information from SYSIN, operator replies to messages, or information from the DFSMSrmm control data set.

```
//INIT     EXEC PGM=EDGINERS,PARM='..,ALVER4,...'
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   DUMMY
```

Figure 138. JCL for Initializing Volumes with ISO/ANSI Version 4 VOL1 and HDR1 Labels

EXEC Parameters for EDGINERS

Figure 139 on page 355 describes the EXEC parameters.

```
►►──┬─────────────────────────────────┬──┬─────────────────────────────┬──────────────►
    │  ┌─parmlib_DEVSUPxx_member_value─┐ │  └─BATCH(number_of_batches)─┘
    └──┼─ALVER3────────────────────────┤─┘
       └─ALVER4────────────────────────┘

►──┬────────────────────┬──┬──────────────────┬──┬──────────────┬─────────────────────►
   │              (1)   │  │              (1) │  │          (1) │
   └─COUNT(count)───────┘  └─INITIALIZE───────┘  └─ERASE────────┘

►──┬──────────────────────────────────────────────────┬───────────────────────────────►
   │            ┌─parmlib_default_medianame─┐  (1)     │
   ├─MEDIANAME(─┴─medianame─────────────────┴─)───────┤
   │              (1)                                 │
   ├─POOL(pool_prefix)────────────────────────────────┤
   ├─LOCATION(─┬─library_name─┬─)─────────────────────┤
   │           └─SHELF────────┘                       │
   │                                  (1)             │
   ├─MEDIATYPE(─┬─*──────────┬─)──────────────────────┤
   │            ├─CST────────┤                        │
   │            ├─ECCST──────┤                        │
   │            ├─EHPCT──────┤                        │
   │            ├─HPCT───────┤                        │
   │            ├─┬─MEDIA5─┬─┤                        │
   │            │ └─ETC────┘ │                        │
   │            ├─┬─MEDIA6─┬─┤                        │
   │            │ └─EWTC───┘ │                        │
   │            ├─┬─MEDIA7─┬─┤                        │
   │            │ └─EETC───┘ │                        │
   │            └─┬─MEDIA8─┬─┘                        │
   │              └─EEWTC──┘                          │
   │                                    (1)           │
   └─RECORDINGFORMAT(─┬─*────────┬─)──────────────────┘
                      ├─18TRACK──┤
                      ├─36TRACK──┤
                      ├─128TRACK─┤
                      ├─256TRACK─┤
                      ├─384TRACK─┤
                      └─EFMT1────┘

►──┬──────────────────────────┬──┬────────────┬──┬───────────────────────────────┬────►◄
   │         ┌─NOTMASTER─┐    │  ├─VERIFY─────┤  │              ┌─FAIL──────┐    │
   └─STATUS(─┼─ALL───────┼─)──┘  └─NOVERIFY───┘  └─WRONGLABEL(─┼─IGNORE────┼─)──┘
             └─SCRATCH───┘                                      ├─PROMPT────┤
                                                                └─RMMPROMPT─┘
```

Notes:

1 Specify this parameter for automatic processing.

Figure 139. EDGINERS EXEC Parameters

ALVER3
Use ALVER3 to set the EDGINERS processing default value for ISO/ANSI label tapes to version 3. To understand how the DFSMSrmm assigns the label version, see “How DFSMSrmm Selects an ISO/ANSI Label Version” on page 366.

ALVER4
Use ALVER4 to set the EDGINERS processing default value for ISO/ANSI label

tapes to version 4. To understand how the DFSMSrmm assigns the label version, see "How DFSMSrmm Selects an ISO/ANSI Label Version" on page 366.

BATCH(*number_of_batches***)**
Use BATCH to specify the number of batches of volumes to be processed in a single run of EDGINERS automatic processing. Use the COUNT parameter to specify the number of volumes in each batch. The COUNT is the number of volumes that are initialized or erased before DFSMSrmm verifies the volumes. After DFSMSrmm verifies the volumes in a batch, EDGINERS starts again to initialize or erase the volumes in the next batch.

If you specified the NOVERIFY parameter, the number of volumes that are processed is the BATCH value or its default, multiplied by the value of COUNT or its default. However, DFSMSrmm does not batch the processing of these volumes.

The default for BATCH is BATCH(1). To process all volumes that have actions pending, specify BATCH(0). DFSMSrmm treats BATCH(0) as BATCH(X'FFFFFFFF'), which is the upper limit for the number of batches that DFSMSrmm can process.

COUNT(*count***)**
Use COUNT to specify the number of volumes to initialize or to erase when DFSMSrmm performs automatic processing. Use COUNT with the BATCH parameter to specify the number of volumes in each batch of volumes to be processed. The maximum value that you can specify is 99. If automatic processing is in effect but COUNT is omitted, then the default value is 10. When you specify COUNT, DFSMSrmm performs automatic processing.

ERASE
Use ERASE to request that DFSMSrmm selects volumes that have the erase action pending. If automatic processing is in effect but ERASE is not specified then DFSMSrmm will only select volumes with the initialize action pending. When you specify ERASE, DFSMSrmm performs automatic processing.

INITIALIZE
Use INITIALIZE to request that DFSMSrmm selects volumes that have the initialize action pending. If automatic processing is in effect but neither INITIALIZE nor ERASE are specified, then INITIALIZE is the default. You can also specify INITIALISE for INITIALIZE. When you specify INITIALIZE, DFSMSrmm performs automatic processing.

LOCATION(*library_name***)**
Use LOCATION to specify a subset of volumes for automatic processing. The *library_name* must be the name of a system-managed tape library that is on the running system or SHELF. If you specify LOCATION, you cannot specify MEDIANAME, MEDIATYPE, POOL, or RECORDINGFORMAT.

There is no default *library_name* value. If you do not specify LOCATION, MEDIANAME, MEDIATYPE, POOL, or RECORDINGFORMAT, DFSMSrmm uses MEDIANAME as the default parameter for automatic processing. This means that all volumes defined with the default medianame are selected if they have the required action pending.

MEDIANAME(*medianame* | *parmlib_default_medianame***)**
Use MEDIANAME to specify a subset of volumes for automatic processing. If you specify MEDIANAME, you cannot specify LOCATION, MEDIATYPE, POOL, or RECORDINGFORMAT. If you do not specify LOCATION, MEDIANAME, MEDIATYPE, POOL, or RECORDINGFORMAT, DFSMSrmm uses

MEDIANAME as the default parameter for automatic processing. This means that all volumes defined with the default medianame are selected if they have the required action pending.

DFSMSrmm does not use MEDIANAME to set a default for the SYSIN INIT and ERASE commands MEDIANAME operand.

The default MEDIANAME is the value that you define with the EDGRMM*xx* parmlib OPTION MEDIANAME operand described in “Defining System Options: OPTION” on page 134.

| **MEDIATYPE(* | CST | ECCST | EHPCT | HPCT | MEDIA5 | MEDIA6 | MEDIA7 |**
| **MEDIA8)**

Use MEDIATYPE to specify a subset of volumes for automatic processing. Specifies the volume’s physical media type. Use one of the following:

*****	The volume is not a cartridge.
CST	Cartridge System Tape
ECCST	Enhanced Capacity Cartridge System Tape
EHPCT	Extended High Performance Cartridge Tape
HPCT	High Performance Cartridge Tape
MEDIA5/ETC	IBM TotalStorage Enterprise Tape Cartridge
MEDIA6/EWTC	IBM TotalStorage Enterprise WORM Tape Cartridge 3592
MEDIA7/EETC	IBM TotalStorage Enterprise Economy Tape Cartridge 3592
MEDIA8/EEWTC	IBM TotalStorage Enterprise Economy WORM Tape Cartridge 3592

When you specify MEDIATYPE, DFSMSrmm performs automatic processing. If you specify MEDIATYPE, you cannot specify LOCATION, MEDIANAME, POOL, or RECORDINGFORMAT.

There is no default MEDIATYPE value. If you do not specify LOCATION, MEDIANAME, MEDIATYPE, POOL, or RECORDINGFORMAT, DFSMSrmm uses MEDIANAME as the default parameter for automatic processing. This means that all volumes defined with the default medianame are selected if they have the required action pending.

POOL(*pool_prefix***)**

Use POOL to specify a subset of volumes for automatic processing. A pool prefix is one-to-five alphanumeric, national, or special characters followed by an asterisk (*). The pool must be one that is defined to DFSMSrmm on the running system. If you specify POOL, you cannot specify LOCATION, MEDIANAME, MEDIATYPE, or RECORDINGFORMAT.

There is no default *pool_prefix* value. If you do not specify LOCATION, MEDIANAME, MEDIATYPE, POOL, or RECORDINGFORMAT, DFSMSrmm uses MEDIANAME as the default parameter for automatic processing. This means that all volumes defined with the default medianame are selected if they have the required action pending.

RECORDINGFORMAT(*| 18TRACK | 36TRACK | 128TRACK | 256TRACK | 384TRACK | EFMT1)

Use RECORDINGFORMAT to specify a subset of volumes for automatic processing. RECORDINGFORMAT specifies the basic recording format for tape volumes.

*****
An asterisk indicates that the format is unknown or that the volume is not a tape volume.

18TRACK
Data has been written to the volume in 18-track format. A recording format of 18TRACK is valid with MEDIATYPE(CST) and MEDIATYPE(ECCST) only.

36TRACK
Data has been written to the volume in 36-track format. A recording format of 36TRACK is valid with MEDIATYPE(CST) and MEDIATYPE(ECCST) only.

128TRACK
Data has been written to the volume in 128-track format. A recording format of 128TRACK is valid with MEDIATYPE(EHPCT) and MEDIATYPE(HPCT) only.

256TRACK
Data has been written to the volume in 256-track format. A recording format of 256TRACK is valid with MEDIATYPE(EHPCT) and MEDIATYPE(HPCT) only.

384TRACK
Data has been written to the volume in 384-track format. A recording format of 384TRACK is valid with MEDIATYPE(EHPCT) and MEDIATYPE(HPCT) only.

EFMT1
Data has been written to the volume in EFMT1 (enterprise format 1) recording format. A recording format of EFMT1 is valid with
| MEDIATYPE(MEDIA5), MEDIATYPE(MEDIA6), MEDIATYPE(MEDIA7),
| and MEDIATYPE(MEDIA8) only.

There is no default RECORDINGFORMAT. If you do not specify LOCATION, MEDIANAME, MEDIATYPE, POOL, or RECORDINGFORMAT, DFSMSrmm uses MEDIANAME as the default parameter for automatic processing. This means that all volumes defined with the default medianame are selected if they have the required action pending.

STATUS

Use STATUS to control the kind of tapes that you want DFSMSrmm to initialize or erase. The default for STATUS is NOTMASTER. Specifying STATUS requests automatic processing.

ALL
EDGINERS processes all volumes that have the INITIALIZE or ERASE action pending.

NOTMASTER
EDGINERS processes all volumes in SCRATCH, USER, INIT, ENTRY or PENDING RELEASE status that have the INITIALIZE or ERASE action pending. EDGINERS does not process any volumes in MASTER status. NOTMASTER is the default.

SCRATCH
EDGINERS processes volumes in SCRATCH, INIT, ENTRY or PENDING RELEASE status that have the INITIALIZE or ERASE action pending. EDGINERS does not process any volumes in MASTER or USER status.

VERIFY|NOVERIFY
Use VERIFY to request that DFSMSrmm ask the operator to remount each volume that has been successfully erased or labeled. The volumes are requested in reverse order, and the volume labels read to ensure no operator errors have occurred, for example, a mismatch between the internal label and the external label.

For automatic processing VERIFY is the default. For manual processing NOVERIFY is the default.

WRONGLABEL
Use WRONGLABEL to specify the processing that DFSMSrmm performs when a wrong volume is mounted. WRONGLABEL processing does not apply to NL tapes. For NL tapes, DFSMSrmm issues the WTOR EDG6628A to obtain the volume serial number or rack number for the volume that has been mounted. You can use WRONGLABEL when you are running EDGINERS in automatic mode and manual mode.

FAIL
DFSMSrmm does not prompt the operator to accept a mounted volume that does not match the requested volume. The mount request is rejected, the volume demounted, and DFSMSrmm issues message EDG6661E or message EDG6662E.

IGNORE
When the wrong volume is mounted, DFSMSrmm does not issue any operator prompt. DFSMSrmm issues message EDG6661E or EDG6662E to log the relabeling and processing proceeds. This is an extremely dangerous option and should be used with caution because any volume can be relabeled as long as the requested volume has the INIT action or is not defined to DFSMSrmm. Use of this option requires CONTROL access to RACF FACILITY class resource STGADMIN.EDG.INERS.WRONGLABEL.

PROMPT
When an incorrect volume label is detected by EDGINERS for the mounted volume, the operator is always prompted to confirm the processing to be performed. DFSMSrmm issues message EDG6661E or message EDG662E, followed by message EDG6663D. Processing continues according to the response to message EDG6663D. This option should be used with caution because any volume can be relabeled as long as the requested volume is either known to DFSMSrmm and has the INIT action, or is not known to DFSMSrmm. No additional authorization is required, other than the authorization required for running EDGINERS.

RMMPROMPT
When the volume serial number of the mounted volume that is defined to DFSMSrmm does not match the volume serial number of the requested volume, DFSMSrmm issues message EDG6663D to prompt the operator to confirm processing. If the magnetic volume serial number of the tape is not known to DFSMSrmm, initialization continues as if the tape had no magnetic label. If the volume is known to DFSMSrmm, DFSMSrmm issues messages EDG6662E and EDG6663D to prompt the operator or issues message EDG6661E to log the relabeling. Use this option when your installation has defined all its volumes to DFSMSrmm; otherwise caution is

required. Use of this option requires UPDATE access to RACF FACILITY class resource STGADMIN.EDG.INERS.WRONGLABEL.

SYSIN Commands for EDGINERS

Use the SYSIN commands for manual processing. You must use separate SYSIN commands for each volume you want initialized or erased. The format of the SYSIN commands and operator replies are shown in Figure 140.

```
►►──┬─INIT──┬──VOLUME(volser─┬─────────────┬─)──────────────────────────►
    └─ERASE─┘                └─,new_volser─┘

►──┬──────────────────────────────────────────────────┬──────────────────►
   │          ┌─SL──────────────────────────────┐     │
   └─LABEL(──┼─AL──┤ ANSI Label Parameters ├──┼─)──┘
              └─NL──────────────────────────────┘

►──┬───────────────────────────────────────────┬──┬─────────────────────┬─►
   │            ┌─parmlib_default_medianame─┐  │  ├─RACK(rack_number)───┤
   └─MEDIANAME(─┴─medianame─────────────────┴─)┘  └─POOL(pool_prefix)───┘

►──┬─────────────────┬──┬──────────────┬─────────────────────────────────►◄
   └─OWNERTEXT(text)─┘  └─VOL1(volser)─┘
```

ANSI Label parameters:

```
├──┬──────────────┬──┬─────────────────────────────────┬──────────────────┤
   └─ACCESS(code)─┘  └─┬─LABELVERSION─┬─(─┬─3─┬─)────┘
                       └─VERSION──────┘   └─4─┘
```

Figure 140. EDGINERS SYSIN Commands

ACCESS(*code*)
: Specifies the ISO/ANSI volume accessibility code. Specify *code* as any character in the ISO/ANSI X3.4–1986 character set. You must specify LABEL(AL) if you specify an accessibility code.

  You must modify the volume access installation exit routine in MVS/ESA to allow subsequent use of the volume if you specify ACCESS.

  The default is blank, allowing unlimited access to the volume.

ERASE
: Specify ERASE to security erase a volume and write a new label on it.

  You must specify either ERASE or INIT.

INIT
: Specify INIT to initialize a volume.

  You must specify either INIT or ERASE.

LABEL(NL|SL|AL)
: Use LABEL to specify the type of label to be written on the volume:

AL	Specifies an ISO/ANSI Label.
NL	Specifies no label.
SL	Specifies an IBM standard label.

If you do not specify the label type and the volume is already defined in DFSMSrmm, DFSMSrmm uses the label type defined in the DFSMSrmm control data set.

If you do not specify the label type and the volume is not already defined in the control data set, DFSMSrmm uses IBM standard label (SL) as the default.

LABELVERSION(3|4)
Use LABELVERSION to specify the ISO/ANSI volume label version. Specify LABELVERSION to update the DFSMSrmm control data set with the required label version for ISO/ANSI output tapes. To understand how DFSMSrmm assigns a label version if you do not specify the LABELVERSION, see "How DFSMSrmm Selects an ISO/ANSI Label Version" on page 366.

3 Use 3 to initialize tape volumes with ISO/ANSI version 3 VOL1 and HDR1 labels

4 Use 4 to initialize tape volumes with ISO/ANSI version 4 VOL1 and HDR1 labels.

MEDIANAME(*medianame*)
Specifies the volume's media name.

The media name that you specify must match the media name defined in the control data set. If the media names do not match, DFSMSrmm fails the request.

If the volume is not defined in the DFSMSrmm control data set, DFSMSrmm uses the value that you specify when you add the volume. If you do not specify a media name, DFSMSrmm uses the value that you defined for your installation with the EDGRMM*xx* parmlib OPTION MEDIANAME operand.

The default is the EDGRMM*xx* parmlib OPTION MEDIANAME operand value.

OWNERTEXT(*text*)
Specifies the owner's name or similar identification. *text* is fourteen characters. Enclose in single quotes if it includes blanks or special characters. The text must be 10 bytes for SL, 14 bytes for AL.

The information is specified as character constants up to 10 bytes long for EBCDIC and BCDIC volume labels and up to 14 bytes long for volume labels written in ISCII/ASCII.

POOL(*pool_prefix*)
Specifies a pool prefix for a pool to which you want to assign the volume. If the volume is not defined to DFSMSrmm, DFSMSrmm selects an available rack number for the volume in the pool you specify. If the volume is already defined in the DFSMSrmm control data set, DFSMSrmm changes the volume's rack number to move the volume.

If you do not supply a pool prefix or a rack number for a volume already defined in the DFSMSrmm control data set, DFSMSrmm uses the volume's existing rack number. If the volume is not defined in the control data set and you do not supply a pool prefix or a rack number, DFSMSrmm assigns the volume a rack number matching its volume serial number.

RACK(*rack_number*)
Specifies a shelf location for the volume. If you do not supply a pool ID or a rack number for a volume already defined in the control data set, DFSMSrmm uses the volume's existing rack number. If the volume is not defined in the

DFSMSrmm control data set and you do not supply a pool ID or a rack number, DFSMSrmm assigns the volume a rack number matching its volume serial number.

VOLUME(*volser***,***new_volser***)**

volser specifies the volume serial number of the volume to be initialized or erased. *volser* is required. If you are adding volumes with a volume serial number less than six characters, you must supply a rack number or a pool, otherwise DFSMSrmm issues an error message.

If the volume is already defined in the DFSMSrmm control data set, DFSMSrmm ensures that the requested action is pending for the volume. If this action is not pending, DFSMSrmm fails the request.

If the volume mounted is already labeled, DFSMSrmm reads the label to ensures that the volume serial number matches the one you specify. If the volume mounted does not have a recognizable volume label but contains data (no label tapes or nonstandard label tapes), DFSMSrmm issues a WTOR. The operator must reply to this message before DFSMSrmm can initialize or erase the volume.

If the volume is not defined in the DFSMSrmm control data set and you do not specify a new volume serial number, DFSMSrmm adds the volume to the control data set.

The value for the variable *new_volser* specifies a new volume serial number. Use it if you want to relabel a volume with a new volume serial number. If this new volume is already defined in the DFSMSrmm control data set, DFSMSrmm fails the request.

DFSMSrmm adds information about the new volume to the DFSMSrmm control data set, using information recorded for the volume you are replacing, and then deletes information about the volume you are replacing.

VOL1(*volser***)**

Use the VOL1 operand to specify the VOL1 label volser that is to be written in the tape label when it is required to be different than the external volser. The value for the variable *volser* is 1-to-6 alphanumeric, national, or special characters.

Volumes with a VOL1 value are treated as duplicate volumes by DFSMSrmm. If you are labeling a duplicate volume, DFSMSrmm uses the previously defined VOL1 value for the volume when you do not specify a VOL1 value for the volume. If you specify a VOL1 value, DFSMSrmm uses that VOL1 in place of the VOL1 value that was previously defined to DFSMSrmm. You can only label a volume as a duplicate when you are initializing or erasing volumes manually

For more information about tape label validation or volume access code, see *z/OS DFSMS: Using Magnetic Tapes*.

Using EDGINERS with System-Managed Tape Libraries

With DFSMS tape support, you can associate tape drives with specific system-managed tape libraries. As a result, you can mount volumes resident in a system-managed tape library only on the drives associated with that library.

Checking Volumes in System-Managed Tape Libraries

EDGINERS performs the following checking to make sure a volume can be used in a system-managed tape library:

- EDGINERS determines if a volume in a system-managed tape library can be mounted on the current system. If the volume cannot be mounted, possibly because it is defined in a TCDB on another system, DFSMSrmm skips that volume.
- In order to initialize a volume in a system-managed tape library, a volume must be in a private category because the automated tape library does not support specific mounts of scratch volumes. RMM TSO command and release processing attempts to ensure that the volume is in the correct category so that EDGINERS does not have to check the volume status. Refer to “Setting Status for a Volume in a System-Managed Tape Library” on page 364 for more information about setting the status for a volume.
- You must define a volume in a system-managed tape library to DFSMSrmm before you can initialize or erase it. Any volume not defined to DFSMSrmm will be mounted on the drive allocated by the TAPE DD statement in the JCL for EDGINERS as long as the drive is not in a system-managed library.

Requesting Volume Mounts for System-Managed Tape Libraries

EDGINERS requests volume mounts only on those drives on which the volumes can be mounted, by testing volume eligibility and using dynamic allocation to obtain a suitable drive for each volume to be processed.

DFSMSrmm uses the pre-allocated drive if it is available. If the pre-allocated drive is not available, EDGINERS dynamically allocates a drive using the first volume obtained from the control data set. Further processing checks that the subsequent volumes are eligible for use on the same drive. If the pre-allocated drive is not in a system-managed tape library, DFSMSrmm uses it for all volumes selected that are not resident in a system-managed tape library.

When dynamic allocation of a tape drive fails, EDGINERS sets return code to 4 and skips processing of the current volume. Messages describing the failure are issued. These might include messages prefixed with IKJ and CBR. In most cases you will see a return and reason code returned from OAM. For example, OAM issues return code 8 and reason code 51 to indicate that the requested volume is in scratch status. This error might be the result of a mismatch between the DFSMSrmm control data set and TCDB volume status. You can correct this by updating the status in the TCDB to match that known by DFSMSrmm. Refer to *z/OS DFSMSdfp Diagnosis Reference* for information about OAM return and reason codes.

Running EDGINERS on Multiple System Complexes

You can run EDGINERS on multiple system complexes to ensure that all required volumes are erased or initialized. To select a subset of volumes to initialize or erase, you can specify POOL and LOCATION with the EDGINERS EXEC parameter.

Running EDGINERS on a 3494 in Manual Mode

DFSMSrmm can determine the status of the vision system and if the library is in manual mode. EDGINERS can automatically handle some errors when an automated tape library is fully functional. For example, when the library is fully functional, DFSMSrmm uses the vision system volume serial number to verify that the correct volume is mounted.

Mounting and Demounting Volumes

EDGINERS provides a way to direct to an automated tape library the WTOR messages and MSGDISP requests to the operator and drive to get a volume mounted and demounted. These messages are issued through the library automation communication services component of OAM. If a mount on an automated tape library drive fails, DFSMSrmm uses the return and reason codes set by the library automation communication services component to test if a volume should be skipped.

During demount processing, DFSMSrmm ensures that errors detected on volumes mounted in an automated tape library are reflected in the TCDB. For example, DFSMSrmm ensures the TCDB contains information about write-protected, wrong volume, and wrong label type errors. DFSMSrmm skips the volume rather than having the operator correct the error.

Using DFSMSdfp Processing to Label Volumes

For volumes in an automated tape library, you have the option to use DFSMSdfp OPEN processing as an alternative to using EDGINERS to label scratch volumes.

If the automated tape library is fully functional (vision system working) and the VOL1 label for a scratch volume does not match the external label, DFSMSdfp rewrites the VOL1 label with the correct volume serial number.

DFSMSrmm defers the initialization of the volumes to DFSMSdfp if you request the initialization prior to entering a scratch volume into the automated tape library. DFSMSrmm turns off the initialize action.

For example you use the RMM ADDVOLUME subcommand to define scratch volumes to DFSMSrmm as shown in Figure 141. Specify INIT(Y) to request that the volume is initialized prior to first use. If you later enter the volumes into an automated tape library, during entry processing DFSMSrmm turns off the initialize action to defer the labeling to OPEN processing under DFSMSdfp control.

```
RMM ADDVOLUME A12345 INIT(Y) STATUS(SCRATCH)
```

Figure 141. Defining Scratch Volumes to be Initialized

If you request that the initialize action is set for a scratch volume that is already resident in the automated tape library, as shown in Figure 142 using the RMM CHANGEVOLUME subcommand, DFSMSrmm does not defer the initialization to DFSMSdfp. You must use EDGINERS to label the tape before it can be used.

```
RMM CHANGEVOLUME A12345 INIT(Y)
```

Figure 142. Changing the Initialization Action for a Volume

Setting Status for a Volume in a System-Managed Tape Library

DFSMSrmm maintains volume status in the TCDB for volumes that are defined in the DFSMSrmm control data set. The status is updated using the RMM TSO commands or when volumes are released. Use the DFSMSrmm EDGRMM*xx* parmlib OPTION command SMSTAPE operand to control when the TCDB is updated. If the TCDB is not updated, the request to initialize or erase a volume in a system-managed tape library might fail because the volume is in a scratch library category. If DFSMSrmm is still running in record-only mode, use the AMS ALTER VOLUMEENTRY command to change the volume status. If DFSMSrmm is running in any other mode, you can use the RMM TSO commands to correct the volume status.

To set the correct status for the volume, issue two commands. Issue the following command to complete the failed request.

```
RMM CHANGEVOLUME A12345 CONFIRMRELEASE(INIT)
```

Then issue following command to request that the volume be initialized.

```
RMM CHANGEVOLUME A12345 INIT(Y)
```

Labeling New Tape Volumes with EDGINERS

When you use the EDGINERS utility to label new tape volumes, EDGINERS reads the VOL1 tape label header of any volume that is to be initialized. When DFSMSrmm reads the VOL1 tape label header for new volumes, this can result in IOS000I messages with NCA (Not Capable). EDGINERS identifies the sense information and initializes the volume without further checking or intervention.

When IOS returns the message ″Format incompatible″ for volumes that do not have readable header information, DFSMSrmm cannot determine if the correct volume is mounted. DFSMSrmm issues message EDG6656E and message EDG6661E. In message EDG6661E, DFSMSrmm displays *FMTIC which is a dummy volume serial number used by EDGINERS.

```
EDG6656E FORMAT OF VOLUME M00021 IS NOT COMPATIBLE WITH CURRENT DEVICE
EDG6661E INCORRECT VOLUME MOUNTED ON DEVICE 0281 - REQUESTED VOLUME M00021
MOUNTED VOLUME *FMTIC.
```

To ensure that new tape volumes are labeled, you can use the EDGINERS WRONGLABEL parameter described in “EXEC Parameters for EDGINERS” on page 354 to specify what DFSMSrmm does when it encounters a new tape volume with unreadable header information. Use the WRONGLABEL(RMMPROMPT) parameter if you want DFSMSrmm to prompt the operator for a reply or the WRONGLABEL(IGNORE) parameter when you want EDGINERS to continue without intervention.

Using the Automatic Cartridge Loader with EDGINERS

You can use cartridge loaders with the EDGINERS utility to automate the mounting of volumes that are to be erased or labeled. To use cartridge loaders set to automatic mode, do not pre-mount volumes. Mount the volumes after EDGINERS issues the first mount message because DFSMSrmm processing depends on the mount message which is not issued when the cartridge loader is set to automatic mode and if a volume is already loaded.

If you use brand new volumes or volumes that might cause I/O errors, Automatic Volume Recognition (AVR) can reject premounted volumes at allocation time. AVR processing is not under DFSMSrmm control. AVR dismounts the premounted and readied volumes when an I/O error is detected and issues message IEF503I.

When you use cartridge loaders with EDGINERS, you can optionally use the COUNT operand to specify how many volumes you plan to load into the cartridge loader. When you use the VERIFY operand, specify the COUNT operand to enable the batch of volumes to be reloaded for verify processing before any other volumes are processed. Use the BATCH operand to specify how many batches of volumes you want to process in a single run of EDGINERS

Controlling Access to EDGINERS

You should control access to EDGINERS because it can overwrite previously labeled tapes regardless of expiration date and security protection. DFSMSrmm helps you prevent unauthorized users from using EDGINERS by using the RACF security resource STGADMIN.EDG.OPERATOR. Only users with access to STGADMIN.EDG.OPERATOR can use EDGINERS.

For more information about using STGADMIN.EDG.OPERATOR, see Chapter 9, "Authorizing DFSMSrmm Users and Ensuring Security," on page 171.

How DFSMSrmm Selects an ISO/ANSI Label Version

You can specify an ISO/ANSI label version, or can use the system default to label tape volumes with ISO/ANSI labels.

You can specify the volume label version in several different ways. EDGINERS uses the following selection order for each volume in order to determine the label version, then uses the first value that it finds to assign the label version to the volume.

1. The LABELVERSION parameter in SYSIN command of EDGINERS
2. The REQUIREDLABELVERSION in the DFSMSrmm control data set volume record
3. The EDGINERS EXEC parameter ALVER3 or ALVER4
4. The parameter ALVERSION() in the parmlib DEVSUP*xx* member
5. The system default is 3

Producing Label Symmetry

If you initialize an ISO/ANSI label using EDGINERS, the labels do not frame an empty or null data set as required for interchange. To produce a label symmetry that meets ISO/ANSI standards, at least a minimal open and close sequence must be performed. For example, a volume previously initialized with EDGINERS results in label symmetry when you use the data set utility IEBGENER before the volume leaves the system for interchange, as shown in Figure 143.

```
//STEP1   EXEC  PGM=IEBGENER
//SYSUT1  DD    DUMMY,DCB=(RECFM=F,BLKSIZE=80,LRECL=80)
//SYSUT2  DD    DSN=DUMMY,UNIT=(tape,,DEFER),LABEL=(,AL),
                DCB=(RECFM=F,BLKSIZE=80,LRECL=80)
//SYSIN   DD    DUMMY
```

Figure 143. Using IEBGENER

How EDGINERS Processing Works

Before initializing or erasing a tape volume, DFSMSrmm verifies that the correct volume is mounted by reading any existing label, obtaining the volume serial number from the tape library vision system, or prompting the operator to confirm that the external label is correct. EDGINERS labels tape volumes as follows:

- For standard label volumes, DFSMSrmm creates a standard volume label, an 80-byte dummy header label, and a tape mark.

  A standard volume label with a serial number you specify and a blank security byte. The format of the ISO/ANSI label is constructed either for Version 3 or Version 4, depending on the options, of the label standard. For a complete description of IBM standard volume labels and ISO/ANSI Version 3 volume labels, see *z/OS DFSMS: Using Magnetic Tapes*.

  An 80-byte dummy header label. For IBM standard labels, this header consists of the characters HDR1 followed by zeros. For ISO/ANSI labels, this header consists of the characters HDR1 followed by zeros, with the following exceptions:

  – Position 54, which contains an ISCII/ASCII space
  – A 1 in the file section, file sequence, and generation number fields
  – A leading space in the creation and expiration date fields

  - A system code of IBMZLA, followed by 13 spaces, to identify the operating system creating the label
- For tapes with no label, DFSMSrmm writes an 80-byte record that is not recognized as any valid label format, however, the DFSMSrmm utility can recognize it as an unlabeled tape once it is initialized.

The first time a tape that is labeled by EDGINERS is used for output, the following sequence of events occurs:
1. The tape mark that EDGINERS created is overwritten.
2. The dummy header label that EDGINERS created is filled in with operating system data and device-dependent information.
3. A HDR2 record, containing data set characteristics, is created.
4. User header labels are written if exits to user label routines are provided.
5. A new tape mark is written.
6. Data is placed on the receiving volume.

When relabeling a volume defined to DFSMSrmm, DFSMSrmm uses the information recorded about the old volume as part of the new volume information recorded in the control data set.

The following information is taken from the old volume record: Expiration Date, Density, Use Count, Store Id, Bin Number, Old Bin Number, Loan Location, Previous Location, Last Read and Write dates, Assigned Date and Time, Owner, Status, Label Type, Release actions, Actions Pending (excluding initialize and erase), Volume access and accessors, Unit name, Rack number, Temporary / Permanent Read / Write error counts.

When a volume that resides in an automated system-managed tape library is rejected at OPEN time because the tape media servo tracks require formatting, DFSMSrmm updates the volume information in the control data set. DFSMSrmm sets the volume INIT action to indicate that the volume must be initialized. If EDGINERS is used to relabel a volume and servo tracks have not been formatted, DFSMSrmm cannot initialize the volume.

Return Codes for EDGINERS

EDGINERS issues one of the return codes shown in Table 53.

Table 53. EDGINERS Return Codes

Return Code	Explanation
0	All requested functions completed successfully.
4	DFSMSrmm encountered a minor error during processing. It issues a warning message and continues processing.
8	DFSMSrmm has stopped at least one requested function. It continues processing the next requested function.
12	DFSMSrmm encountered a severe error during processing of one of the requested functions. DFSMSrmm stops the utility.
16	DFSMSrmm encountered a severe error during a required communication with the DFSMSrmm subsystem. DFSMSrmm stops the utility.

EDGINERS Examples

The following examples illustrate some of the uses of EDGINERS. To use the examples, replace the **tape** specified in the examples with actual device numbers or esoteric unit names, unless your installation has the required device numbers defined to the esoteric name ‘TAPE’. The actual device numbers and esoteric unit names depend on how your installation has defined the devices to your system. See *z/OS DFSMS: Using Magnetic Tapes* for devices supported by this utility.

Example 1: Write IBM Standard Labels on Three Tapes

Figure 144 is a manual processing example. Serial numbers 001234, 001235, and 001236 are placed on three tape volumes. The labels are written in EBCDIC at 800 bits per inch. Each volume labeled is mounted, when it is required, on a single 9-track tape unit. You must specify SYSIN commands for each volume you want to label, as Figure 144 shows.

```
//LABEL1   JOB  ...
//STEP1    EXEC PGM=EDGINERS
//SYSPRINT DD  SYSOUT=A
//TAPE     DD  DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN    DD  *
INIT VOLUME(001234) LABEL(SL)
INIT VOLUME(001235) LABEL(SL)
INIT VOLUME(001236) LABEL(SL)
/*
```

Figure 144. Writing IBM Standard Labels on Three Tapes

Control statement description:
- TAPE DD defines the tape unit used in the labeling operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- INIT specifies that the tapes are to be labeled.

Example 2: Write an ISO/ANSI Label on a Tape

Figure 145 is a manual processing example. Serial number 001001 is placed on one ISO/ANSI tape volume; the label is written at 800 bits per inch. The volume labeled is mounted, when it is required, on a 9-track tape unit. You must specify SYSIN commands for each volume you want to label, as Figure 145 shows.

```
//LABEL2   JOB  ...
//STEP1    EXEC PGM=EDGINERS
//SYSPRINT DD  SYSOUT=A
//TAPE     DD  DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN    DD  *
INIT VOLUME(001001) LABEL(AL)
/*
```

Figure 145. Writing an ISO/ANSI Label on a Tape

Control statement description:
- TAPE DD defines the tape unit to be used in the labeling operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- INIT specifies the serial number and label type for the volume that is being labeled.

Example 3: Place Two Groups of Serial Numbers on Six Tape Volumes

Figure 146 is a manual processing example. Two groups of serial numbers (001234, 001235, 001236, and 001334, 001335, 001336) are placed on six tape volumes. The labels are written in EBCDIC at 800 bits per inch. Each volume labeled is mounted, when it is required, on a single 9-track tape unit. You must specify SYSIN commands for each volume you want to label, as Figure 145 on page 368 shows.

```
//LABEL3   JOB  ...
//STEP1    EXEC PGM=EDGINERS
//SYSPRINT DD  SYSOUT=A
//TAPE     DD  DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN    DD  *
INIT  VOL(001234) LABEL(SL)
INIT  VOL(001235) LABEL(SL)
INIT  VOL(001236) LABEL(SL)
INIT  VOL(001334) LABEL(SL)
INIT  VOL(001335) LABEL(SL)
INIT  VOL(001336) LABEL(SL)
/*
```

Figure 146. Numbering Tape Volumes

Control statement description:
- TAPE DD defines the tape unit to be used in the labeling operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- INIT specifies that the tapes are to be labeled.

Example 4: Place Serial Numbers on Eight Tape Volumes

Figure 147 is a manual processing example. Serial numbers 001234, 001244, 001254, 001264, 001274, and so on in a sequence, are placed on eight tape volumes. The labels are written in EBCDIC at 800 bits per inch. Each volume labeled is mounted, when it is required, on a single 9-track tape unit. You must specify SYSIN commands for each volume you want to label, as Figure 147 shows.

```
//LABEL4   JOB  ...
//STEP1    EXEC PGM=EDGINERS
//SYSPRINT DD  SYSOUT=A
//TAPE     DD  DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN    DD  *
INIT VOLUME(001234) LABEL(SL)
INIT VOLUME(001244) LABEL(SL)
INIT VOLUME(001254) LABEL(SL)
INIT VOLUME(001264) LABEL(SL)
INIT VOLUME(001274) LABEL(SL)
INIT VOLUME(001284) LABEL(SL)
INIT VOLUME(001294) LABEL(SL)
INIT VOLUME(001304) LABEL(SL)
/*
```

Figure 147. Placing Serial Numbers on Eight Tape Volumes

Control statement description:
- TAPE DD defines the tape unit used in the labeling operation.
- SYSIN DD defines the control data set, which follows in the input stream.
- The INIT statements specify the tapes to be labeled.

Example 5: Relabel a Volume

Figure 148 is a manual processing example for relabeling a volume. Supply both the current and the new volume serial number in your SYSIN statement. In the example the volume that is currently labeled as CURR01 is relabeled to NEW001.

If volume CURR01 is already defined to DFSMSrmm, DFSMSrmm defines a new volume entry NEW001 to the DFSMSrmm control data set using information from the existing volume entry CURR01 and deletes the existing volume entry. If volume CURR01 is not defined to DFSMSrmm, DFSMSrmm defines a new volume entry NEW001 in the control data set.

```
//INIT     EXEC PGM=EDGINERS
//SYSPRINT DD SYSOUT=*
//TAPE     DD UNIT=(tape,,DEFER)
//SYSIN    DD *
INIT VOLUMEUME(CURR01,NEW001) LABEL(SL)
```

Figure 148. Relabeling a Volume

Control statement description:
- TAPE DD defined the tape unit to be used in the labeling operation.
- INIT specifies the current and new tape labels.

Example 6: Automatically Initialize or Erase 3480 Volumes

Figure 149 is an automatic processing example. EDGINERS requests that DFSMSrmm scan its control data set for the first ten 3480 volumes waiting to be initialized or erased.

```
//LABEL5   JOB  ...
//STEP1    EXEC PGM=EDGINERS,PARM='INITIALIZE,ERASE,COUNT(10)'
//SYSPRINT DD SYSOUT=*
//TAPE     DD UNIT=(3480,,DEFER)
```

Figure 149. Automatically Initialize or Erase 3480 Volumes

Control statement description:
- TAPE DD defines the tape unit used in the labeling operation.

Example 7: Initialize and Erase Volumes in a System-Managed Library

In Figure 150 an automatic run of EDGINERS is scheduled to find, initialize, and erase up to 99 volumes residing in an automated tape library called MYATL. All tape cartridges will be labeled as appropriate for the drive type on which they are mounted, and for their current media characteristics.

```
//LABEL6 JOB ....
//STEP1 EXEC PGM=EDGINERS,
//     PARM='COUNT(99),LOCATION(MYATL),INITIALIZE,ERASE'
//SYSPRINT  DD  SYSOUT=A
```

Figure 150. Initialize and Erase Volumes in a System-Managed Library

Control statement description:
- | TAPE and SYSIN DD are not required.
- PARM values request automatic processing with the default of VERIFY of all labeled volumes.

Example 8: Initialize 50 Scratch Enhanced Capacity Cartridges

In Figure 151, an automatic run of EDGINERS is used to initialize 50 scratch enhanced capacity cartridges defined to DFSMSrmm in the previous job step, and initialize them for use in a non-system managed tape library. This example assumes that no other volumes in the pool need to be initialized.

```
//LABEL7  JOB ...
//STEP1 EXEC PGM=IKJEFT01
//SYSTPRINT  DD  SYSOUT=A
//SYSTSIN  DD  *
  RMM ADDVOLUME A00100 STATUS(SCRATCH) INIT(Y) -
          COUNT(50) LOCATION(SHELF) MEDIATYPE(ECCST)
/*
//STEP2 EXEC PGM=EDGINERS,
//    PARM='COUNT(50),POOL(A*),INITIALIZE,NOVERIFY'
//SYSPRINT  DD  SYSOUT=A
//TAPE  DD  UNIT=(tape,1,DEFER)
```

Figure 151. Initialize 50 Scratch Enhanced Capacity Cartridges

Control statement description:
- SYSTSIN DD includes commands to define required volumes.
- TAPE DD allocates a drive capable of labeling enhanced capacity cartridges.

Example 9: Erase a Volume

In Figure 152, volume 0A7100 is erased and defined to DFSMSrmm. The volume is successfully erased and the volume label is written as an ISO/ANSI label.

```
//LABEL8 JOB ....
//STEP1 EXEC PGM=EDGINERS
//SYSPRINT  DD  SYSOUT=A
//TAPE  DD  UNIT=(tape,,DEFER)
//SYSIN  DD  *
  ERASE VOL(0A7100) LABEL(AL)
/*
```

Figure 152. Erase a Volume

Control statement description:
| - TAPE DD allocates a drive suitable for this volume.
- SYSIN DD includes the command requested to erase the volume.

Example 10: Initialize Volumes Using Multiple Tape Drives

Figure 153 on page 372 shows how to run multiple copies of EDGINERS to automatically label volumes driven by initialize actions in the DFSMSrmm control data set. Two jobs are provided but you can run one for each tape drive that you want to use. Change the jobname for each copy.

```
//LABEL10A JOB ......
 //AUTOINIT EXEC PGM=EDGINERS,PARM='INITIALIZE,BATCH(0)',
 // 'NOVERIFY,MEDIANAME(CARTS)'
 //SYSPRINT DD SYSOUT=*
 //TAPE DD UNIT=(tape,,DEFER)
 //
 //LABEL10B JOB ......
 //AUTOINIT EXEC PGM=EDGINERS,PARM='INITIALIZE,BATCH(0)',
 // 'NOVERIFY,MEDIANAME(CARTS)'
 //SYSPRINT DD SYSOUT=*
 //TAPE DD UNIT=(tape,,DEFER)
```

Figure 153. Initialize Volumes Using Multiple Tape Drives

Control statement description:

| • TAPE DD allocates a drive suitable for this volume.

Example 11: Labeling Duplicate Volumes Using EDGINERS

You can label volumes as duplicate volumes using the EDGINERS utility that uses manual processing, with the VOL1 operand on the command in the SYSIN file, as shown in the example in Figure 154. The VOL1 value is written to the tape label.

```
//MANUAL   EXEC PGM=EDGINERS
//SYSPRINT DD SYSOUT=*
//TAPE     DD UNIT=(tape,,DEFER)
//SYSIN    DD *
INIT VOLUME(D00001) LABEL(SL) VOL1(ABC123)
/*
```

Figure 154. Labeling Duplicate Volumes Using EDGINERS

Control statement description:

| • TAPE DD allocates a drive suitable for this volume.

Example 12: Selecting EHPCT Volumes for Processing Automatically

You can select volumes for processing as shown in the example in Figure 155.

```
//INIT     EXEC PGM=EDGINERS,
// PARM='MEDIATYPE(EHPCT),NOVERIFY,BATCH(0),STATUS(NOTMASTER)'
//SYSPRINT DD SYSOUT=*
//TAPE     DD UNIT=(TAPEEH,,DEFER)
```

Figure 155. Selecting Volumes for Automatic Processing

Control statement description:
- MEDIATYPE(EHPCT) is used to select only those volumes that are EHPCT media. EDGINERS processing is performed without operator intervention by using the DFSMSrmm control data set as input.
- BATCH(0) ensures that all volumes that need to be processed are processed.
- STATUS(NOTMASTER) is used to select only scratch volumes and volumes that are pending release.
- TAPE DD uses your installation's unit names to select a suitable drive on which to mount EHPCT volumes. Change the TAPEEH value to the value required for your installation. If all tapes are system-managed, you can remove the TAPE DD because DFSMSrmm dynamically allocates the drives that are required, and the TCDB media type ensures the correct drives are allocated.

Chapter 17. Customizing DFSMSrmm

This chapter helps you customize DFSMSrmm to best meet your installation's needs.

Changing the Initial Entry Point to the DFSMSrmm Dialog

Normally, when you enter the DFSMSrmm ISPF dialog, the first panel you see is the Primary Option Menu. You can change this initial entry point so tape librarians always see the Librarian Menu when they enter DFSMSrmm or general users see the DFSMSrmm User Menu.

To change the initial entry point, use the RMMISPF exec with one of its optional parameters which changes the panel navigation to go directly to a lower level. Figure 156 shows the operands you can use with the RMMISPF exec.

```
►►──RMMISPF──┬────────────┬──┬────────────────┬──────────────────────────►◄
             ├─ADMIN──────┤  └──TRACE(option)─┘
             ├─COMMAND────┤
             ├─CONTROL────┤
             ├─DATASET────┤
             ├─LIBRARIAN──┤
             ├─LOCAL──────┤
             ├─OWNER──────┤
             ├─PRODUCT────┤
             ├─RACK───────┤
             ├─REPORT─────┤
             ├─SUPPORT────┤
             ├─USER───────┤
             ├─VOLUME─────┤
             └─VRS────────┘
```

Figure 156. RMMISPF Exec Syntax Diagram

Each of the operands, except for TRACE, represents a specific user or function menu from which you can request functions. Use the TRACE(option) operand to diagnose problems in any of the REXX execs supplied in the dialog. For more information about using TRACE, see *z/OS DFSMSrmm Diagnosis Guide*.

To start the librarian at the Librarian Menu, add the following line to the ISPF selection panel as shown in Figure 157 on page 374.

```
%    R +DFSMSrmm   - Librarian dialog
```

Although the example shows the ISRUTIL panel, you can use any other selection panel to make the changes.

```
%-------------------------  UTILITY SELECTION MENU  ---------------------
%OPTION  ===>_ZCMD
%
%    1 +LIBRARY    - Compress or print data set.  Print index listing.
+                     Print, rename, delete, or browse members
%    2 +DATASET    - Allocate, rename, delete, catalog, uncatalog, or
+                     display information of an entire data set
%    3 +MOVE/COPY  - Move, copy, or promote members or data sets
%    4 +DSLIST     - Print or display (to process) list of data set names
+                     Print or display VTOC information
%    5 +RESET      - Reset statistics for members of ISPF library
%    6 +HARDCOPY   - Initiate hardcopy output
%    8 +OUTLIST    - Display, delete, or print held job output
%    9 +COMMANDS   - Create/change an application command table
%   10 +CONVERT    - Convert old format menus/messages to new format
%   11 +FORMAT     - Format definition for formatted data Edit/Browse
%   12 +SUPERC     - Compare data sets (Standard dialog)
%   13 +SUPERCE    - Compare data sets (Extended dialog)
%   14 +SEARCH-FOR - Search data sets for strings of data
%    R +DFSMSrmm   - Librarian dialog
)INIT
  .HELP = ISR30000
)PROC
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
               1,'PGM(ISRUDA) PARM(ISRUDA1)'
               2,'PGM(ISRUDA) PARM(ISRUDA2)'
               3,'PGM(ISRUMC)'
               4,'PGM(ISRUDL) PARM(ISRUDLP)'
               5,'PGM(ISRURS)'
               6,'PGM(ISRUHC)'
               8,'PGM(ISRUOLP)'
               9,'PANEL(ISPUCMA)'
              10,'PGM(ISRQCM) PARM(ISRQCMP)'
              11,'PGM(ISRFMT)'
              12,'PGM(ISRSSM)'
              13,'PGM(ISRSEPRM) NOCHECK'
              14,'PGM(ISRSFM)'
               R,'CMD(%RMMISPF LIBRARIAN) NEWAPPL(EDG)'
             ' ',' '
               *,'?' )
  &ZTRAIL = .TRAIL
)END
```

Figure 157. Adding DFSMSrmm Librarian Option to ISPF

Adding Local Dialog Extensions

You can add your own local extensions to the DFSMSrmm ISPF dialog. DFSMSrmm has a dummy panel, EDGP@LCL, that provides easy access to local dialog extensions. You can use these extensions without having to modify DFSMSrmm. Use the following steps to add your extensions:

1. Create a selection menu, EDGP@LCL, that includes the extensions you want to add.
2. For each extension, create the required procedures and panels.
3. Make the local menu available to ISPF in a panel library so that it is selected ahead of the menu DFSMSrmm supplies.
4. Start the DFSMSrmm ISPF dialog.
5. Select Option 6 (LOCAL). You should see the modified local selection menu you just created.

You do not have to limit the local extensions to functions that use DFSMSrmm commands and utilities. Add any function that is useful to DFSMSrmm users, such as the tape librarian and storage administrator. For example, you could add:

- Service level reporter charts on tape activities
- A link to ISMF or RACF dialogs
- Lists of DFSMShsm volumes to compare with DFSMSrmm definitions
- A volume loan facility that enforces local standards for location names

When including RMM TSO subcommands, use the REXX procedure language instead of CLIST. RMM TSO subcommands set REXX variables so you do not need to trap and interpret the command output. Refer to the REXX procedures DFSMSrmm supplies for examples of using the subcommands and receiving variables. Do not modify the supplied execs because they can change from one release to the next.

For more information on using the RMM TSO subcommands within REXX and the variables each subcommand issues, see *z/OS DFSMSrmm Guide and Reference*.

Customizing the Local Dialog with ’U’ Line Command

You can customize the dummy panel, EDGP@LCL, to add facilities to the ISPF dialog. The ’U’ line command allows you locally-provided line command support and calls the EDGRLCL exec. A set of variables are saved in the variable pool for use by the EDGRLCL exec. You can use this exec to implement any local extensions to the search results line commands. If the exec does not exist, for example, the installation has not provided one, one of the following error messages is displayed: U line command not in use or You must use the EDGRLCL exec to implement the U line command.

The set of variables saved for use by the EDGRLCL exec depends on the search results list being processed. Each of the fields in the current row of the table are available as well as the table name, an indication of what type of search the results are for, the current row, and other table settings that may be required by the exec. The exec does not need to process all rows of the table, only the current row. It is called for each row of the table for which the ’U’ line command has been specified.

Changing the ADD Product Volume Defaults

When you use the DFSMSrmm ISPF dialog to define product volumes to DFSMSrmm, the dialog issues either the RMM TSO CHANGEVOLUME or ADDVOLUME subcommand.

When ADDVOLUME is used, the dialog has some hard-coded default values set for RETPD and RELEASEACTION. You can modify the dialog EXEC EDGRPADV to change the values to ones that suit your installation.

The EDGRPADV dialog default is:

```
command = edgcmd" ADDVOLUME "edg@vol" STATUS(MASTER)" ,
          "NUMBER('"edgraddq(edg@pnum)"') LEVEL("edg@ver")",
          "FEAT("edg@fcd")" ,
          "MEDIANAME('"edgraddq(edg@medn)"')",
          "LOCATION("edg@loc") RETPD(90) RELEASE(RETURN)"
```

To modify the values that the EDGRPADV EXEC uses, install your changes using an SMP/E USERMOD after editing a copy of the EDGRPADV EXEC. The only modifications you should make are to either add new operands to the ADDVOLUME command, or to change the values set for the MASTER, RETPD, and RELEASE operands.

Customizing DFSMSrmm Messages for Report Titles and User Notification

DFSMSrmm provides messages for report titles and user notification. You can customize them by performing these tasks:

1. Updating the text of a message in the DFSMSrmm message table EDGMTAB.
2. Applying changes to EDGMTAB by creating an SMP/E-installable USERMOD.

Customizing DFSMSrmm Report Titles

You can customize the title text on DFSMSrmm reports. For example, you can add your company's name to the bottom of each report page.

To customize, update the text of a message in the DFSMSrmm message table EDGMTAB. The report utilities get the message text from the message table and use it when creating the report trailer lines. Table 54 shows the message numbers DFSMSrmm uses for each report utility:

Table 54. Customizing Report Titles

Utility	Message Number	Used For
EDGAUD	5839	security report trailer 1
EDGAUD	5840	security report trailer 2
EDGAUD	5846	audit report trailer 1
EDGAUD	5847	audit report trailer 2
EDGHSKP	2203	EDGHSKP REPORT file header
EDGRPTD	5825	trailer line 1

To customize the message text, edit the message text in the EDGMTAB source, supplied with the product. You could change trailer 1 for EDGRPTD, as shown in the following examples. Figure 158 shows the text before modification.

```
EDGMSGB 5825,TYPE=I,                                        X
      '                                 ',                  X
      MOD=NO
```

Figure 158. Before Modifying the Trailer 1 for EDGRPTD

Figure 159 shows the text after modification.

```
EDGMSGB 5825,TYPE=I,                                        X
      'Warwick Corporation',                                X
      MOD=NO
```

Figure 159. After Modifying the Trailer 1 for EDGRPTD

The report utility centers the text before printing.

Customizing Notification Messages and Notes

With DFSMSrmm you can set up automatic notification to owners by setting a parmlib option and defining owner electronic address to DFSMSrmm. Refer to Chapter 8, “Using the Parmlib Member EDGRMMxx,” on page 127 for information on setting up automatic notification and controlling message text case.

DFSMSrmm provides a series of messages in the DFSMSrmm message module, EDGMTAB. You can modify them to provide specific information to users:

Messages EDG2405 through EDG2409
Notify users that volumes they own are eligible for release.

Messages EDG2700 through EDG2713
Create a note you send to product owners when new volumes for a product are entered into the library.

When modifying messages, you should:

- Keep message variables in the same order as in the original message.
- Use the same number of substitution characters that is shown in the original message text.
- Keep the messages in the same order they are displayed.
- Do not delete any messages from EDGMTAB. If you do not want to use a particular message, leave the message text blank.

Modifying Text for Release Notification

Modify messages 2405-2409 to change the message text to notify users that their volumes are eligible for release.

You can substitute the message text without restriction for all the messages except message 2406. For message 2406, you must specify the secondary currency character X'4A' to represent substitution characters for variables that DFSMSrmm supplies. The secondary currency character shown in Figure 160 on page 378 is ¢. You must also keep the variables in the order shown in Figure 160 on page 378. The &NAM variable shown in message 2405 is replaced with the product name and need not be specified.

Figure 160 on page 378 shows the messages in EDGMTAB before modification.

```
          EDGMSGB 2405,TYPE=I,                                          X
                'Subject: &NAM volume expiration',                      X
                MOD=NO,MSGID=NO
          SPACE 2
          EDGMSGB 2406,TYPE=I,                                          X
                'Volume ¢¢¢¢¢¢ assigned to owner ¢¢¢¢¢¢¢¢ on ¢¢¢¢¢¢¢¢¢¢ X
                at ¢¢:¢¢:¢¢',                                           X
                MOD=YES,MSGID=NO
          SPACE 2
          EDGMSGB 2407,TYPE=I,                                          X
                'is now pending release.',                              X
                MOD=NO,MSGID=NO
          SPACE 2
          EDGMSGB 2408,TYPE=I,                                          X
                'If you wish the volume to be retained, please take immeX
                diate action.',                                         X
                MOD=NO,MSGID=NO
          SPACE 2
          EDGMSGB 2409,TYPE=I,                                          X
                'You can use the dialog functions or the RMM CHANGEVOLUMX
                E TSO command.',                                        X
                MOD=NO,MSGID=NO
```

Figure 160. Notify Owner Messages

Figure 161 shows the notification text that results from Figure 160.

```
Subject: DFSMSrmm volume expiration
Volume LAUREN assigned to owner KRISTINE on 1998/01/01 at 09:45:11
is now pending release.
If you wish the volume to be retained, please take immediate action.
You can use the dialog functions or the RMM CHANGEVOLUME TSO COMMAND.
```

Figure 161. Default Notification Text

Figure 162 shows the same messages after modification.

```
          EDGMSGB 2405,TYPE=I,                                          X
                'Subject:   Bld 88 Tape Volume Expiration               X
                MOD=NO,MSGID=NO
          SPACE 2
          EDGMSGB 2406,TYPE=I,                                          X
                'Volume ¢¢¢¢¢¢ assigned to owner ¢¢¢¢¢¢¢¢ on ¢¢¢¢¢¢¢¢¢¢ x
                at ¢¢:¢¢:¢¢',                                           x
                MOD=YES,MSGID=NO
          SPACE 2
          EDGMSGB 2407,TYPE=I,                                          X
                'has now expired.  Please come to the Computer Room Tapex
                 Library Window',                                       x
                MOD=NO,MSGID=NO
          SPACE 2
          EDGMSGB 2408,TYPE=I,                                          X
                'to pick up the tape.  Contact the tape librarian at 555x
                -5796 for any',                                         x
                MOD=NO,MSGID=NO
          SPACE 2
          EDGMSGB 2409,TYPE=I,                                          X
                'questions.  DO NOT REPLY to this automated system messax
                ge.  Thanks.  ',                                        x
                MOD=NO,MSGID=NO
          SPACE 2
```

Figure 162. Modified Messages

Figure 163 on page 379 shows the text that results from Figure 162.

```
Subject: Bld 88 Tape Volume Expiration.
Volume LAUREN assigned to owner KRISTINE on 1998/01/01 at 09:45:11
has now expired.  Please come to the Computer Room Tape Library Window
to pick up the tape.  Contact the tape librarian at 555-5796 for any
questions.  DO NOT REPLY to this automated system message.  Thanks.
```

Figure 163. Modified Notification Text

Modifying Text in Notes to Product Owners

DFSMSrmm provides note text for notifying product owners under the following conditions:

- A new software product volume is added to DFSMSrmm.
- A change is made to add a volume to a product.
- The software level of a product volume is changed.

You can modify the note by modifying messages EDG2700 through EDG2713. Figure 164 shows an example of a note produced by modifying EDG2700 through EDG2713.

```
Subject: Volume PP0001 has been added for software product 5647-A01
A volume has been added on 1999/12/27 at 20:19:31 to a DFSMSrmm
software product which you own:

 Product Number = 5694-A01  Level = V01R01M00
 Name           = z/OS  Version 1.1
 Description    = includes rmm

 Volume   Rack     Feature Code   Description
 ------   ------   ------------   ------------------------------
 PP0001   PP1233   5678           z/OS     V1.1 (volume 1 of 1)
```

Figure 164. Notifying Product Owner

Figure 165 shows the messages you can modify. The messages must be displayed in sequence to produce the note. Several messages consist of blank lines so you have flexibility in the note you send. You can also line up the columns in your message text as shown in message EDG2710.

```
          EDGMSGB 2700,TYPE=I,                                        X
                'Subject: Volume ¢¢¢¢¢¢ has been added for software prodX
                uct ¢¢¢¢¢¢¢¢',                                        X
                MOD=YES,MSGID=NO
          SPACE 2
          EDGMSGB 2701,TYPE=I,                                        X
                'A volume has been added on ¢¢¢¢¢¢¢¢¢¢ at ¢¢:¢¢:¢¢ to a X
                &NAM',                                                X
                MOD=YES,MSGID=NO
          SPACE 2
          EDGMSGB 2702,TYPE=I,                                        X
                'software product which you own:',                    X
                MOD=NO,MSGID=NO
```

Figure 165. Notify Product Owner Messages (Part 1 of 3)

```
SPACE 2
EDGMSGB 2703,TYPE=I,                                          X
      '                                                       X
                 ',                                           X
      MOD=NO,MSGID=NO
SPACE 2
EDGMSGB 2704,TYPE=I,                                          X
      ' Product Number = ¢¢¢¢¢¢¢¢  Level = V¢¢R¢¢M¢¢',        X
      MOD=YES,MSGID=NO
SPACE 2
EDGMSGB 2705,TYPE=I,                                          X
      ' Name           = ¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢',     X
      MOD=YES,MSGID=NO
SPACE 2
EDGMSGB 2706,TYPE=I,                                          X
      ' Description    = ¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢',     X
      MOD=YES,MSGID=NO
SPACE 2
EDGMSGB 2707,TYPE=I,                                          X
      '                                                       X
                 ',                                           X
      MOD=NO,MSGID=NO
```

Figure 165. Notify Product Owner Messages (Part 2 of 3)

```
SPACE 2
EDGMSGB 2708,TYPE=I,                                          X
      ' Volume   Rack     Feature Code   Description',        X
      MOD=NO,MSGID=NO
SPACE 2
EDGMSGB 2709,TYPE=I,                                          X
      ' ------   ------   ------------   ---------------------X
      ---------',                                             X
      MOD=NO,MSGID=NO
SPACE 2
EDGMSGB 2710,TYPE=I,                                          X
      ' ¢¢¢¢¢¢   ¢¢¢¢¢¢  ¢¢¢¢          ¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢X
       ¢¢¢¢¢¢¢¢¢',
      MOD=YES,MSGID=NO
SPACE 2
EDGMSGB 2711,TYPE=I,                                          X
      '                                                       X
                 ',                                           X
      MOD=NO,MSGID=NO
SPACE 2
EDGMSGB 2712,TYPE=I,                                          X
      '                                                       X
                 ',                                           X
      MOD=NO,MSGID=NO
SPACE 2
EDGMSGB 2713,TYPE=I,                                          X
      '                                                       X
                 ',                                           X
      MOD=NO,MSGID=NO
```

Figure 165. Notify Product Owner Messages (Part 3 of 3)

Managing VM Tape Volumes

DFSMSrmm Samples Provided in SAMPLIB

- EDGCLIBQ Sample Exec to Create Reports for VM Tape Volumes
- EDGJVME Sample JCL for Creating Reports for VM Tape Volumes

DFSMSrmm provides SAMPLIB members to help you manage VM tapes. The sample, EDGJVME, is a batch job that creates a list of volumes that are flagged for VM use. EDGJVME gets the information from the extract data set. The job sends this list to the relevant VM system.

When you have received the list on the VM system, you can use the sample, EDGCLIBQ. EDGCLIBQ is an exec that produces information about the volumes on the list, including shelf location, owner, and users that are allowed to access the volumes.

DFSMSrmm provides the LIBQ exec to help you query information about VM volumes. Use the LIBQ exec to get the location where a volume resides. This example describes one scenario for using the LIBQ exec:

1. A user requests a volume to be mounted on VM by sending a message to the operator.
2. The operator issues the LIBQ exec, with the volume serial number requested as 'LIBQ volser'.
3. The LIBQ exec returns the rack number where the volume resides to the operator.
4. The operator decides whether to mount the volume as requested by the user.
5. The operator attaches the drive with the volume mounted, as requested.

To use the LIBQ exec to find a scratch volume, modify the exec to search for scratch volumes in the file it processes. You can tailor the rules you want to use for 'in use' or scratch volumes, or modify the exec and use it as part of your VM tape automation. You might also tailor the process to work with other non-MVS platforms.

Replenishing Scratch Volumes in a System-Managed Library

DFSMSrmm Sample Provided in SAMPLIB

- EDGXPROC Sample to Replenish Scratch Volumes in a System-Managed Library
- EDGSETT Sample that you can use in IBM Tivoli Workload Scheduler for z/OS in place of EDGXPROC.

When a tape library detects a low-on-scratch condition, where more scratch volumes are needed, OAM issues a write-to-operator message (CBR3660A). DFSMSrmm intercepts the message and starts the procedure you define with the SCRATCHPROC value in parmlib. See "Defining System Options: OPTION" on page 134 for more information about specifying SCRATCHPROC. You must run DFSMSrmm with a scratch procedure. You can modify the DFSMSrmm supplied sample, EDGXPROC, to support your location procedures. You can use the scratch procedure to take any action you would like. For example, you can code the procedure to trigger the required inventory management expiration processing job, to run inventory management, or to take no action.

You can use the DFSMSrmm supplied sample procedure, EDGXPROC, which runs DFSMSrmm expiration processing to replenish the automated tape library's scratch volumes. If you use the DFSMSrmm supplied sample, EDGXPROC, you must define the procedure in the installation procedure library, SYS1.PROCLIB as described in "Step 8: Updating the Procedure Library" on page 31.

You can modify this procedure, for example, to alert the operator if scratch volumes are not replenished. Figure 166 shows the EDGXPROC procedure.

```
//EDGXPROC PROC OPT=EXPROC
//EDGHSKP  EXEC PGM=EDGHSKP,
//         PARM='&OPT.'
//SYSPRINT DD SYSOUT=*
//MESSAGE  DD DSN=MESSAGE.FILE.NAME,DISP=SHR
```

Figure 166. EDGXPROC Procedure

DFSMSrmm keeps track of the time and date the procedure starts and when it last ran expiration processing to ensure that one procedure completes processing before a new procedure begins. You can display this information by using the RMM LISTCONTROL subcommand with the CNTL operand.

Automating Backup

DFSMSrmm Sample Provided in SAMPLIB

EDGBETT sample that you can use in the IBM Tivoli Workload Scheduler for z/OS for automating backup.

Create a backup procedure in the system procedure library. Figure 167 shows an example of a procedure that runs backup as part of inventory management. Alternatively, you might want to use the procedure to submit a batch job to perform the backup or to inform your job scheduler to submit the job.

Specify your backup procedure name with the BACKUPPROC value in parmlib. See "Defining System Options: OPTION" on page 134 for more information about specifying BACKUPPROC.

```
//CDSBKUP  PROC
//EDGHSKP  EXEC PGM=EDGHSKP,PARM='BACKUP(DSS)'
//MESSAGE  DD DISP=SHR,DSN=RMM.MESSAGE
//SYSPRINT DD SYSOUT=*
//BACKUP   DD DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.CDS(+1),
//         LABEL=(,SL)
//JRNLBKUP DD DISP=(,CATLG),UNIT=TAPE,DSN=BACKUP.JRNL(+1),
//         LABEL=(2,SL),VOL=REF=*.BACKUP
//         PEND
```

Figure 167. Example BACKUPPROC procedure

Using the LABEL Procedure

You can run the EDGINERS utility as an operator started procedure, so that the operator or librarian can make requests for tape labeling and erasing.

Figure 168 on page 383 shows a copy of the sample procedure you can use in your installation. Rename the procedure to LABEL. Specify additional processing options that suit the needs of your installation.

You must define the LABEL procedure as a RACF started procedure. The LABEL procedure requires UPDATE access to the STGADMIN.EDG.OPERATOR RACF facility class profile. See Chapter 9, "Authorizing DFSMSrmm Users and Ensuring Security," on page 171 for information about STGADMIN.EDG.OPERATOR.

See *z/OS DFSMSrmm Guide and Reference* for information on using the LABEL procedure as part of your operator's tasks.

```
//LABEL   PROC OPT=NOVERIFY,U=3480,SOUT=DUMMY
//*
//*      RMM procedure for volume labelling and erasing
//*        See DFSMS/MVS Version 1 Release 2
//*            DFSMSrmm Guide and Reference SC26-4931
//*        for details of the LABEL utility.
//*
//*      &OPT are the options for the EDGINERS utility.
//*        See DFSMS/MVS Version 1 Release 2
//*            DFSMSrmm Implementation and Customization Guide SC26-4932
//*        for details of the EDGINERS utility.
//*
//*      &U is the device type or number requested for a tape drive
//*
//*      &SOUT Allows you to specify that the message file is
//*      printed. For example S LABEL,SOUT='SYSOUT=A'
//*
//*      If either VERIFY OR NOVERIFY is the only option specified
//*      in the &OPT parameter, requests are entered via the
//*      system console as WTOR replies.
//*
//INIT     EXEC PGM=EDGINERS,PARM='&OPT'
//SYSPRINT DD  &SOUT
//TAPE     DD  UNIT=(&U,,DEFER)
```

Figure 168. Sample Label Procedure

Processing NL Label Tapes: EDG019VM

The EDG019VM sample exit that DFSMSrmm provides can be used as a replacement for IFG019VM. The exit uses the tape volume mount exit to obtain tape label information from the operator when an NL tape is mounted as a scratch tape on a non-specific NL mount request.

Input

The invocation environment must be identical to the environment that is provided at entry to the exit IFG019VM.

Output

The parameter list provided as input is updated. R15 is set on exit with one of the possible values for the return codes:

0	Accept the volume. The parameter list might have been updated.
4	Continue normal processing. The parameter list has not been updated.
8	Reject the volume.

Processing

EDG019VM returns the rack number as the mounted volume for use by open processing.

```
RMMV01D device REPLY WITH RACK NUMBER FOR NL REQUEST – OR REPLY "REJECT"
```

Environment

See *z/OS DFSMS Installation Exits* for information about setting up the exit.

Chapter 18. Using the Problem Determination Aid Facility

Perform this step once for each MVS image. You only need to perform this step if you want an external DASD record of trace data.

The problem determination aid (PDA) facility gathers DFSMSrmm processing information to enable analysis to pinpoint module flow and resource usage related to DFSMSrmm problems. The PDA facility is required for IBM service because it traces module and resource flow. The PDA facility consists of in-storage trace, optional DASD log data sets, EDGRMM*xx* parmlib member options, and operator commands to control tracing.

DFSMSrmm accumulates problem determination information at specific module points in the form of trace data, and it records this data in main storage. At predetermined intervals, the trace data is scheduled for output to DASD. The DFSMSrmm trace recording function receives the trace data scheduled for output and writes this data to a file on DASD. The PDA facility consists of two separate log data sets. DFSMSrmm recognizes these log data sets by their DD names, EDGPDOX and EDGPDOY. Recording takes place in the data set defined by EDGPDOX. When that data set is filled, the two data set names are swapped, and recording continues on the newly renamed data set.

When this data set is filled, the names are again swapped, and the output switches to the other data set, thus overlaying the previously recorded data. The larger the data sets, the longer the period of time that is represented by the accumulated data.

Establish a protocol that automatically copies the EDGPDOY data set to tape as a generation-data-group data set each time DFSMSrmm issues message EDG9117I. This practice provides a sequential history of trace data over time so that the data is available when needed for resolving problems.

DFSMSrmm trace data can be formatted with the DFSMShsm ARCPRPDO utility. You are authorized to use ARCPRPDO even if you are not licensed to use DFSMShsm. See *z/OS DFSMShsm Diagnosis Guide and Reference* for details about ARCPRPDO. *z/OS DFSMSrmm Diagnosis Guide* provides information about using the DFSMSrmm trace data to determine possible sources of errors. See *z/OS MVS System Messages, Vol 1 (ABA-AOM)* and *z/OS MVS System Messages, Vol 2 (ARC-ASA)* for the messages issued by the DFSMShsm ARCPRPDO utility.

Roadmap for Using the Problem Determination Aid

The following table shows the tasks and associated procedures for using the problem determination aid.

Task	Associated procedure
Plan to use the PDA Facility.	"Planning to Use the PDA Facility" on page 386
Determine how long to keep trace information.	"Determining How Long to Keep Trace Information" on page 386
Determine the size of the PDA log data set.	"Determining Problem Determination Aid (PDA) Log Data Set Size" on page 386

Task	Associated procedure
Enable the PDA facility.	“Enabling the Problem Determination Aid (PDA) Facility” on page 387
Allocate the PDA log data sets.	“Allocating the Problem Determination Aid (PDA) Log Data Sets” on page 387
Archive the PDA log data sets.	“Archiving the Problem Determination Aid (PDA) Log Data Sets” on page 388
Copy the PDA log data sets to tape.	“Copying the Problem Determination Aid (PDA) Log Data Sets to Tape” on page 388
Print the PDA log data sets.	“Printing the Problem Determination Aid (PDA) Log Data Sets” on page 388

Planning to Use the PDA Facility

Before you can use the PDA facility, you need to perform the following tasks:

1. Determine how long you want to keep trace information.
2. Optionally allocate storage on DASD for the PDA log data sets, EDGPDOX and EDGPDOY.
3. Implement the PDA facility based on how long you want to keep trace data.

Determining How Long to Keep Trace Information

How many hours, days, or weeks of trace history does your site want to keep? The minimum recommended trace history is four hours; however, a longer trace history gives a greater span both forward and backward in time. Your choice of a trace history interval falls into one of the following categories:

Short-Term Trace History

One to two days is typically considered a short-term trace history interval. Short-term trace histories can be obtained without using generation data groups (GDGs).

Long-Term Trace History

Two or more days is typically considered a long-term trace history interval. Long-term trace histories are best implemented with the use of generation data sets (GDSs) that are appended sequentially to form a generation data group (GDG).

A long-term trace history is preferred because some DFSMSrmm processing occurs only on a periodic basis. With a longer trace history, you might be able to see more patterns to help you perform problem analysis.

Determining Problem Determination Aid (PDA) Log Data Set Size

Allocate storage for your PDA log data sets based on the amount of trace data activity at your site and on how long you want to keep trace information. If you choose to keep trace information for two days or less, see Appendix F, “Problem Determination Aid Log Data Set Size Work Sheet for Short-Term Trace History,” on page 435. If you choose to keep trace information for longer than two days, see Appendix E, “Problem Determination Aid Log Data Set Size Work Sheet for Long-Term Trace History,” on page 433.

Enabling the Problem Determination Aid (PDA) Facility

The PDA facility default operating mode is trace enabled at DFSMSrmm startup.

You should continuously enable PDA tracing when DFSMSrmm is active since the processing overhead is minimal. If you define the EDGPDOX and EDGPDOY DD statements in the DFSMSrmm startup procedure, the EDGPDOX and EDGPDOY data sets are swapped and trace output is logged in the data sets.

You can control PDA tracing by using the following MVS MODIFY command keywords. You can also enable or disable the PDA facility by using the parmlib OPTION command PDA operands that are described in "Defining System Options: OPTION" on page 134.

PDA=ON|OFF Specify to turn PDA tracing on or off.

PDALOG=ON|OFF|SWAP
Specify to control the LOGGING function during PDA tracing.

Example: Use the MVS MODIFY command to enable PDA tracing.

```
F DFRMM,PDA=ON
```

The PDA log data sets are automatically swapped at DFSMSrmm startup. After startup, use the MVS MODIFY command PDALOG=SWAP to manually swap the data sets as required. For information on how to manually swap the PDA log data sets, see the *z/OS DFSMSrmm Diagnosis Guide* manual for details.

Allocating the Problem Determination Aid (PDA) Log Data Sets

Example: To allocate and catalog the problem determination log data sets:

```
/************************************************************************/
/* SAMPLE JOB THAT ALLOCATES AND CATALOGS THE PDA LOG DATA SETS.       */
/************************************************************************/
//ALLOPDO JOB MSGLEVEL=1
//STEP1   EXEC PGM=IEFBR14
//DD1     DD DSN=?UID..?HOSTID..RMMPDOX,DISP=(,CATLG),
//           UNIT=?TRACEUNIT.,
//           VOL=SER=?TRACEVOL.,SPACE=(CYL,(20))
//DD2     DD DSN=?UID..?HOSTID..RMMPDOY,DISP=(,CATLG),
//           UNIT=?TRACEUNIT.,
//           VOL=SER=?TRACEVOL.,SPACE=(CYL,(20))
```

Change the User ID (?UID.), MVS system image ID (?HOSTID.), the trace unit (?TRACEUNIT.), and the volume serial number (?TRACEVOL.) parameters to names that are valid for your environment. The LRECL and RECFM fields will be set by DFSMSrmm when the data set is opened and are not required in the JCL. Do not add the RLSE parameter to the DD statement.

The EDGPDOX and EDGPDOY data sets must be allocated to the same volume. The EDGPDOX and EDGPDOY data sets must be cataloged, variable blocked physical sequential and must not be striped. The data sets are required only if you want to keep a permanent history of the trace data on DASD. Begin with an initial data set size of 20 cylinders. You can adjust the size as you gain more experience with the amount of activity that is traced in your installation. If you allocate them as SMS-managed data sets they must be associated with a storage class having the GUARANTEED SPACE attribute. They should not be associated with a storage class that will conflict with the required data set attributes.

You must define a pair of trace output data sets for each MVS image. Do not share the trace data sets with multiple DFSMSrmm systems or with other system components.

Archiving the Problem Determination Aid (PDA) Log Data Sets

DFSMSrmm writes trace data to the data set name defined by the EDGPDOX DD statement. When DFSMSrmm swaps the output data sets, trace data recorded prior to the swap is available in the data set name defined by the EDGPDOY DD statement. If you want to archive trace data, you should copy the EDGPDOY data set at the time DFSMSrmm issues message EDG9117I.

Example: To define the generation data group (GDG) name for the archived problem determination output data set. The MVS system image ID (?HOSTID.) must be valid for your environment.

```
/**********************************************************************/
/* SAMPLE JOB THAT DEFINES THE GENERATION DATA GROUP NAME FOR THE      */
/* ARCHIVED PDA LOG DATA SET.                                           */
/**********************************************************************/
/*
//DEFGDG   JOB MSGLEVEL=1
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
          DEFINE   GDG(NAME('?UID..?HOSTID..RMMTRACE') -
          LIMIT(30) NOSCRATCH NOEMPTY)
/*
```

Copying the Problem Determination Aid (PDA) Log Data Sets to Tape

Example: The data set name for the DD statement, SYSUT1, must correspond to the name specified for the EDGPDOY data set. Change the MVS system image ID (?HOSTID.) to an ID that is valid for your environment. To copy the inactive trace data set to a scratch tape as a generation data set (GDS):

```
/**********************************************************************/
/* SAMPLE JOB THAT COPIES THE INACTIVE PDA LOG DATA SET TO TAPE        */
/**********************************************************************/
/*
//PDOCOPY     JOB MSGLEVEL=1
//STEP1       EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=?UID..?HOSTID..RMMPDOY,DISP=SHR
//SYSUT2   DD DSN=?UID..?HOSTID..RMMTRACE(+1),
//            UNIT=TAPE,
//            DISP=(NEW,CATLG,DELETE),
//            DCB=(?UID..?HOSTID..RMMPDOY)
```

Printing the Problem Determination Aid (PDA) Log Data Sets

For information about printing the problem determination aid logs, see *z/OS DFSMShsm Diagnosis Guide and Reference* for information about the ARCPRPDO utility.

Chapter 19. Setting Up DFSMSrmm Disposition Processing

DFSMSrmm disposition processing is optional and provides support for the following functions:

- Providing operators with information to help them perform tasks like moving a tape to a specific location
- Generating sticky labels
- Updating the location where a volume resides.

You can request DFSMSrmm disposition processing by identifying a disposition control data set using the DFSMSrmm parmlib EDGRMM*xx* OPTION DISPDDNAME operand and coding the same DD name in a job step that creates or references a tape data set. Figure 170 on page 392 shows how you might code the DD name in a job step. DFSMSrmm disposition processing occurs at CLOSE and EOV for each volume. DFSMSrmm does not provide support for asynchronous processing in a separate job step.

If you use the input disposition control file and, optionally, customize via the DFSMSrmm EDGUX100 installation exit, DFSMSrmm disposition processing can be expanded to support:

- Messages that are issued to one or more route codes by defining multiple messages in the disposition control file.
- Suppression of label output you might have set up using the control file. Modify label output by using the DFSMSrmm EDGUX100 installation exit.
- Assignment of loan location, storage location, or library location for a volume.

Implementing DFSMSrmm Disposition Control File Processing

Follow these steps to implement DFSMSrmm disposition processing.

1. Define disposition control information in a disposition control file as described in “Modifying the Contents of the Disposition Control File” on page 390. You can define separate disposition control files for individual users, a separate job, or a separate job step. Define the data sets with LRECL 80. The data sets can be sequential files or can be members of a partitioned data set. You can also include data in your JCL stream instead of defining a data set.
2. Add the disposition control file DD name to the batch job step of the tape jobs that process the tape files for which messages or labels or location assignment is required. The DD name is the same name as you specified in DISPDDNAME in parmlib.
3. Define the disposition control file DD name in the DFSMSrmm EDGRMM*xx* parmlib member by using the OPTION command DISPDDNAME operand as described in “Defining System Options: OPTION” on page 134. You can also define a message ID in parmlib by using the OPTION command DISPMSGID operand. DFSMSrmm builds a WTO message by using the message number from the DISPMSGID parmlib option and the message text defined in the disposition control file.
4. If you do not add the OUTPUT JCL statement, sticky labels are produced using a WTO on route code 13. Add the OUTPUT JCL statement to the DFRMM started procedure as shown in “Step 8: Updating the Procedure Library” on page 31 if you want DFSMSrmm to dynamically allocate a SYSOUT file for each label. Use the attributes of the OUTPUT statement to define how the label output should be printed.

5. Document the procedures you use for printing label output and applying sticky labels to the correct volumes. If you are using DFSMSrmm disposition control for the first time, work with your operations staff to develop, test, and document procedures for responding to DFSMSrmm messages, printing labels, and using the DFSMSrmm EDGUX100 exit to modify output. See *z/OS DFSMSrmm Guide and Reference* for information about procedures for your operators.

Tip: DFSMSrmm disposition processing can fail when your application closes a DASD data set and a tape data set at the same time. The system issues error code 50D under these conditions.

- When the DASD data set was allocated with RLSE for the SPACE parameter in the DD statement or RELEASE in the TSO ALLOCATE command,
- When the SMS management class specifies YI or CI for the partial release attribute.

Partial release processing during CLOSE holds an exclusive enqueue on the task input/output table (SYSZTIOT) resource and can prevent DFSMSrmm from opening the disposition control file at the same time. To ensure that the disposition control file can be opened, code a separate CLOSE for every data set or remove the RLSE parameter from the DD statement for the DASD data set.

Modifying the Contents of the Disposition Control File

You identify the input disposition control file by using DFSMSrmm EDGRMM*xx* parmlib DISPDDNAME operand. The disposition control file is a fixed format sequential file with a record length of 80 characters. All the parameters including the message text are position dependent. Figure 169 shows the layout for the disposition control file.

```
ddname_1nrlOUT=vvvv,message text
ddname_1nrlmessage text
ddname_2nrlLOC=vvvvvvvv,message text
ddname_3nrlLOC=vvvvvvv,message text
```

Figure 169. Disposition Control File Record Format

ddname
1-to-8 character ddname of an input file or output file. This *ddname* must match the ddname you defined for the input tape file or the output tape file processed by the current job step. DFSMSrmm only processes the statements that match to the DD name of the file being processed by CLOSE or EOV.

You can code multiple lines for each ddname specified and can include as many different ddnames as is required to cover tape data sets requiring disposition support. As each file is processed by CLOSE or EOV, DFSMSrmm uses only those statements that match the ddname of that file. *ddname* must be coded in position 1 through 8.

n Code any non-blank character in position 9 to cause the message to be issued in a non-roll deletable message to the console.

r Code a character to define the route code to use for the WTO. Use one of the characters shown in Table 55 on page 391 to use specific route codes. Use any character other than the characters in Table 55 on page 391 to use the default route code 13.

Table 55. Setting the Message Route Code

If You Code	Then You Are Requesting This Route Code
Any	13 (default value). Any value other than those below results in the default value being used.
A	2
B	3
C	5
*	11
F	2,11
G	3,11
H	5,11

I Indicates if a sticky label is to be produced as part of disposition processing and must be coded in position 11. DFSMSrmm ignores any other value and no label is produced. The valid text codes are shown in Table 56.

Table 56. Coding Sticky Label Text

If You Code	Then You Are Requesting That
L	a label is produced by disposition processing. You can code multiple L's with the same ddname and route code if more message text is required.
M	a label is produced by disposition processing but the message text is passed as user data to the DFSMSrmm EDGUX100 installation exit for further processing.

OUT=*vvvvvvvv*

This is an optional keyword starting in position 12. Use OUT to change the location for a volume. When you use OUT to change a volume's location, you do not need to confirm the volume move. When OUT is specified, OUT must be followed by a comma, if you also specified message text. *vvvvvvvv* is a 1 to 8 character location name, that is to be used by DFSMSrmm as a location name for the current volume.

LOC=*vvvvvvvv*

This is an optional keyword. If you specify LOC, it must be followed by a comma if the optional message text is specified. *vvvvvvvv* is a 1 to 8 character location name. DFSMSrmm uses the value as a location name for the current volume. When you use LOC=, you request that DFSMSrmm update the volume's' destination so that the move can be tracked and confirmed later.

LOC and OUT are mutually exclusive. If you specify multiple statements for the same DD name, DFSMSrmm uses the last location name that you specify for the volume.

You can optionally blank pad location names on the right up to the maximum length of a location name.

DFSMSrmm checks the location name you specify against the location names you defined using the DFSMSrmm LOCDEF location information defined to DFSMSrmm at startup time. If the location you specified, is not defined using LOCDEF DFSMSrmm treats the location as if it is a loan location. If the location you specified is identified as a storage location, use OUT= if you do not want to confirm the move. Use LOC= if you want the move confirmed at a later time.

You can use EDGUX100 to control how the location name is used as described in “Changing Location Information with EDGUX100” on page 234. You can override the location type determined from the LOCDEF location information to specify that a location is a loan location. You can control whether a confirm move is required for DFSMSrmm storage locations.

When you specify a location name, DFSMSrmm uses this information just as if you had entered the value on an RMM CHANGEVOLUME subcommand with either LOANLOC or LOCATION. If the location is a bin-managed storage location, the required location is set to this value and inventory management DSTORE processing assigns a bin number. You can override any location assignment by defining vital record specification movement policies.

message_text
This is up to 69 characters of message text to be issued as a WTO. It must begin in position 12 or after the comma which separates the message text from the location name. If additional text is required, include another record in the control file specifying the same ddname and route code. Each message is issued as a separate WTO.

If the *l* value is M, the message text is limited to up to 69 characters of text and is treated as user data for label processing. No additional ddname records are supported for the M user data option.

Figure 170 shows the sample JCL, if you want to perform the following tasks:

- Issue a WTO on route code 2 and route code 1
- Generate a sticky label
- Set the volume location to a location called FICH
- Confirm that the volume is moved to the location FICH. In Figure 170, when IEBGENER closes the SYSUT2 output file, DFSMSrmm scans the DISPDD file for the SYSUT2 DD statement, if the DISPDD file is defined. Figure 170 shows two entries for SYSUT2 which is the file being closed. The first entry requests that DFSMSrmm issue the message ’SEND THIS TAPE TO THE FICHE PRINTER’ on route codes 2 and 11, that a sticky label is generated, and that the volume’s move to location FICH is already confirmed. The second entry provides user data that is passed to the DFSMSrmm EDGUX100 installation exit. DFSMSrmm includes the user data in the sticky label it generates as described in “Creating Sticky Labels” on page 231

```
// EXEC PGM=IEBGENER
//SYSUT2 DD DISP=(,KEEP),DSN=MY.FICHE.DATA,UNIT=TAPE,LABEL=(,SL)
//SYSUT1 DD DISP=SHR,DSN=MASTER.FICHE.DATA
//SYSIN  DD DUMMY
//DISPDD DD *
SYSUT2   FLOUT=FICH,SEND THIS TAPE TO THE FICHE PRINTER
SYSUT2    MUSER DATA TO BE SENT TO EDGUX100
/*
```

Figure 170. Sample JCL to Request Disposition Processing

The message DFSMSrmm issues is EDG4054I, but you can use the DISPMSGID parmlib option to change the message number to an installation selected value.

Selecting the Method Used for Label Processing

You can request that DFSMSrmm produce labels using a WTO on route code 13 or using the OUTPUT JCL statement to send labels to a spool file or a printer. You control the method DFSMSrmm uses by specifying an OUTPUT JCL statement in the DFSMSrmm started procedure as described in “Step 8: Updating the Procedure Library” on page 31. The name on the OUTPUT JCL statement must exactly match that specified for the DD name of the disposition control file.

If you use the OUTPUT JCL statement method, DFSMSrmm dynamically allocates a sysout file for each label using the DISPDDNAME OUTPUT JCL statement. You use the attributes of the OUTPUT statement to define how the label output is to be printed. For example, you can route the output to another system, specify a special forms type or use any of the OUTPUT statement keywords.

If you do not provide an OUTPUT JCL statement in the DFSMSrmm started procedure, you must configure a console to accept WTO messages on route code 13 so that the labels can be printed.

Modifying Tape Labels

The default label is 10 rows with a maximum of 80 characters per row. The default LRECL is set to 80. The maximum number of data characters supported in a label is 2000 characters. DFSMSrmm provides two default label styles for your use. You can modify these label styles using the DFSMSrmm EDGUX100 installation exit. Figure 171 shows the default label for cartridges. The default label consists of eight data rows, two of which are used for spacing. Cartridge labels are identified by media type other than *, and a density of either *, IDRC, or 3480. Figure 172 shows the default label for all other types of volumes. The default label consists of seven data rows, 3 of which are used for spacing the labels.You can use the DFSMSrmm installation exit EDGUX100 to update the default labels as described in “Modifying DFSMSrmm Label Output” on page 232.

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7.
dsname______________________________________
userdata______________________________________________________________

 jobname_     crdate____
              expdt_____
 dens comp lrecl  blksiz recf

 volser seqn lab       devc
```

Figure 171. Default Label Format for a Tape Cartridge

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7.
dsname______________________________________
userdata______________________________________________________________
    jobname_                  crdate____

 dens comp lrecl  blksiz recf expdt_____

               volser seqn lab      devc
```

Figure 172. Default Label Format for a Round Tape

The values for the variables shown in Figure 171 on page 393 and Figure 172 on page 393 are:

dsname	The data set name of the file being processed. 1 to 44 characters.
userdata	The user data specified via message text in the disposition control file. 0 to 69 characters.
jobname	The current job name. 1 to 8 characters.
crdate	The data set create date. 1 to 10 characters in the date format specified by the DATEFORM parmlib option.
expdate	The data set expiration date. 1 to 10 characters in the date format specified by the DATEFORM parmlib option.
dens	The recording density of the volume. 1 to 4 characters.
comp	Indication of compaction of data on the volume. 4 characters.
lrecl	The logical record length of the data. 1 to 5 characters.
blksiz	The block size of the data. 1 to 6 characters.
recf	The record format. 1 to 4 characters.
volser	The volume serial number. 1 to 6 characters.
seqn	The volume sequence number. 1 to 4 characters.
lab	The volume label type. 1 to 3 characters.
devc	The number of the drive on which the file is processed. 4 characters.

Use the EDGSLAB macro to map the label data area as described in “Sticky Label Data: EDGSLAB” on page 422.

Chapter 20. Running DFSMSrmm with the IBM Tivoli Workload Scheduler for z/OS

DFSMSrmm provides sample jobs and procedures that you can modify to set up automation software, such as the IBM Tivoli Workload Scheduler for z/OS, to run DFSMSrmm tasks. The sample jobs and procedures are shipped in SAMPLIB.

Recommendation: Before you use the DFSMSrmm-supplied jobs, customize the jobs for your installation. You must make the job control language (JCL) available to the IBM Tivoli Workload Scheduler for z/OS for submission. The procedures must be in a procedure library that you use with jobs that are submitted by the IBM Tivoli Workload Scheduler for z/OS.

The naming conventions are EDGJ*xxxx* for IBM Tivoli Workload Scheduler for z/OS jobs and EDGP*xxxx* for IBM Tivoli Workload Scheduler for z/OS procedures. Each EDGJ*xxxx* job contains JCL SET statements for variables that are required for correct JCL generation. For example, you must set the variable RMMPREF to the data set name prefix for use on all output data sets. You can tailor the jobs so you can use the jobs for normal job submission or for recovery or restart. Each EDGP*xxxx* procedure can be tailored as well. You might have to customize space requirements for the files that are allocated for use as output files.

The supplied jobs support DFSMSrmm tasks that are performed daily, weekly, and monthly and fall into these categories:

1. Scheduled tasks that are run on a regularly scheduled basis.
2. Event triggered tasks that are run on as-needed basis. See “Event Triggered Tracking” on page 401 for descriptions of DFSMSrmm event triggered tasks.

Using a Tivoli Special Resource When Running DFSMSrmm with the IBM Tivoli Workload Scheduler for z/OS

Use an IBM Tivoli Workload Scheduler for z/OS special resource to perform the following functions:

- Allow some DFSMSrmm jobs to run in parallel.
- Prevent more than one EDGHSKP job from running at the same time.
- Avoid jobs failing because inventory management is already running.
- Prevent the EDGUTIL VERIFY job from running at the same time that other DFSMSrmm inventory management jobs are running.
- Allow the IBM Tivoli Workload Scheduler for z/OS to handle the dependencies for jobs that do not always need to run.

The special resource is named in the DFSMSrmm sample IBM Tivoli Workload Scheduler for z/OS loader job EDGJLOPC. The special resource is defined when you load the DFSMSrmm application to the IBM Tivoli Workload Scheduler for z/OS. You can customize the name of the special resource in EDGJLOPC. Be sure to change all occurrences of the special resource name in the application definition, so that the IBM Tivoli Workload Scheduler for z/OS job scheduling can maintain inventory management serialization properly.

Most of the DFSMSrmm tasks in the job flow are automated. These tasks require manual intervention.

1. Confirmation of moves requires intervention at an IBM Tivoli Workload Scheduler for z/OS terminal to mark volume moves completed. You must confirm volume movement before you can run the confirm job.
2. During restart or recovery of the DFSMSrmm-supplied Main job, you might have to perform visual checking and some other recovery actions based on the reason for the failure. For example, when recovery completed, the Main job restarts from the beginning. For EDGHSKP, return code 4 is an acceptable completion code. Any higher return code value triggers restart or recovery processing.

DFSMSrmm provides a sample procedure called EDGJLOPC. The sample procedure includes code that supports the inventory management schedule that is recommended in “Scheduling DFSMSrmm Utilities” on page 275.

Setting Up DFSMSrmm to Use the IBM Tivoli Workload Scheduler for z/OS

Follow this procedure to set up IBM Tivoli Workload Scheduler for z/OS for management of DFSMSrmm tasks:

1. Customize the batch loader statements in EDGJLOPC as described in “Customizing the IBM Tivoli Workload Scheduler for z/OS Batch Loader Statements” on page 401.
2. Ensure that the IBM Tivoli Workload Scheduler for z/OS workstations that are required by the DFSMSrmm applications are defined to the IBM Tivoli Workload Scheduler for z/OS as described in “Setting Up IBM Tivoli Workload Scheduler for z/OS Workstations” on page 401.
3. Make the customized jobs and procedures available to IBM Tivoli Workload Scheduler for z/OS and to your running systems as described in “Descriptions of DFSMSrmm Jobs to Run with the IBM Tivoli Workload Scheduler for z/OS” on page 397.
4. Run EDGJHKPA to define GDG bases and to create first generations if needed. DFSMSrmm provides a job called preparation, that is described in “Descriptions of DFSMSrmm Jobs to Run with the IBM Tivoli Workload Scheduler for z/OS” on page 397, that you can use to create the GDGs.
5. Change the DFSMSrmm PARMLIB OPTION command BACKUPPROC and SCRATCHPROC operands to use the new sample procedures for use with the IBM Tivoli Workload Scheduler for z/OS as described in “Defining System Options: OPTION” on page 134.
6. Run the DFSMSrmm sample EDGJLOPC job as described in “Descriptions of DFSMSrmm Jobs to Run with the IBM Tivoli Workload Scheduler for z/OS” on page 397.
7. Add the event trigger tracking entries to the IBM Tivoli Workload Scheduler for z/OS by using the dialog as described in “Event Triggered Tracking” on page 401.
8. Set up the IBM Tivoli Workload Scheduler for z/OS restart management to handle restart activities. Restart management uses the DFSMSrmm application programming interface to issue commands based on the IBM Tivoli Workload Scheduler for z/OS restart/recovery options that you define. If you do not use IBM Tivoli Workload Scheduler for z/OS, you can issue DFSMSrmm TSO subcommands to change information as needed.

If not all of the data sets on a volume are created successfully, you can use the subcommand shown in Figure 173 to override normal vital record specification management processing.

```
RMM CHANGEDATASET dsname VOLUME(volser) SEQ(number) ABEND
```

Figure 173. Overriding Vital Record Specification Management Processing with the RMM CHANGEDATASET TSO Subcommand

Descriptions of DFSMSrmm Jobs to Run with the IBM Tivoli Workload Scheduler for z/OS

All the sample jobs that are used with the IBM Tivoli Workload Scheduler for z/OS job management include the required control statements about recovery, restart, or modifications to make at submit time. The JCL uses procedures that are provided to avoid duplicate JCL and to simplify job construction.

Preparation
You must first allocate files that are required for the regular processing. EDGHSKP has special requirements for the data sets that are used for processing. Use EDGJHKPA to define the GDGs and set up the first generation of the files as required. EDGJHKPA uses the procedures EDGPHKPA, EDGPVRSA, EDGPMSGA, EDGPACTA and EDGPRPTA, and REXX procedure EDGRHKPA. Specify parameters in EDGHKPA to customize the GDG names and limits. Customize the file size by altering the SPACE values in the lowest level procedures.

CDS Verify
Run this job when you want to verify the contents of the DFSMSrmm control data set. The EDGJVFY job runs EDGUTIL with PARM=VERIFY on the current DFSMSrmm control data set. The expected completion is return code 0. You must perform manual recovery before dependent jobs can be run when the return code is greater than 0.

CDS Backup
Run this job when you want to back up the DFSMSrmm control data set. The EDGJBKP1 and EDGBKP2 jobs run EDGHSKP with PARM=BACKUP. Backup is performed at the start and at the end of the main inventory management application processing. You can modify the jobs to use DFSMSdss or AMS for backup. You make the choice of backup to DASD or tape when running the preparation jobs.

Jobs EDGJBKP1 and EDGJBKP2 run procedure EDGPBKP to perform the backup. Expected completion is return code 4 or less. You must perform manual recovery before dependent jobs can be run when the return code is greater than 4.

Inventory Management
Run this job when you want to perform DFSMSrmm inventory management processing. These jobs run EDGHSKP with VRSEL and EXPROC. These jobs also include running storage location management on a weekly basis to trigger movement decisions for off-site storage. Job EDGJDHKP and job EDGJWHKP run procedures EDGPHSKP, EDGPMSGA, and EDGPVRSA to process and set up the files for the next run. The job copies the message file to SYSOUT for easier information analysis. The expected completion is return code 4 or less. You must perform manual recovery before dependent jobs can be run when the return code is greater than 4. As part of the recovery, run the EDGJVRSV job to run VRSEL with VERIFY and to report on the ACTIVITY file for analysis of VRSMIN and VRSCHANGE conditions. For return code 8, you might have to

change vital record specifications or add additional empty bin numbers. Higher return codes or abends need to be researched by the support programmer.

Erasing and Labeling Volumes
Run this job when you want to initialize and erase volumes depending on the release actions set for the volumes. The EDGJINER job runs an automatic EDGINERS job to handle the INIT and ERASE release actions. You can run dependent jobs that are run regardless of successful completion.

Scratch Processing
Run this job when you want to process volumes that are returning to scratch status. The EDGJSCR job runs EDGHSKP with the EXPROC parameter. This job handles the return to scratch of volumes requiring confirmed actions; whether INIT/ERASE or movement actions. You can run dependent jobs regardless of successful completion.

Scratch Reporting
Run this job when you want to obtain information about volumes in scratch status. The EDGJSCR job produces an up-to-date extract and uses EDGRPTD with SCRLIST and NEWSCR files to produce the latest scratch list reports. You can run dependent jobs regardless of successful completion.

Ejecting Volumes
Run this job to eject volumes from system-managed tape libraries. This is an optional job that you can use if your installation uses system-managed tape. The job also includes support for VTS export processing. You can run dependent jobs regardless of successful completion. You can customize this job to include similar processing that is required for any non-system-managed volumes that you have in your library.

Producing Reports
Run this job to produce volume movement reports. The EDGJMOVE job is dependent on the inventory management job and optionally the ejecting volume job processing. This job creates an up-to-date extract and uses EDGRPTD to produce the movement reports. You can run dependent jobs regardless of successful completion.

Move Confirmation
Run this job to perform global confirmation of volume movement. The EDGJCMOV job issues a global confirm move. You can run this job after a manual action is taken by the installation to mark the movement complete at the dependent IBM Tivoli Workload Scheduler for z/OS workstation. You can run dependent jobs regardless of successful completion.

Daily jobs are always run on processing days. Weekly and monthly jobs are included in the job flow and the IBM Tivoli Workload Scheduler for z/OS automatically adjusts dependencies that are based on the rules you specify. When a job fails, the IBM Tivoli Workload Scheduler for z/OS automatically prevents dependent jobs from processing if any manual recovery actions have been defined. To alter the dependencies, you can modify the sample IBM Tivoli Workload Scheduler for z/OS applications that are described in “IBM Tivoli Workload Scheduler for z/OS Applications for DFSMSrmm” on page 399.

CDS backup and scratch processing are functions that are always run in the daily cycle. CDS backup and scratch processing can also be added dynamically to the cycle as required. When CDS backup and scratch processing are added dynamically, they run independently. They are added automatically to the schedule when the write-to-operator (WTO) from DFSMSrmm is detected and the BACKUPPROC or SCRATCHPROC is triggered. When a started procedure matches an application defined to the IBM Tivoli Workload Scheduler for z/OS running on a started task processor, IBM Tivoli Workload Scheduler for z/OS modifies the current plan.

IBM Tivoli Workload Scheduler for z/OS Applications for DFSMSrmm

The application definition implements the schedule suggested in the “Scheduling DFSMSrmm Utilities” on page 275. In addition, EDGJLOPC includes support for events that are triggered by alerts such as the CBR3660A, short on scratch message, and the EDG2107E, journal threshold reached message.

The applications defined in the DFSMSrmm sample IBM Tivoli Workload Scheduler for z/OS setup job are shown in Table 57 on page 400:

Table 57. IBM Tivoli Workload Scheduler for z/OS Applications

Application	Description
GRMMDAY	GRMMDAY is a grouping application used to set a common start time for daily jobs and applications. GRMMDAY is scheduled to run every working day of the year.
GRMMMTH	GRMMMTH is a grouping application used to set a common start time for monthly jobs and applications. GRMMMTH is scheduled to run once each month on the first Friday of the month.
GRMMWK	GRMMWK is a grouping application used to set a common start time for weekly jobs and applications. GRMMWK is scheduled to run every Monday.
RMMBKP	RMMBKP is the first daily application. RMMBKP backs up the control data set and journal data set, and clears the journal. RMMBKP is dependent on the monthly RMMMTH job EDGJVFY.
RMMEXP	RMMEXP runs expiration processing and produces scratch list reports. RMMEXP is dependent on prior daily jobs and the weekly movement processing performed by the RMMMOVES job.
RMMHKPD	RMMHKPD is the main daily inventory management run. RMMHKPD is replaced once a week with a weekly inventory management run that includes movement processing.
RMMMOVES	RMMMOVES includes jobs to: • Eject system-managed volumes that are moving • Produce and print movement reports • Confirm volume moves once volumes have been moved RMMMOVES includes a manual action to complete a task at a workstation when volumes have been moved. RMMEXP is dependent on the completion of RMMMOVES.
RMMMTH	RMMMTH performs a monthly verify of the DFSMSrmm control data set. When you run RMMMTH, all other applications are dependent on RMMMTH running successfully.
RMMPOST	RMMPOST backs up the control data set and journal data set, and clears the journal. RMMPOST runs EDGINERS to process any new release actions.
RMMWK	The main inventory management run once each week instead of the daily inventory management application. RMMWK is dependent on RMMBKP.
RMMVRSVER	RMMVRSVER is an application that is set up only for use during recovery of the main inventory management application. RMMVRSVER is dynamically added if the EDGHSKP step fails with return code 8. RMMVRSVER runs VRSEL with VERIFY and produces a report from the ACTIVITY file.

You can use the sample job EDGJLOPC to run the IBM Tivoli Workload Scheduler for z/OS batch loader utility to define DFSMSrmm as an application to IBM Tivoli Workload Scheduler for z/OS to manage the regular scheduling of DFSMSrmm functions.

Customize the EDGJLOPC sample by:

- Modifying the supplied JCL to run in your environment
- Tailoring and customizing the sample application definition and schedule to meet your specific needs.

Then run the sample to load the application definition to the active control file or to a VSAM file to be used with the IBM Tivoli Workload Scheduler for z/OS subsystem.

Customizing the IBM Tivoli Workload Scheduler for z/OS Batch Loader Statements

The DFSMSrmm sample job EDGJLOPC contains the application definitions for the applications and jobs described in “IBM Tivoli Workload Scheduler for z/OS Applications for DFSMSrmm” on page 399. You can alter or add to the definitions before running the IBM Tivoli Workload Scheduler for z/OS batch loader to load the definitions to your IBM Tivoli Workload Scheduler for z/OS subsystem or to another VSAM file you plan to use as an IBM Tivoli Workload Scheduler for z/OS AD database.

Setting Up IBM Tivoli Workload Scheduler for z/OS Workstations

You should ensure that you define these workstations to your IBM Tivoli Workload Scheduler for z/OS subsystem or alter them to those you want to use with the DFSMSrmm applications.

The sample definitions use the following workstations:

STC1	A computer workstation for the running started tasks
CPU1	A computer workstation for the running batch jobs
PRT1	A workstation for printing movement reports
TLIB	A manual workstation for use by the tape librarian or operator to mark movement of volumes completed.

Event Triggered Tracking

The applications are set up to take advantage of the IBM Tivoli Workload Scheduler for z/OS event trigger tracking for these conditions:

- Journal threshold is reached - backup is required to clear the journal
- Low on scratch - expiration processing is required to return pending release volumes to scratch status and produce new scratch lists.

To use the applications, follow these steps:

1. Define these tasks to the IBM Tivoli Workload Scheduler for z/OS under Event Trigger Tracking; EDGSETT triggers RMMEXP, and EDGBETT triggers RMMBKP.
2. Change the DFSMSrmm parmlib OPTION command BACKUPPROC and SCRATCHPROC operands to name special procedures EDGBETT and EDGSETT. Specify EDGBETT and EDGSETT to enable the event trigger processing.

   The options for each entry in the ETT table are:

   - Event type=J
   - Job replace=Y
   - Dependency resolution=N
   - Availability status=N

   EDGBETT and EDGSETT sample procedures are only intended for tracking by IBM Tivoli Workload Scheduler for z/OS because they only execute IEFBR14.

Appendix A. DFSMSrmm Installation Verification Procedures

DFSMSrmm Samples Provided in SAMPLIB

- EDGIVPPM Sample Parmlib for Use in the IVP
- EDGIVP1 IVP Job 1 to Initialize Tape Volumes
- EDGIVP2 IVP Job 2 to Use Tape Volumes

This section helps you prepare for and run the DFSMSrmm installation verification procedures (IVP). You can use the IVP to ensure that the DFSMSrmm functional component has been successfully installed by SMP/E and can be activated on your system. The IVP does not test all the functions in DFSMSrmm but validates that the key interfaces are in place.

Preparing to Run the IVP

Before you run the IVP, you need to activate some of DFSMSrmm's functions. This section lists and describes the steps you should perform to set up DFSMSrmm for the IVP. Chapter 2, “Implementing DFSMSrmm” contains all the steps needed to install DFSMSrmm.

If this is first time you are setting up DFSMSrmm, follow all the steps described in this section. If your system has previously been set up for use with DFSMSrmm, you might not need to perform all the steps listed here. Evaluate your installation setup to determine which steps you can omit.

1. Install DFSMSrmm with SMP/E.

   Ensure that DFSMS/MVS including DFSMSrmm is SMP/E applied. Refer to *z/OS Program Directory* that you received with the product tape or to *ServerPac: Installing Your Order* for complete installation instructions.

   Once you have used SMP/E to install DFSMSrmm, IPL your system without performing any implementation tasks and have DFSMSrmm take no part in removable media management. The ability to run without DFSMSrmm is especially helpful if you are running another tape management product in production.
2. Update SYS1.PARMLIB members.

   Refer to “Step 5: Updating SYS1.PARMLIB Members” on page 25 for detailed instructions. At a minimum, you should update IEFSSN*xx* and IKJTSO*xx*. Also update IFAPRD*xx*.
3. Update the procedure library.

   Refer to “Step 8: Updating the Procedure Library” on page 31 for detailed instructions. Use member EDGDFRMM of SYS1.SAMPLIB as a sample DFSMSrmm procedure.
4. Assign DFSMSrmm a RACF user ID.

   Perform this step if you want to use a specific RACF user ID for DFSMSrmm during the IVP. When running on a system with RACF installed, assign DFSMSrmm a RACF user ID by adding a profile in the STARTED class as described in “Step 9: Assigning DFSMSrmm a RACF User ID” on page 34. You can use the DFSMSrmm procedure name that you created in step 3 as the RACF user ID but any installation-selected RACF user ID is acceptable. As data sets are created for use by the DFSMSrmm procedure, add the RACF user ID to the access list for the data sets. Table 6 on page 34 lists the data sets that the DFSMSrmm procedure should be able to access.

If you are using an equivalent security product, review the RACF-related information to determine the changes that might be required to run DFSMSrmm with that product.

5. Define parmlib member EDGRMM*xx*.

   Refer to "Step 10: Defining Parmlib Member EDGRMMxx" on page 35 for detailed instructions. Use member EDGIVPPM of SYS1.SAMPLIB as a sample parmlib member.

6. Specify DFSMSrmm options.

   Refer to Chapter 8, "Using the Parmlib Member EDGRMMxx," on page 127 for information on tailoring the DFSMSrmm sample parmlib member EDGIVPPM to specify DFSMSrmm options for the IVP.

   During the IVP, DFSMSrmm runs in record-only mode. DFSMSrmm records information about tape volumes, but does no validation. You can tailor EDGIVPPM to specify that DFSMSrmm run in warning mode or protect mode if you want DFSMSrmm to validate volumes.

   If you are running the IVP on a system with no other tape management system you can select any mode: record-only, warning or protect. If there is a possibility of accidental use of the wrong tape volumes, we suggest that you run in protect mode. However, if you run the IVP on a system where others are using tape including the use of scratch tapes, be aware that DFSMSrmm rejects all scratch tapes not defined to it while running in protect mode. See "Defining System Options: OPTION" on page 134 for information about DFSMSrmm modes of operation.

7. Create the DFSMSrmm control data set.

   Refer to "Step 12: Creating the DFSMSrmm Control Data Set" on page 36 for detailed instructions. You can use the sample JCL in member EDGJMFAL in SYS1.SAMPLIB to allocate a control data set. Ensure that the control data set name is the same as that specified in the parmlib member EDGRMMxx that you created earlier. Initialize the control data set by running the EDGUTIL utility. You can use the sample JCL in member EDGJUTIL in SYS1.SAMPLIB. Set the rack and bin count fields to 0.

8. Create the journal.

   Refer to "Step 13: Creating the Journal" on page 40 for detailed instructions. You can use sample JCL in member EDGJNLAL in SYS1.SAMPLIB to allocate a journal data set.

9. Make the DFSMSrmm ISPF Dialog available to users.

   Refer to "Step 15: Making the DFSMSrmm ISPF Dialog Available to Users" on page 43 for instructions to make the DFSMSrmm dialog available to users.

10. Restart MVS with DFSMSrmm implemented.

    You are ready to start the system with DFSMSrmm implemented. Refer to "Step 16: Restarting MVS with DFSMSrmm Implemented" on page 46 for information on conditions that determine if you need to IPL the system to restart MVS with DFSMSrmm implemented. Perform this step so that the changes you made to IEFSSNxx and other parmlib members when you performed step 2 on page 403 take effect.

11. Start DFSMSrmm.

    Refer to "Step 18: Starting DFSMSrmm" on page 47 for detailed instructions. When you start DFSMSrmm, if it issues message EDG0103D, reply 'RETRY'. If you do not reply 'RETRY', DFSMSrmm will not record any tape usage activity.

DFSMSrmm is activated and you are ready to run the IVP.

Running the IVP

To run the IVP, perform the following steps:

1. You need three tape volumes that do not have any data on them, and a single tape unit online to your system. Ask your tape librarian to externally label these volumes EDG000, EDG001, and EDG002 for your testing.

   Ensure that the tape volumes you use are suitable for use with DFSMSrmm during the IVP. For example, if you have an existing tape management system, check that the volumes are either not managed by it or are designated for use with DFSMSrmm for testing.
2. Ensure that TSO help information has been correctly installed by entering the following command from a TSO terminal:

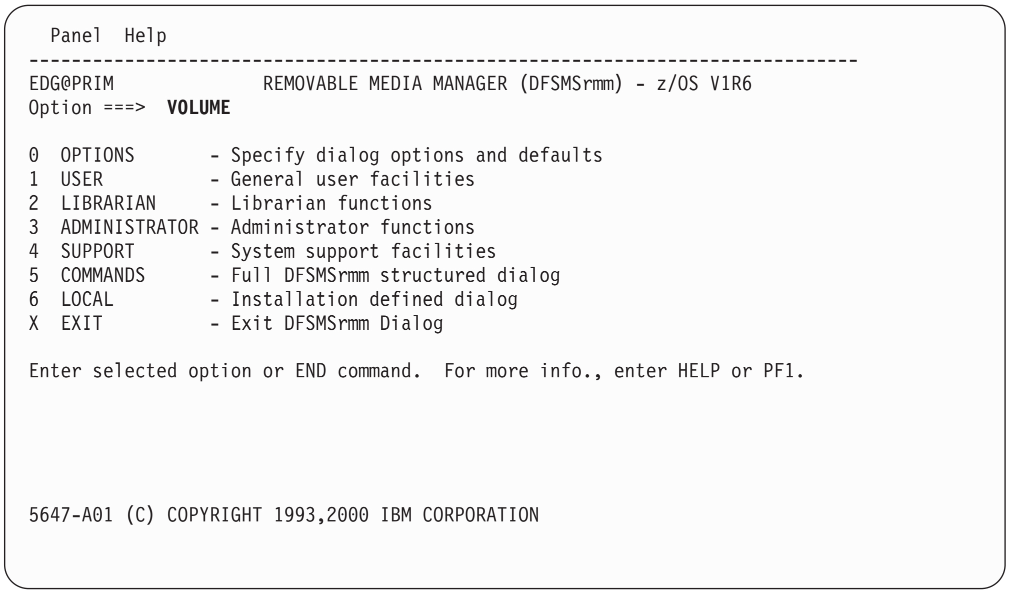
   ```
   HELP RMM
   ```

   DFSMSrmm lists help information for the RMM TSO subcommand, including a list of subcommands, function, syntax, and operands.
3. Add some shelf locations to DFSMSrmm by entering the following RMM TSO command from a TSO terminal:

   ```
   RMM ADDRACK RMM000 COUNT(10)
   ```
4. Add some volumes to DFSMSrmm by using the DFSMSrmm ISPF dialog. Enter the following command from a TSO terminal:

   ```
   %RMMISPF
   ```

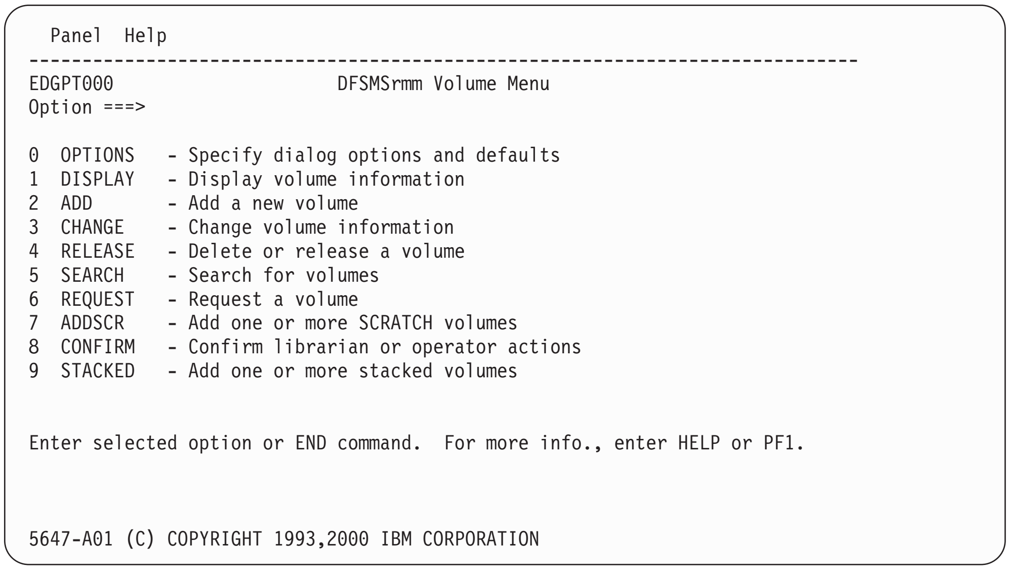
   DFSMSrmm displays the DFSMSrmm ISPF dialog primary option menu as shown in Figure 174.



```
  Panel  Help
------------------------------------------------------------------------------
EDG@PRIM              REMOVABLE MEDIA MANAGER (DFSMSrmm) - z/OS V1R6
Option ===>  VOLUME

0  OPTIONS       - Specify dialog options and defaults
1  USER          - General user facilities
2  LIBRARIAN     - Librarian functions
3  ADMINISTRATOR - Administrator functions
4  SUPPORT       - System support facilities
5  COMMANDS      - Full DFSMSrmm structured dialog
6  LOCAL         - Installation defined dialog
X  EXIT          - Exit DFSMSrmm Dialog

Enter selected option or END command.  For more info., enter HELP or PF1.




5647-A01 (C) COPYRIGHT 1993,2000 IBM CORPORATION
```

Figure 174. DFSMSrmm Primary Option Menu

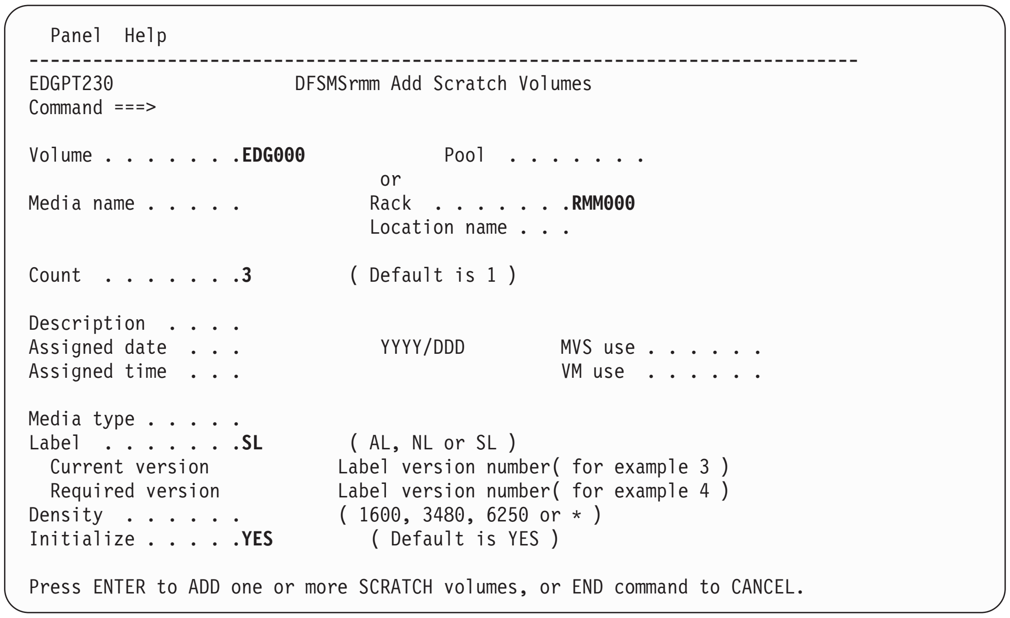
   Enter VOLUME on the option line to display the DFSMSrmm Volume Menu as shown in Figure 174. DFSMSrmm displays a panel as shown in Figure 175 on page 406.



```
  Panel  Help
------------------------------------------------------------------------------
EDGPT000                    DFSMSrmm Volume Menu
Option ===>

0  OPTIONS   - Specify dialog options and defaults
1  DISPLAY   - Display volume information
2  ADD       - Add a new volume
3  CHANGE    - Change volume information
4  RELEASE   - Delete or release a volume
5  SEARCH    - Search for volumes
6  REQUEST   - Request a volume
7  ADDSCR    - Add one or more SCRATCH volumes
8  CONFIRM   - Confirm librarian or operator actions
9  STACKED   - Add one or more stacked volumes


Enter selected option or END command.  For more info., enter HELP or PF1.



5647-A01 (C) COPYRIGHT 1993,2000 IBM CORPORATION
```

Figure 175. DFSMSrmm Volume Menu

Select option 7, ADDSCR, and press ENTER. DFSMSrmm displays the DFSMSrmm Add Scratch Volumes panel that is shown in Figure 176. Complete the details as shown in the panel and press ENTER:



```
  Panel  Help
------------------------------------------------------------------------------
EDGPT230                DFSMSrmm Add Scratch Volumes
Command ===>

Volume . . . . . . .EDG000            Pool  . . . . . . .
                                 or
Media name . . . . .            Rack  . . . . . . .RMM000
                                Location name . . .

Count  . . . . . . .3         ( Default is 1 )

Description  . . . .
Assigned date  . . .             YYYY/DDD        MVS use . . . . . .
Assigned time  . . .                             VM use  . . . . . .

Media type . . . . .
Label  . . . . . . .SL        ( AL, NL or SL )
  Current version            Label version number( for example 3 )
  Required version           Label version number( for example 4 )
Density  . . . . . .         ( 1600, 3480, 6250 or * )
Initialize . . . . .YES         ( Default is YES )

Press ENTER to ADD one or more SCRATCH volumes, or END command to CANCEL.
```

Figure 176. DFSMSrmm Add Scratch Volumes Panel

DFSMSrmm displays the message `3 volumes added` in the top right hand corner of the screen.

Exit the DFSMSrmm ISPF dialog by entering **=X** on the command line.

5. Initialize tape volumes by editing and submitting the JCL in member EDGIVP1 in SYS1.SAMPLIB. Mount the three tape volumes requested by this job in the sequence EDG002, EDG001, and EDG000.

   Ensure that the job completes with a return code of zero and the expected messages in EDGIVP1 are in the job output.

6. Write data to tape volumes by editing and submitting the JCL in member EDGIVP2 in SYS1.SAMPLIB. Mount the three tape volumes requested by this job in the sequence EDG000, EDG001, and EDG002. Use the three volumes initialized in step 5 on page 406.

   Ensure that all steps of the job complete with a return code of zero. Message IEC502E is issued when the job finishes with the second volume, EDG001. Check that the message in the SYSLOG contains the text RACK=RMM001 on the right hand side as follows:

   ```
   IEC502E RK ddd,EDG001,SL,jjjjjjjj,WRITE22 - RACK=RMM001
   ```

7. To display data set information that is recorded by DFSMSrmm, enter the following RMM TSO subcommands:

   ```
   RMM LISTDATASET 'RMMIVP.TEST1' VOLUME(EDG000) SEQ(1)
   RMM LISTDATASET 'RMMIVP.TEST2' VOLUME(EDG000) SEQ(2)
   RMM LISTDATASET 'RMMIVP.TEST3' VOLUME(EDG001) SEQ(1)
   RMM LISTDATASET 'RMMIVP.TEST4' VOLUME(EDG001) SEQ(2)
   RMM LISTDATASET 'RMMIVP.TEST4' VOLUME(EDG002) SEQ(1)
   ```

   DFSMSrmm displays data set information as shown in Figure 177.

```
Data set name = RMMIVP.TEST1
Job name      = EDGIVP2
Volume        = EDG000                Physical file sequence number = 1
Device number = 0E76       Owner       = RMMIVP   Data set sequence = 1
Create date   = 1997/02/02 Create time = 07:41:29 System ID         = E4E4
Block size    = 80         Block count = 10
Logical Record Length = 80         Record Format  = FB
Date last written     = 1994/02/02 Date last read = 1994/02/02
Step name             = WRITE11    DD name        = SEQOUT
Management class      = MGMT01     VRS management value  =
Storage group         = SGT0S1     VRS retention date    = 1994/07/02
Storage class         = SCFAST     Matching VRS type     =
Data class            = DCCLS1     Matching VRS job name =
Matching VRS name =
Security Class    = U         Description   = UNCLASSIFIED
```

Figure 177. Sample Data Set Information

To cleanup after running the IVP or to prepare to rerun the IVP, issue these commands to remove information from the DFSMSrmm control data set.

```
RMM DELETEVOLUME EDG000 FORCE
RMM DELETEVOLUME EDG001 FORCE
RMM DELETEVOLUME EDG002 FORCE
RMM DELETERACK RMM000 COUNT(10)
```

When you have completed running the IVP, you can return the three volumes to your tape library.

Appendix B. DFSMSrmm Mapping Macros

Note

The following mapping macros have been moved to the *z/OS DFSMSrmm Reporting* document.

- Report Extract Data Set Mapping Macros in SYS1.MACLIB.

  You use the extract data set as input to the DFSMSrmm utility EDGRPTD to create reports.

  The extract data set contains information extracted from the DFSMSrmm control data set.

  The extract data set records contain all major key fields so that you can select fields and sort them for reports. Variable length fields are expanded to maximum length and redundant control information is removed.
- SMF Records Mapping Macros in SYS1.MODGEN.

  DFSMSrmm requires two record types to support audit needs and security needs. You specify the exact SMF record types in EDGRMMxx, using the SMFAUD macro for auditing and the SMFSEC macro for security records.

DFSMSrmm provides the macros identified in this appendix as programming interfaces for customers.

Attention:

Do not use as programming interfaces any DFSMSrmm macros other than those identified in this document.

- Library Control System Interface Macro in SYS1.MODGEN.
  "OAM Interface: EDGLCSUP"
- DFSMSrmm Installation Exit Mapping Macros in SYS1.MODGEN.
  "Installation Exit Mapping Macro: EDGPL100" on page 416
  "Installation Exit Mapping Macro: EDGPL200" on page 420
- DFSMSrmm Sticky Label Mapping Macro in SYS1.MACLIB.
  - "Sticky Label Data: EDGSLAB" on page 422

General-use Programming Interface Mapping Macros

"General-use Programming Interface Mapping Macros" contains General-use Programming Interface and Associated Guidance Information.

OAM Interface: EDGLCSUP

EDGLCSUP maps the Library Control System interface parameter list. See "Managing System-Managed Tape Library Volumes: EDGLCSUX" on page 202 for more information about using the OAM interface.

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE		LCSUP	, LOCV PARAMETER LIST
0	(0)	DBL WORD	8	LCSUP_HDR(0)	CONTROL BLOCK HEADER
THE FOLLOWING FIELDS ARE VALIDATED BY EDGLCSUX BEFORE PROCESSING					
0	(0)	CHARACTER	8	LCSUP_IDENT	CONTROL BLOCK ID
8	(8)	ADDRESS	1	LCSUP_VERNO	CONTROL BLOCK VERSION NUMBER

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
9	(9)	ADDRESS	1	LCSUP_REVNO	CONTROL BLOCK REVISION NUMBER
10	(A)	ADDRESS	2	LCSUP_SUBPOOL	CONTROL BLOCK SUBPOOL NUMBER
12	(C)	ADDRESS	4	LCSUP_LENGTH	CONTROL BLOCK LENGTH
INPUT FIELDS START HERE					
16	(10)	BITSTRING	1	LCSUP_FUNCTION	REQUESTED FUNCTION
		1...		LCSUP_ENT	″X'80'″ CALLER IS CBRUXENT
		.1..		LCSUP_EJC	″X'40'″ CALLER IS CBRUXEJC
		..1.		LCSUP_CUA	″X'20'″ CALLER IS CBRUXCUA
		...1		LCSUP_VNL	″X'10'″ CALLER IS CBRUXVNL
		.... 1...		LCSUP_ACTVNL	″X'08'″ REENTRY FROM CBRUXVNL FOR ACTION
17	(11)	BITSTRING	1		RESERVED
18	(12)	BITSTRING	1	LCSUP_STATUSD	Corresponds to MVSFLGD byte
		1...		LCSUP_MVOREAD	″X'80'″ Owner may read volume
		.1..		LCSUP_MVOUPD	″X'40'″ Owner may update volume
		..1.		LCSUP_MVOALT	″X'20'″ Owner may alter volume
		...1		LCSUP_MVPROTR	″X'10'″ Read-Only protection
		.... 1...		LCSUP_MVPROTU	″X'08'″ Update protection
		.... .1..		LCSUP_MVMVSUSE	″X'04'″ May be used on MVS systems
		.... ..1.		LCSUP_MVVMUSE	″X'02'″ May be used on VM systems
19	(13)	BITSTRING	1	LCSUP_STATUSE	Corresponds to MVSFLGE byte
		1...		LCSUP_MVRETSCR	″X'80'″ Return to scratch pending
		.1..		LCSUP_MVREPREL	″X'40'″ Replace tape on release pending
		..1.		LCSUP_MVREINIT	″X'20'″ Init pending
		...1		LCSUP_MVDEGAUS	″X'10'″ Degaus/security erase pending
		.... 1...		LCSUP_MVROWNER	″X'08'″ Return to owner pending
		.... .1..		LCSUP_MVNOWNER	″X'04'″ Notify owner pending
20	(14)	SIGNED	4	LCSUP_LCSPL	POINTER TO LCS PARAMETER LIST
OUTPUT FIELDS START HERE					
24	(18)	SIGNED	4	LCSUP_LCSRC	RETURN CODE FOR LCS
28	(1C)	SIGNED	4	LCSUP_LCSRS	REASON CODE FOR LCS
OUTPUT FIELDS FOR CBRUXVNL					
32	(20)	CHARACTER	8	LCSUP_LOANLOC	LOAN LOCATION
40	(28)	CHARACTER	8	LCSUP_LOCATION	CURRENT VOLUME LOCATION
48	(30)	BITSTRING	1	LCSUP_LOCTYPE	CURRENT LOCATION TYPE
		....		LCSUP_TYPE_SHELF	″X'00'″ X'00' - SHELF LOCATION
		1		LCSUP_TYPE_STORE	

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
					″X'01'″ X'01' - STORAGE LOCATION
		.... ..1.		LCSUP_TYPE_MTL	
					″X'02'″ X'02' - MANUAL LIBRARY
		11		LCSUP_TYPE_ATL	
					″X'03'″ X'03' - AUTOMATIC LIBRARY
49	(31)	CHARACTER	8	LCSUP_DEST	CURRENT VOLUME DESTINATION
57	(39)	BITSTRING	1	LCSUP_DESTYPE	CURRENT DESTINATION TYPE Values as for LCSUP_LOCTYPE
58	(3A)	CHARACTER	8	LCSUP_HOME	VOLUME HOME LOCATION
66	(42)	BITSTRING	1	LCSUP_HOMETYPE	
					VOLUME HOME LOCATION TYPE Values as for LCSUP_LOCTYPE
67	(43)	CHARACTER	6	LCSUP_RACK	RACK NUMBER
73	(49)	CHARACTER	6	LCSUP_BIN	BIN NUMBER
79	(4F)	BITSTRING	1	LCSUP_STATUS	VOLUME STATUS (MVFLGA)
		1...		LCSUP_MSTFLG	″X'80'″ ** VOLUME IS MASTER
		.1..		LCSUP_RLSFLG	″X'40'″ ** VOLUME PENDING RELEASE
		..1.		LCSUP_VRFLG	″X'20'″ ** VITAL RECORD - DO NOT RELEASE
		...1		LCSUP_ASSFLG	″X'10'″ ** USER TAPE (ASSIGNED BY LIB)
		.... 1...		LCSUP_LONFLG	″X'08'″ ** TAPE IS ON LOAN
		.... .1..		LCSUP_OPNFLG	″X'04'″ ** TAPE OPENED AND NOT YET CLOSED002
		.... ..1.		LCSUP_SCRFLG	″X'02'″ ** VOLUME IS SCRATCH
		1		LCSUP_OCEFLG	″X'01'″ ** VOLUME RECORDED BY O/C/EOV
80	(50)	BITSTRING	1	LCSUP_STATUSX	VOLUME STATUS (MVFLGAX)
		1...		LCSUP_GVCFLG	″X'80'″ ** SCRATCH VOL CLAIMED VIA GETVOL002
		.1..		LCSUP_XINFLG	″X'40'″ ** SCRATCH VOLUME HAS NEVER ** BEEN INITIALISED
		..1.		LCSUP_INIFLG	″X'20'″ ** SCRATCH VOLUME WITH INIT ** ACTION PENDING
		...1		LCSUP_ENTFLG	″X'10'″ ** SCRATCH VOLUME WAITING TO ** ENTER ATL
		.... 1...		LCSUP_FABEND	″X'08'″ ** ABEND IN PROCESS WHEN A DATA ** SET CLOSED
		.... .1..		LCSUP_FOCEAB	″X'04'″ ** ABEND PROBABLY IN O/C/EOV **
		.... ..1.		LCSUP_ATIFLG	″X'02'″ ** INIT REQUESTED FOR ATL VOL
81	(51)	BITSTRING	1	LCSUP_FLAGS	FLAG BYTE
		1...		LCSUP_TRANSIT	″X'80'″ VOLUME MOVING STATUS
84	(54)	SIGNED	4	LCSUP_TDSI	TAPE DEVICE SELECTION INFO.
88	(58)	CHARACTER	16	LCSUP_INCONTAINER	
					In container
104	(68)	SIGNED	1	LCSUP_VOLUMETYPE	
					Volume type
105	(69)	CHARACTER	55		Reserved

```
END OUTPUT FIELDS FOR CBRUXVNL
```

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
160	(A0)	DBL WORD	8	LCSUP_END(0)	END OF LCSUP
		.1.1 1...		LCSUP_LEN#	″LCSUP_END-LCSUP_HDR″ LENGTH OF LCSUP
CONSTANTS USED TO INITIALIZE THE LCSUP HEADER SECTION					
		.... ..1.		LCSUP_VER#	″2″ VERSION NUMBER
		....		LCSUP_REV#	″0″ REVISION NUMBER
		....		LCSUP_SP#	″0″ SUBPOOL NUMBER
CONSTANTS FOR LCSUP_VOLUMETYPE					
		....		LCSUP_VOLUMETYPE_PHYSICAL	″0″ PHYSICAL VOLUME
		1		LCSUP_VOLUMETYPE_LOGICAL	″1″ LOGICAL VOLUME
CONSTANTS FOR RETURN CODES IN R15					
		....		LCSUP_RC_OK	″0″ SUCCESSFUL. A REASON CODE IS SET.
		.... .1..		LCSUP_RC_SSNA	″4″ DFSMSRMM SUBSYSTEM NOT AVAILABLE.
		.... 1...		LCSUP_RC_LERR	″8″ LOGICAL ERROR.
		.... 11..		LCSUP_RC_ENV	″12″ ENVIRONMENT ERROR. A REASON CODE IS SET.
CONSTANTS FOR REASON CODES IN R0 WHEN R15 = LCSUP_RC_OK					
		....		LCSUP_RS_OK	″0″ REQUEST SUCCESSFULLY PROCESSED
		1		LCSUP_RS_NOACTION	″1″ NO ACTION PERFORMED BY DFSMSRMM
		.... ..1.		LCSUP_RS_DONT	″2″ DONT NEED RMM EXITS TO BE CALLED
CONSTANTS FOR REASON CODES IN R0 WHEN R15 = LCSUP_RC_ENV					
		1		LCSUP_RS_IDENT	″1″ INCORRECT VALUE IN LCSUP_IDENT
		.... ..1.		LCSUP_RS_VERNO	″2″ INCORRECT VALUE IN LCSUP_VERNO
		11		LCSUP_RS_REVNO	″3″ INCORRECT VALUE IN LCSUP_REVNO
		.... .1..		LCSUP_RS_SUBPOOL	″4″ INCORRECT VALUE IN LCSUP_SUBPOOL
		.... .1.1		LCSUP_RS_LENGTH	″5″ INCORRECT VALUE IN LCSUP_LENGTH
		.... .11.		LCSUP_RS_FUNCTION	″6″ INCORRECT VALUE IN LCSUP_FUNCTION
		.... .111		LCSUP_RS_NSUPV	″7″ EDGLCSUX NOT SUPERVISOR STATE
		.... 1...		LCSUP_RS_LCSUP	″8″ EDGLCSUX PARAMETER LIST COULD NOT BE ADDRESSED
		.... 1..1		LCSUP_RS_CBRPL	

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
					″9″ CBRUXXPL PARAMETER LIST COULD NOT BE ADDRESSED
		.... 1.1.		LCSUP_RS_ABEND	″10″ EDGLCSUX ABENDED
CONSTANTS FOR REASON CODES RETURNED IN LCSUP_LCSRS					
		1		LCSUP_RS_PBD	″1″ INCONSISTENT PARAMETER LIST
		.... ..1.		LCSUP_RS_NMV	″2″ VOLUME NOT TO BE USED WITH MVS
		11		LCSUP_RS_DEB	″3″ SPECIFIED DESTINATION NOT CURRENT LIB
		.... .1..		LCSUP_RS_RJP	″4″ UNDEF. VOL. REJECTED BY REJECT PREFIX
		.... .1.1		LCSUP_RS_SCR	″5″ PRIVATE TO SCRATCH CHANGE INVALID
		.... .11.		LCSUP_RS_IVU	″6″ USER ID NOT VALID FOR RMM
		.... .111		LCSUP_RS_RPX	″7″ RET. PERIOD EXCEEDS INSTALLATION MAX.
		.... 1...		LCSUP_RS_NRM	″8″ VOLUME NOT RMM MANAGED
		.... 1..1		LCSUP_RS_RIU	″9″ RACK TO MATCH VOLSER NOT AVAILABLE
		.... 1.1.		LCSUP_RS_NSL	″10″ LABEL TYPE IS NOT SUPPORTED IN A LIB.
		.... 1.11		LCSUP_RS_IRK	″11″ VOLUME RACK INCONSISTENT
		.... 11..		LCSUP_RS_REL	″12″ VOLUME PENDING RELEASE
		.... 11.1		LCSUP_RS_STA	″13″ VOLUME STATUS IS SCRATCH
		.... 111.		LCSUP_RS_INI	″14″ VOLUME INIT ACTION PENDING
		 1111		LCSUP_RS_DUPLV	″15″ PHYSICAL VOLUME DUPLICATES EXISTING LOGICAL VOLUME
		...1		LCSUP_RS_NOTEXP	″16″ LOGICAL VOLUME IS NOT EXPORTED
		...1		LCSUP_RS_DUPPV	″17″ LOGICAL VOLUME DUPLICATES EXISTING PHYSICAL VOLUME

EDGLCSUP Cross Reference

Name	Offset	Hex Tag	Level
LCSUP_ACTVNL	10	8	2
LCSUP_ASSFLG	4F	10	2
LCSUP_ATIFLG	50	2	2
LCSUP_BIN	49	40404040	2
LCSUP_CUA	10	20	2
LCSUP_DEST	31	40404040	2
LCSUP_DESTYPE	39	0	2
LCSUP_EJC	10	40	2
LCSUP_END	58		2

Name	Offset	Hex Tag	Level
LCSUP_ENT	10	80	2
LCSUP_ENTFLG	50	10	2
LCSUP_FABEND	50	8	2
LCSUP_FLAGS	51	0	2
LCSUP_FOCEAB	50	4	2
LCSUP_FUNCTION	10	0	2
LCSUP_GVCFLG	50	80	2
LCSUP_HDR	0		2
LCSUP_HOME	3A	40404040	2
LCSUP_HOMETYPE	42	0	2
LCSUP_IDENT	0	C5C4C7D3	2
LCSUP_INCONTAINER	58	0	2
LCSUP_INIFLG	50	20	2
LCSUP_LCSPL	14	0	2
LCSUP_LCSRC	18	0	2
LCSUP_LCSRS	1C	0	2
LCSUP_LEN#	58	58	2
LCSUP_LENGTH	C		2
LCSUP_LOANLOC	20	40404040	2
LCSUP_LOCATION	28	40404040	2
LCSUP_LOCTYPE	30	0	2
LCSUP_LONFLG	4F	8	2
LCSUP_MSTFLG	4F	80	2
LCSUP_MVDEGAUS	13	10	2
LCSUP_MVMVSUSE	12	4	2
LCSUP_MVNOWNER	13	4	2
LCSUP_MVOALT	12	20	2
LCSUP_MVOREAD	12	80	2
LCSUP_MVOUPD	12	40	2
LCSUP_MVPROTR	12	10	2
LCSUP_MVPROTU	12	8	2
LCSUP_MVREINIT	13	20	2
LCSUP_MVREPREL	13	40	2
LCSUP_MVRETSCR	13	80	2
LCSUP_MVROWNER	13	8	2
LCSUP_MVVMUSE	12	2	2
LCSUP_OCEFLG	4F	1	2
LCSUP_OPNFLG	4F	4	2
LCSUP_RACK	43	40404040	2
LCSUP_RC_ENV	58	C	2
LCSUP_RC_LERR	58	8	2
LCSUP_RC_OK	58	0	2
LCSUP_RC_SSNA	58	4	2
LCSUP_REV#	58	0	2
LCSUP_REVNO	9		2
LCSUP_RLSFLG	4F	40	2
LCSUP_RS_ABEND	58	A	2
LCSUP_RS_CBRPL	58	9	2
LCSUP_RS_DEB	58	3	2
LCSUP_RS_DONT	58	2	2
LCSUP_RS_FUNCTION	58	6	2
LCSUP_RS_IDENT	0	C5C4C7D3	2
LCSUP_INCONTAINER	58		2
LCSUP_RS_INI	58	E	2

Name	Offset	Hex Tag	Level
LCSUP_RS_IRK	58	B	2
LCSUP_RS_IVU	58	6	2
LCSUP_RS_LCSUP	58	8	2
LCSUP_RS_LENGTH	58	5	2
LCSUP_RS_NMV	58	2	2
LCSUP_RS_NOACTION	58	1	2
LCSUP_RS_NRM	58	8	2
LCSUP_RS_NSL	58	A	2
LCSUP_RS_NSUPV	58	7	2
LCSUP_RS_OK	58	0	2
LCSUP_RS_PBD	58	1	2
LCSUP_RS_REL	58	C	2
LCSUP_RS_REVNO	58	3	2
LCSUP_RS_RIU	58	9	2
LCSUP_RS_RJP	58	4	2
LCSUP_RS_RPX	58	7	2
LCSUP_RS_SCR	58	5	2
LCSUP_RS_STA	58	D	2
LCSUP_RS_SUBPOOL	58	4	2
LCSUP_RS_VERNO	58	2	2
LCSUP_SCRFLG	4F	2	2
LCSUP_SP#	58	0	2
LCSUP_STATUS	4F	0	2
LCSUP_STATUSD	12	0	2
LCSUP_STATUSE	13	0	2
LCSUP_STATUSX	50	0	2
LCSUP_SUBPOOL	A		2
LCSUP_TDSI	54	0	2
LCSUP_TRANSIT	51	80	2
LCSUP_TYPE_ATL	30	3	2
LCSUP_TYPE_MTL	30	2	2
LCSUP_TYPE_SHELF	30	0	2
LCSUP_TYPE_STORE	30	1	2
LCSUP_VER#	58	1	2
LCSUP_VERNO	8		2
LCSUP_VNL	10	10	2
LCSUP_VOLUMETYPE		2	
LCSUP_VRFLG	4F	20	2
LCSUP_XINFLG	50	40	2

Product-sensitive Programming Interface Mapping Macros

This section contains product-sensitive programming interface and associated guidance information. The macros in this section map the parameter lists for the DFSMSrmm EDGPL100 installation exit and the EDGPL200 installation exit, and the mapping for the EDGSLAB sticky label data area.

Installation Exit Mapping Macro: EDGPL100

EDGPL100 maps the DFSMSrmm installation exit, EDGUX100, parameter list. See “Using the DFSMSrmm EDGUX100 Installation Exit” on page 221 for information about using the EDGPL100 installation exit.

Offset Dec	Offset Hex	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	0	PL100	, EGDUX100 PARAMETER LIST
0	(0)	DBL WORD	8	PL100_HDR(0)	CONTROL BLOCK HEADER
THE FOLLOWING FIELDS CAN BE VALIDATED BY EDGUX100 BEFORE PROCESSING					
0	(0)	CHARACTER	8	PL100_IDENT	CONTROL BLOCK ID
8	(8)	ADDRESS	1	PL100_VERNO	CONTROL BLOCK VERSION NUMBER
9	(9)	ADDRESS	1	PL100_REVNO	CONTROL BLOCK REVISION NUMBER
10	(A)	ADDRESS	2	PL100_SUBPOOL	CONTROL BLOCK SUBPOOL NUMBER
12	(C)	ADDRESS	4	PL100_LENGTH	CONTROL BLOCK LENGTH
INPUT FIELDS START HERE					
16	(10)	BITSTRING	1	PL100_VALID	VALID FUNCTIONS
		1...		PL100_CAN_IGNORE	″X’80’″ CAN REQUEST VOLUME IS IGNORED
		.1..		PL100_CAN_VRS	″X’40’″ CAN REQUEST VRS VALUE SET
		..1.		PL100_CAN_RACKNO	″X’20’″ CAN RETURN RACKNO
		...1		PL100_CAN_IGNORE_FILE2_TON	″X’10’″ CAN REQUEST DS RECORD IGNORE
		.... .1..		PL100_CAN_POOL	″X’04’″ CAN RETURN POOL NAME
		.... ..1.		PL100_ITS_CLOSE	″X’02’″ Exit called at CLOSE/EOV
17	(11)	BITSTRING	1	PL100_INFO	INFORMATION BYTE
		1...		PL100_INFO_IGNORE	″X’80’″ VOLUME IGNORED BY RMM
		.1..		PL100_INFO_NOTRMM	″X’40’″ VOLUME NOT RMM MANAGED
		..1.		PL100_INFO_DISPDD	″X’20’″ DISPDD ENTRY PROCESSED
		...1		PL100_INFO_CMOVE	″X’10’″ CMOVE REQUIRED LATER
		.... 1...		PL100_INFO_USERDATA	″X’08’″ USERDATA PROVIDED
		.... .1..		PL100_INFO_MTL	″X’04’″ ALLOCATED TAPE DRIVE MTL
18	(12)	BITSTRING	2		RESERVED
20	(14)	CHARACTER	6	PL100_REQ_VOLSER	REQUESTED VOLUME SERIAL NUMBER
26	(1A)	CHARACTER	6	PL100_MOUNT_VOLSER	MOUNTED VOLUME SERIAL NUMBER
32	(20)	SIGNED	4	PL100_WTOPTR	ADDRESS OF WTO MESSAGE
OUTPUT FIELDS START HERE					

Offset Dec	Offset Hex	Type	Len	Name(Dim)	Description
36	(24)	BITSTRING	1	PL100_FUNCTION	REQUESTED FUNCTION
		1...		PL100_SET_IGNORE	″X'80'″ CAN REQUEST VOLUME IS IGNORED
		.1..		PL100_SET_IGNORE_MOUNTED	″X'40'″ REQUEST MOUNTED VOL IS IGNORED
		..1.		PL100_SET_IGNORE_REQUESTED	″X'20'″ REQUEST REQ-ED VOL IS IGNORED
		...1		PL100_SET_IGNORE_FILE2_TON	″X'10'″ DS IGNORE HAS BEEN SET
		.... 1...		PL100_SET_NOLABEL	″X'08'″ SUPRESS STICKY LABEL
		.... .1..		PL100_SET_POOL	″X'04'″ POOL NAME HAS BEEN SET
		1		PL100_SET_ACLOFF	″X'01'″ DO NOT PERFORM AN ACL LOAD
37	(25)	BITSTRING	1	PL100_FUNCTION2	REQUESTED FUNCTION
		1...		PL100_SET_CMOVE	″X'80'″ CMOVE REQUIRED LATER
		.1..		PL100_SET_NOCMOVE	″X'40'″ UPDATE VOLUME LOCATION
		..1.		PL100_SET_IGNORE_SGNAME	″X'20'″ SYSTEM BASED SCRATCH POOLING
38	(26)	BITSTRING	2		RESERVED
40	(28)	SIGNED	4	PL100_JFCBPTR	ADDRESS OF JFCB COPY
44	(2C)	CHARACTER	8	PL100_VRS	NEW VRS MANAGEMENT VALUE
52	(34)	CHARACTER	6	PL100_RACKNO	EXTERNAL VOLSER
		..11 .1..		PL100_POOL	″PL100_RACKNO″ SCRATCH POOL NAME
60	(3C)	SIGNED	4	PL100_LABINFO	ADDRESS OF LABEL INFO BLOCK
START OF VERSION 2 FIELDS					
64	(40)	CHARACTER	69	PL100_LAB_USERDATA	USER DATA FOR LABEL PROCESSING
136	(88)	SIGNED	4	PL100_LABPTR	ADDRESS OF PREPARED LABEL
140	(8C)	CHARACTER	8	PL100_LOCATION	TARGET LOCATION NAME
148	(94)	BITSTRING	1	PL100_LOCTYPE	TARGET LOCATION TYPE
		....		PL100_LOC_LOAN	″X'00'″ LOAN LOCATION TYPE
		1		PL100_LOC_STORE	″X'01'″ STORAGE LOCATION TYPE
		.... ..1.		PL100_LOC_LIBRARY	″X'02'″ SYSTEM MANAGED LIBRARY TYPE
149	(95)	CHARACTER	3		RESERVED
152	(98)	CHARACTER	8	PL100_DDNAME	DD NAME
160	(A0)	CHARACTER	8	PL100_DISPDD	DISPDDNAME
168	(A8)	CHARACTER	8	PL100_ACCODE	ACCODE parameter value or blank if no ACCODE or reduced form 'ACCODE=' specified
176	(B0)	SIGNED	4	PL100_ACEROPTR	ADDRESS OF IGDACERO
180	(B4)	CHARACTER	12		RESERVED
192	(C0)	DBL WORD	8	PL100_END(0)	END OF PL100
		11..		PL100_LEN#	″PL100_END-PL100_HDR″ LENGTH OF PL100

CONSTANTS USED TO INITIALIZE THE PL100 HEADER SECTION

Offset Dec	Offset Hex	Type	Len	Name(Dim)	Description
		.... ..1.		PL100_VER#	″2″ VERSION NUMBER
		....		PL100_REV#	″0″ REVISION NUMBER
		....		PL100_SP#	″0″ SUBPOOL NUMBER

EDGPL100 Cross Reference

Name	Offset	Hex Tag	Level
PL100	0		1
PL100_ACCODE	A8		2
PL100_ACEROPTR	B0		2
PL100_AL	2C	40	2
PL100_ASA	5E	4	2
PL100_BIDRC	62	20	2
PL100_BLK#	58		2
PL100_BLKSI	54		2
PL100_BLP	2C	10	2
PL100_CAN_IGNORE	10	80	2
PL100_CAN_IGNORE_FILE2_TON	10	10	2
PL100_CAN_POOL	10	4	2
PL100_CAN_RACKNO	10	20	2
PL100_CAN_VRS	10	40	2
PL100_CRDATE	6B		2
PL100_CRDT	2D		2
PL100_CRJOB	7D		2
PL100_CRTIME	6F		2
PL100_DDNAME	98		2
PL100_DEN	60		2
PL100_DISPDD	A0		2
PL100_DSN	0		2
PL100_END	C0		2
PL100_FEOV	33	20	2
PL100_FIX	5E	80	2
PL100_FOUT	33	80	2
PL100_FSCT	5C		2
PL100_FUNCTION	24	0	2
PL100_FUNCTION2	25		2
PL100_HDR	0		2
PL100_IDENT	0	C5C4C7D7	2
PL100_INFO	11	0	2
PL100_INFO_CMOVE	11	10	2
PL100_INFO_DISPDD	11	20	2
PL100_INFO_IGNORE	11	80	2
PL100_INFO_MTL	11	4	2
PL100_INFO_NOTRMM	11	40	2
PL100_INFO_USERDATA	11	8	2
PL100_ITS_CLOSE	10	2	2
PL100_JFCBPTR	28		2
PL100_JOBNAM	3A		2
PL100_LAB_USERDATA	40		2
PL100_LABDS	0		1
PL100_LABINFO	3C		2
PL100_LABPTR	88		2

Name	Offset	Hex Tag	Level
PL100_LEN#	C0	C0	2
PL100_LENGTH	C		2
PL100_LOC_LIBRARY	94	2	2
PL100_LOC_LOAN	94	0	2
PL100_LOC_STORE	94	1	2
PL100_LOCATION	8C		2
PL100_LOCTYPE	94		2
PL100_LRECL	52		2
PL100_LTYP	2C		2
PL100_MAC	5E	2	2
PL100_MEDIANAME	63		2
PL100_MED1	61	1	2
PL100_MED2	61	2	2
PL100_MED3	61	3	2
PL100_MED4	61	4	2
PL100_MOUNT_VOLSER	1A		2
PL100_NL	2C	1	2
PL100_NOCMP	62	10	2
PL100_NSL	2C	4	2
PL100_NVOL	5F		2
PL100_OFLAG	33		2
PL100_POOL	34	34	2
PL100_PREVOL	77		2
PL100_RACKNO	34		2
PL100_RDCOM	62	1	2
PL100_RECFM	5E		2
PL100_REQ_VOLSER	14		2
PL100_REV#	C0	0	2
PL100_REVNO	9		2
PL100_RFB	5E	10	2
PL100_RFO	5E	20	2
PL100_RFS	5E	8	2
PL100_SET_ACLOFF	24	1	2
PL100_SET_CMOVE	25	80	2
PL100_SET_IGNORE	24	80	2
PL100_SET_IGNORE_FILE2_TON	24	10	2
PL100_SET_IGNORE_MOUNTED	24	40	2
PL100_SET_IGNORE_REQUESTED	24	20	2
PL100_SET_IGNORE_SGNAME	25	20	2
PL100_SET_NOCMOVE	25	40	2
PL100_SET_NOLABEL	24	8	2
PL100_SET_POOL	24	4	2
PL100_SL	2C	2	2
PL100_SP#	C0	0	2
PL100_STPNAM	42		2
PL100_SUBPOOL	A		2
PL100_SYSTEM	4A		2
PL100_TDSI1	61		2
PL100_TDSI2	62		2
PL100_UL	2C	8	2
PL100_UND	5E	C0	2
PL100_UNIT	56		2
PL100_VALID	10	0	2
PL100_VAR	5E	40	2

Name	Offset	Hex Tag	Level
PL100_VER#	C0	2	2
PL100_VERNO	8		2
PL100_VOLSER	34		2
PL100_VRS	2C		2
PL100_WTOPTR	20		2
PL100_XPDATE	73		2
PL100_XPDT	30		2
PL100_128TRK	61	30	2
PL100_18TRK	61	10	2
PL100_36TRK	61	20	2

Installation Exit Mapping Macro: EDGPL200

EDGPL200 maps the DFSMSrmm installation exit, EDGUX200, parameter list. See “Using the DFSMSrmm EDGUX200 Installation Exit” on page 248 for information about using the EDGPL100 installation exit.

Offset	Offset				
Dec	Hex	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	0	PL200	, EDGUX200 PARAMETER LIST
0	(0)	DBL WORD	8	PL200_HDR(0)	CONTROL BLOCK HEADER
THE FOLLOWING FIELDS CAN BE VALIDATED BY EDGUX200 BEFORE PROCESSING					
0	(0)	CHARACTER	8	PL200_IDENT	CONTROL BLOCK ID
8	(8)	ADDRESS	1	PL200_VERNO	CONTROL BLOCK VERSION NUMBER
9	(9)	ADDRESS	1	PL200_REVNO	CONTROL BLOCK REVISION NUMBER
10	(A)	ADDRESS	2	PL200_SUBPOOL	CONTROL BLOCK SUBPOOL NUMBER
12	(C)	ADDRESS	4	PL200_LENGTH	CONTROL BLOCK LENGTH
INPUT FIELDS START HERE					
16	(10)	BITSTRING	1	PL200_VALID	VALID FUNCTIONS
		1...		PL200_CAN_SCRTCH	″X’80’″ CAN HANDLE RETURN TO SCRATCH
17	(11)	BITSTRING	3		RESERVED
20	(14)	CHARACTER	6	PL200_VOLSER	RMM DEFINED VOLUME SERIAL NUMBER
26	(1A)	CHARACTER	6	PL200_RACK_NUMBER	RMM DEFINED RACK NUMBER
32	(20)	CHARACTER	8	PL200_MEDIA_NAME	VOLUME MEDIA NAME
40	(28)	CHARACTER	8	PL200_LOCATION	VOLUME LOCATION
48	(30)	BITSTRING	2	PL200_VOLSEQ	VOLUME SEQUENCE NUMBER
50	(32)	CHARACTER	44	PL200_DSNAME	1ST FILE DATA SET NAME
94	(5E)	BITSTRING	1	PL200_VOLUME_FLAGS	STATUS FLAGS FOR VOLUME
		1...		PL200_SMS_VOL	″X’80’″ VOLUME IS SMS MANAGED
		.1..		PL200_HOME_LOCDEF	″X’40’″ Volume is in storage location defined as home

Offset	Offset				
Dec	Hex	Type	Len	Name(Dim)	Description
		..1.		PL200_MANUAL_SCRATCH	″X'20'″ Volume is in VLPOOL with AUTOSCRATCH(NO)
OUTPUT FIELDS START HERE					
95	(5F)	BITSTRING	1	PL200_FUNCTION	REQUESTED FUNCTION
		1...		PL200_SET_NOSCRTCH	″X'80'″ DO NOT RETURN TO SCRATCH
		.1..		PL200_SET_IGNORE_DSN	″X'40'″ IGNORE DATA SET INFORMATION
96	(60)	CHARACTER	30	PL200_DESCRIPTION	USER DESCRIPTION
126	(7E)	CHARACTER	8	PL200_OWNER	VOLUME OWNER ID
136	(88)	SIGNED	4	PL200_EDGVREC_ADDR	Address of volume information
140	(8C)	CHARACTER	8	PL200_CATSYSID(16)	CATSYSID list for the running system
272	(110)	DBL WORD	8	PL200_END(0)	END OF PL200
272	(110)		0	PL200_LEN#	″PL200_END-PL200_HDR″ LENGTH OF PL200
CONSTANTS USED TO INITIALIZE THE PL200 HEADER SECTION					
		1		PL200_VER#	″1″ VERSION NUMBER
		1		PL200_REV#	″1″ REVISION NUMBER
		....		PL200_SP#	″0″ SUBPOOL NUMBER

EDGPL200 Cross Reference

Name	Offset	Hex Tag	Level
PL200	0		1
PL200_CAN_SCRTCH	10	80	2
PL200_CATSYSID	8C	40404040	2
PL200_DESCRIPTION	60	40404040	2
PL200_DSNAME	32	40404040	2
PL200_EDGVREC_ADDR	88	0	2
PL200_END	110		2
PL200_FUNCTION	5F	0	2
PL200_HDR	0		2
PL200_HOME_LOCDEF	5E	40	2
PL200_IDENT	0	C5C4C7D7	2
PL200_LEN#	110	110	2
PL200_LENGTH	C		2
PL200_LOCATION	28	40404040	2
PL200_MANUAL_SCRATCH	5E	20	2
PL200_MEDIA_NAME	20	40404040	2
PL200_OWNER	7E	40404040	2
PL200_RACK_NUMBER	1A	40404040	2
PL200_REV#	110	1	2
PL200_REVNO	9		2
PL200_SET_IGNORE_DSN	5F	40	2
PL200_SET_NOSCRTCH	5F	80	2
PL200_SMS_VOL	5E	80	2
PL200_SP#	110	0	2

Name	Offset	Hex Tag	Level
PL200_SUBPOOL	A		2
PL200_VALID	10	0	2
PL200_VER#	110	1	2
PL200_VERNO	8		2
PL200_VOLSEQ	30	0	2
PL200_VOLSER	14	40404040	2
PL200_VOLUME_FLAGS	5E	0	2

Sticky Label Data: EDGSLAB

EDGSLAB maps the DFSMSrmm sticky label data area. See Chapter 19, “Setting Up DFSMSrmm Disposition Processing,” on page 389 for more information about the default sticky labels you can request with DFSMSrmm disposition processing.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE		SLAB	
0	(0)	DBL WORD	8	SLABENT (0)	** BEGIN OF SLAB **
0	(0)	CHARACTER	8	SLABID	** SLAB EYECATCHER **
8	(8)	BITSTRING	1	SLABSPL	** SLAB SUBPOOL NUMBER **
9	(9)	BITSTRING	3	SLABSIZE	** SLAB TOTAL SIZE **
12	(C)	BITSTRING	1	SLABKEY	** SLAB PROTECTION KEY**
13	(D)	BITSTRING	1	SLABVER	** SLAB VERSION **
14	(E)	BITSTRING	1	SLABLRECL	** SLAB OUTPUT FILE LRECL**
15	(F)	CHARACTER	1	SLABTYPE	** SLAB LABEL TYPE**
		1...		SLABTYPE_CART	** ″X’80’″ SLAB CARTRIDGE LABEL BUILT **
		.1..		SLABTYPE_REEL	** ″X’40’″ SLAB REEL LABEL BUILT **
16	(10)	BITSTRING	1	SLABCOL	** SLAB NUMBER OF COLUMNS**
17	(11)	BITSTRING	1	SLABROW	** SLAB NUMBER OF ROWS**
20	(14)	SIGNED	4	SLABLAB (0)	** SLAB STICKY LABEL **
20	(14)	CHARACTER	2000	SLABMAX	** **
20	(14)	SIGNED	4	SLABCART (0)	** SLAB CARTRIDGE LABEL LAYOUT**
20	(14)	CHARACTER	80	SLABCLN1	** SLAB RECORD 1 **
100	(64)	CHARACTER	80	SLABCLN2	** SLAB RECORD 2 **
180	(B4)	CHARACTER	80	SLABCLN3	** SLAB RECORD 3 **
260	(104)	CHARACTER	80	SLABCLN4	** SLAB RECORD 4 **
340	(154)	CHARACTER	80	SLABCLN5	** SLAB RECORD 5 **
420	(1A4)	CHARACTER	80	SLABCLN6	** SLAB RECORD 6 **
500	(1F4)	CHARACTER	80	SLABCLN7	** SLAB RECORD 7 **
580	(244)	CHARACTER	80	SLABCLN8	** SLAB RECORD 8 **
660	(294)	CHARACTER	80	SLABCLN9	** SLAB RECORD 9 **
740	(2E4)	CHARACTER	80	SLABCLNA	** SLAB RECORD 10 **
20	(14)	CHARACTER	44	SLABCDSN	** SLAB CART. LABEL DSNAME **
100	(64)	CHARACTER	69	SLABUSR	** SLAB CART. USER DATA **
260	(104)	CHARACTER	1		** **
261	(105)	CHARACTER	8	SLABCJBN	** SLAB CART. LABEL JOBNAME **
269	(10D)	CHARACTER	5		** **
274	(112)	CHARACTER	10	SLABCCRD	** SLAB CART. LABEL CREATE DATE **

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
340	(154)	CHARACTER	14		** **
354	(162)	CHARACTER	10	SLABCEXP	** SLAB CART. LABEL EXPIR. DATE **
420	(1A4)	CHARACTER	1		** **
421	(1A5)	CHARACTER	4	SLABCDEN	** SLAB CART. LABEL DENSITY **
425	(1A9)	CHARACTER	1		** **
426	(1AA)	CHARACTER	4	SLABCCMP	** SLAB CART. LABEL COMPACTION **
430	(1AE)	CHARACTER	1		** **
431	(1AF)	CHARACTER	5	SLABCLRC	** SLAB CART. LABEL LRECL **
436	(1B4)	CHARACTER	2		** **
438	(1B6)	CHARACTER	5	SLABCBLK	** SLAB CART. LABEL BLKSIZE **
443	(1BB)	CHARACTER	2		** **
445	(1BD)	CHARACTER	4	SLABCRCF	** SLAB CART. LABEL RECFM **
580	(244)	CHARACTER	1		** **
581	(245)	CHARACTER	6	SLABCVSL	** SLAB CART. LABEL VOLSER **
587	(24B)	CHARACTER	1		** **
588	(24C)	CHARACTER	4	SLABCSQN	** SLAB CART. LABEL VOL SEQUENCE **
592	(250)	CHARACTER	1		** **
593	(251)	CHARACTER	3	SLABCLAB	** SLAB CART. LABEL VOL LABL TYPE **
596	(254)	CHARACTER	7		** **
603	(25B)	CHARACTER	4	SLABCDVC	** SLAB CART. LABEL DEVICE NUMBER **
20	(14)	SIGNED	4	SLABTAPE (0)	** SLAB TAPE LABEL LAYOUT **
20	(14)	CHARACTER	80	SLABTLN1	** SLAB RECORD 1 **
100	(64)	CHARACTER	80	SLABCTLN2	** SLAB RECORD 2 **
180	(B4)	CHARACTER	80	SLABTLN3	** SLAB RECORD 3 **
260	(104)	CHARACTER	80	SLABTLN4	** SLAB RECORD 4 **
340	(154)	CHARACTER	80	SLABTLN5	** SLAB RECORD 5 **
420	(1A4)	CHARACTER	80	SLABTLN6	** SLAB RECORD 6 **
500	(1F4)	CHARACTER	80	SLABTLN7	** SLAB RECORD 7 **
580	(244)	CHARACTER	80	SLABTLN8	** SLAB RECORD 8 **
660	(294)	CHARACTER	80	SLABTLN9	** SLAB RECORD 9 **
740	(2E4)	CHARACTER	80	SLABTLNA	** SLAB RECORD 10 **
20	(14)	CHARACTER	44	SLABTDSN	** SLAB TAPE LABEL DSNAME **
100	(64)	CHARACTER	69	SLABTUSR	** SLAB TAPE USER DATA **
180	(B4)	CHARACTER	4		** **
184	(B8)	CHARACTER	8	SLABTJBN	** SLAB TAPE. LABEL JOBNAME **
192	(C0)	CHARACTER	18		** **
210	(D2)	CHARACTER	10	SLABTCRD	** SLAB TAPE LABEL CREATE DATE **
340	(154)	CHARACTER	1		** **
341	(155)	CHARACTER	4	SLABTDEN	** SLAB TAPE DENSITY **
345	(159)	CHARACTER	1		** **
346	(15A)	CHARACTER	4	SLABTCMP	** SLAB TAPE LABEL COMPACTION **

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
350	(15E)	CHARACTER	1		** **
351	(15F)	CHARACTER	5	SLABTLRC	** SLAB TAPE LABEL LRECL **
356	(164)	CHARACTER	2		** **
358	(166)	CHARACTER	5	SLABTBLK	** SLAB TAPE LABEL BLKSIZE **
363	(16B)	CHARACTER	2		** **
365	(16D)	CHARACTER	4	SLABTRCF	** SLAB TAPE LABEL RECFM **
369	(171)	CHARACTER	1		** **
370	(172)	CHARACTER	10	SLABTEXP	** SLAB TAPE LABEL EXPIR. DATE **
500	(1F4)	CHARACTER	15		** **
515	(203)	CHARACTER	6	SLABTVSL	** SLAB TAPE LABEL VOLSER **
521	(209)	CHARACTER	1		** **
522	(20A)	CHARACTER	4	SLABTSQN	** SLAB TAPE LABEL VOL SEQUENCE **
526	(20E)	CHARACTER	1		** **
527	(20F)	CHARACTER	3	SLABTLAB	** SLAB TAPE LABEL VOL LABL TYPE **
530	(212)	CHARACTER	6		** **
536	(218)	CHARACTER	4	SLABTDVC	** SLAB TAPE LABEL DEVICE NUMBER **
2024	(7E8)	DBL WORD	8	SLABEND (0)	**END OF SLAB**
					″SLABEND-SLABENT″
				SLABLNG	** LENGTH OF SLAB **
FLAG BYTES					
		1		SLABVER#	″1’″ ** VERSION NUMBER **
		.... .1.1		SLABKEY#	″5″ ** KEY NUMBER**
		111. .11.		SLABSP#	″230″ ** SUBPOOL NUMBER**
		.... 1.1.		SLABROW#	″10″ ** DEFAULT NUMBER OF ROWS**
		.1.1		SLABCOL#	″80″ ** DEFAULT NUMBER OF COLS**
		.1.1		SLABLRECL#	″80″ ** DEFAULT LRECL **

EDGSLAB Cross Reference

Name	Offset	Hex Tag	Level
SLABCART	14		2
SLABCBLK	1B6		2
SLABCCMP	1AA		2
SLABCCRD	112		2
SLABCDEN	1A5		2
SLABCDSN	14		2
SLABCDVC	25B		2
SLABCEXP	162		2
SLABCJBN	105		2
SLABCLAB	251		2
SLABCLNA	2E4		2
SLABCLN1	14		2
SLABCLN2	64		2
SLABCLN3	B4		2
SLABCLN4	104		2

Name	Offset	Hex Tag	Level
SLABCLN5	154		2
SLABCLN6	1A4		2
SLABCLN7	1F4		2
SLABCLN8	244		2
SLABCLN9	294		2
SLABCLRC	1AF		2
SLABCOL	10		2
SLABCOL#	7E8	50	2
SLABCRCF	1BD		2
SLABCSQN	24C		2
SLABCUSR	64		2
SLABCVSL	245		2
SLABEND	7E8		2
SLABENT	0		2
SLABID	0		2
SLABKEY	C		2
SLABKEY#	7E8	5	2
SLABLAB	14		2
SLABLNG	7E8	7E8	2
SLABLRECL	E		2
SLABLRECL#	7E8	50	2
SLABMAX	14		2
SLABROW	11		2
SLABROW#	7E8	A	2
SLABSIZE	9		2
SLABSP#	7E8	E6	2
SLABSPL	8		2
SLABTAPE	14		2
SLABTBLK	166		2
SLABTCMP	15A		2
SLABTCRD	D2		2
SLABTDEN	155		2
SLABTDSN	14		2
SLABTDVC	218		2
SLABTEXP	172		2
SLABTJBN	B8		2
SLABTLAB	20F		2
SLABTLNA	2E4		2
SLABTLN1	14		2
SLABTLN2	64		2
SLABTLN3	B4		2
SLABTLN4	104		2
SLABTLN5	154		2
SLABTLN6	1A4		2
SLABTLN7	1F4		2
SLABTLN8	244		2
SLABTLN9	294		2
SLABTLRC	15F		2
SLABTRCF	16D		2
SLABTSQN	20A		2
SLABTUSR	64		2
SLABTVSL	203		2
SLABTYPE	F		2
SLABTYPE_CART	F	80	2

Name	Offset	Hex Tag	Level
SLABTYPE_REEL	F	40	2
SLABVER	D		2
SLABVER#	7E8	1	2

Appendix C. Using DFSMSrmm Samples

DFSMSrmm provides several samples in SAMPLIB, SMPSTS, and SYS1.SEDGEXE1. Table 58 lists the samples that are available and where they can be found after SMP/E APPLY processing. After SMP/E ACCEPT processing, samples in SAMPLIB move to ASAMPLIB and samples in SMPSTS move to the AEDGSRC1 library.

You can use the IBM Tivoli Workload Scheduler for z/OS sample jobs or procedures with other scheduling systems. In some cases, you must modify the sample jobs.

Table 58. SAMPLIB and SMPSTS Members

Member Name	Shows You How To	Supplied In
CBRUXCUA	Use programming interface to EDGLCSUX	SMPSTS
CBRUXEJC	Use programming interface to EDGLCSUX	SMPSTS
CBRUXENT	Use programming interface to EDGLCSUX	SMPSTS
CBRUXVNL	Use programming interface to EDGLCSUX	SMPSTS
EDG3IIP1	Update IATIIP1 to force DEFER for all tape requests	SAMPLIB
EDG3LVVR	Update IATLVVR to AWAIT MSGDISP for scratch mounts	SAMPLIB
EDG3UX29	Install a JES3 USERMOD	SAMPLIB
EDG3UX62	Update IATUX62 to override JES3 rejection of standard label tapes	SAMPLIB
EDG3UX71	Update IATUX71 to replace and append text to JES3 fetch and mount messages and to provide text for tape drive displays	SAMPLIB
EDGBETT	Sample procedure for Tivoli event trigger tracking of backup	SAMPLIB
EDGBKP1	Sample Tivoli job for running backup	SAMPLIB
EDGBKP2	Sample Tivoli job for running backup	SAMPLIB
EDGCLIBQ	Use reports for VM tape volumes	SAMPLIB
EDGCLMS	Convert volume information into commands	SAMPLIB
EDGDFRMM	Create a procedure in SYS1.PROCLIB	SAMPLIB
| EDGHCLT	Sample to issue RMM subcommands using DFSMSrmm classes and methods	SAMPLIB
EDGIVPPM	Parmlib member for supplied Installation Verification Program (IVP)	SAMPLIB
EDGIVP1	IVP job 1 - initializes tape volumes	SAMPLIB
EDGIVP2	IVP job 2 - uses tape volumes	SAMPLIB
EDGJACTP	Print the ACTIVITY file	SAMPLIB
EDGJAUDM	Create a monthly archive from weekly audit reports	SAMPLIB
EDGJAUDW	Create a weekly archive from daily audit reports	SAMPLIB
EDGJBCAV	Build RMM ADDVOLUME subcommands from a list of barcode scanned volumes	SAMPLIB
EDGJBKUP	Sample JCL for using the backup program	SAMPLIB
EDGCMOV	Sample Tivoli job for confirming volume moves	SAMPLIB
EDGJCOMB	Audit tape library using a list of barcode scanned volumes	SAMPLIB
EDGJCVB	Create a report of volumes in a storage location	SAMPLIB
EDGJDHKP	Sample Tivoli job for running daily inventory management	SAMPLIB
EDGJDSN	Create a report of data sets sorted by data set name	SAMPLIB
EDGJEJC	Sample Tivoli job for ejecting volumes from system-managed libraries	SAMPLIB
EDGJEXP	Sample Tivoli job for running expiration processing	SAMPLIB
EDGJHKPA	Sample JCL for allocating the data sets required for inventory management	SAMPLIB

Table 58. SAMPLIB and SMPSTS Members (continued)

Member Name	Shows You How To	Supplied In
EDGJHSKP	Sample JCL for using the utility program EDGHSKP	SAMPLIB
EDGJIMPC	Sample JCL to create an import list from CLIST output	SAMPLIB
EDGJINER	Sample JCL for using the utility program EDGINERS	SAMPLIB
EDGJLOPC	Sample JCL for running the IBM Tivoli Workload Scheduler for z/OS batch loader utility to define DFSMSrmm as an application to IBM Tivoli Workload Scheduler for z/OS	SAMPLIB
EDGJMFAL	Sample JCL for allocating the control data set	SAMPLIB
EDGJMOVE	Sample Tivoli job for creating movement reports	SAMPLIB
EDGJNLAL	Sample JCL for allocating the journal	SAMPLIB
EDGJNSCR	Create a report of volumes recently returned to scratch status	SAMPLIB
EDGJRACK	Create a report based on rack number prefixes	SAMPLIB
EDGJRECL	Create a report containing information about lost volumes	SAMPLIB
EDGJRECV	Build RMM subcommands to add volumes to DFSMSrmm	SAMPLIB
EDGJROWN	Create a report about owners sorted by name and department number	SAMPLIB
EDGJRPT	Sample JCL to create reports using the extended report extract file	SAMPLIB
EDGJRVOL	Create a report about volumes; by volume serial number, by rack number, by security level, by owner, and by expiration date	SAMPLIB
EDGJSCRL	Sample Tivoli job for creating scratch listings	SAMPLIB
EDGJSMF	Create a report of SMF records	SAMPLIB
EDGJSMFP	Create a list of types of SMF record found	SAMPLIB
EDGJUTIL	Sample JCL for initializing the control data set	SAMPLIB
EDGJVFY	Sample Tivoli job for verifying control data set contents	SAMPLIB
EDGJVLT	Create a report about volumes currently in storage locations sorted by volume serial number	SAMPLIB
EDGJVLTM	Create a report about volumes moving to storage locations	SAMPLIB
EDGJVME	Sample JCL for creating reports for VM tape volumes	SAMPLIB
EDGJVOL	Create a report about volumes sorted by volume serial number	SAMPLIB
EDGJVRSV	Sample Tivoli job for running vital record processing trial run	SAMPLIB
EDGJWHKP	Sample Tivoli job for running weekly inventory management	SAMPLIB
EDGLABEL	Started procedure for initializing and erasing tapes	SAMPLIB
EDGPACTA	Sample Tivoli procedure for allocating ACTIVITY report files	SAMPLIB
EDGPACTC	Sample Tivoli procedure sort input	SAMPLIB
EDGPACTD	Sample Tivoli procedure sort input	SAMPLIB
EDGPACTI	Sample Tivoli procedure ICETOOL control statements	SAMPLIB
EDGPACTM	Sample Tivoli procedure sort input	SAMPLIB
EDGPACTP	Sample Tivoli procedure reporting on ACTIVITY file	SAMPLIB
EDGPACTT	Sample Tivoli procedure sort input	SAMPLIB
EDGPACTV	Sample Tivoli procedure sort input	SAMPLIB
EDGPBKUP	Sample Tivoli procedure for control data set backup	SAMPLIB
EDGPCMOV	Sample Tivoli procedure for global volume move confirmation	SAMPLIB
EDGPEJC	Sample Tivoli procedure for ejecting volumes	SAMPLIB
EDGPEXP	Sample Tivoli procedure for running expiration processing	SAMPLIB

Table 58. SAMPLIB and SMPSTS Members (continued)

Member Name	Shows You How To	Supplied In
EDGPHKP	Sample Tivoli procedure for running inventory management	SAMPLIB
EDGPHKPA	Sample Tivoli procedure for allocating inventory management data sets	SAMPLIB
EDGPINER	Sample Tivoli procedure for labeling and erasing tapes	SAMPLIB
EDGPMOVE	Sample Tivoli procedure for creating movement reports	SAMPLIB
EDGPMSGA	Sample Tivoli procedure for allocating the next generation of the MESSAGE file	SAMPLIB
EDGPMSGC	Sample Tivoli procedure for copying the MESSAGE file and creating the next generation of the MESSAGE file	SAMPLIB
EDGPRPTA	Sample Tivoli procedure for allocating the next generation of the report extract file	SAMPLIB
EDGPRPTX	Sample Tivoli procedure for creating the report extract file	SAMPLIB
EDGPSCRL	Sample Tivoli procedure for creating the scratch list report	SAMPLIB
EDGPVFY	Sample Tivoli procedure for verifying the contents of the control data set	SAMPLIB
EDGPVRSA	Sample Tivoli procedure for allocating the next generation of the REPORT file and the ACTIVITY file	SAMPLIB
EDGRHKPA	Sample Tivoli exec for defining GDG bases	SAMPLIB
EDGRCSCR	REXX Exec to convert pool information to ACS routine input and VLPOOL definitions	EDGEXE1
EDGRRPTE	REXX Exec to create reports using the extended report extract file	EDGEXE1
EDGRRPTM	REXX Exec to create an extended extract file only for multiple data sets and multivolume reporting	EDGEXE1
EDGRRPTR	REXX Exec to create an extended report extract file	EDGEXE1
EDGRVCLN	REXX Exec to report and update existing vital record specifications	EDGEXE1
EDGSETT	Sample Tivoli procedure for event trigger tracking of low-on-scratch volume condition	SAMPLIB
EDGUX100	Use the installation exit EDGUX100	SAMPLIB
EDGUX200	Use the installation exit EDGUX200	SAMPLIB
| EDGXCAPI	Sample shows how to issue RMM subcommands using the DFSMSrmm API classes and methods	SAMPLIB
EDGXMP1	REXX EXEC to list all volumes in a multivolume set	SAMPLIB
EDGXMP2	REXX EXEC to list all data set information for a given volume	SAMPLIB
| EDGXMP3	REXX EXEC to show how the EDGRLCL exec can be coded to handle the new 'U' line command.	SAMPLIB
EDGXPROC	Replenish scratch volumes in a automated tape library	SAMPLIB
IGXMSGEX	Use programming interface to EDGMSGEX	SMPSTS

Appendix D. Evaluating Removable Media Management Needs

Use the following list of questions to assess your current tape management practices and anticipate future requirements. You need your answers to these questions later, when you assess direct access storage device (DASD) needs for the control data set, journal, and report extract data set.

If you plan to change anything about the removable media library, such as increasing the number of volumes, consider the changes shown in Table 59 when identifying your DASD needs.

Table 59. Evaluating Removable Media Management Needs

Task	Subtask
Determine the number of resources you have in your removable media library.	How many volumes do you have in your removable media library? A *volume* is any type of removable media, such as a tape cartridge or an optical disk. Add an average of five volumes in your count for each software product in your installation. How many shelf locations or slots do you maintain in your removable media library and in your storage locations? A *shelf location* is a single space on a shelf where you store a volume. Count all shelves in the library and in your storage locations. For DFSMSrmm subcommands and the ISPF dialog, shelf locations in the removable media library are called *rack numbers*. Shelf locations in storage locations are called *bin numbers*. How many data sets do you have on removable media? Count any data sets on your removable media. How many different individuals or groups use removable media? DFSMSrmm can keep track of owners and of removable media in the DFSMSrmm control data set.
Determine the number of requests submitted to your removable media library.	How many scratch tape mounts are performed daily? A *scratch* tape mount is a non-specific tape mount as, for example, when someone requests a blank tape. How many non-scratch tape mounts are performed daily? A *non-scratch* tape mount is a specific tape mount as, for example, when someone requests a tape he or she owns or a software product tape.
Determine the types of activities taking place in your media library.	What activities are performed to support disaster recovery and vital records management? How many volumes enter and leave your removable media library daily? This includes volumes moving to storage locations for disaster recovery and vital records, as well, as foreign tapes entering your library. How many volumes are returned to scratch daily? This number can be used to calculate the space required for the journal. How many volumes expire daily? This number can be used to calculate the space required for the journal. How many logical volumes are imported and exported daily? This number can be used to calculate the space required for the journal.
Determine the number of information changes that might be made to DFSMSrmm information.	This number can be used to calculate the space required for the journal. Changes include information about data sets, owners, software products, or volumes made by using the DFSMSrmm TSO subcommands or DFSMSrmm ISPF dialog.

Appendix E. Problem Determination Aid Log Data Set Size Work Sheet for Long-Term Trace History

Use the following work sheet to calculate the size of your PDA log data set (long term).

1. Fill in the blanks with values for your installation.

   ```
   _______  =  ?UID         -  The high-level qualifier you want to
                               use for the PDA log data sets.
   _______  =  ?HOSTID      -  The identifier for the processing unit
                               at your site.
   _______  =  ?TRACEUNIT   -  The unit identifier for the device on
                               which you want to allocate the PDA log
                               data sets.
   _______  =  ?TRACEVOL    -  The serial number for the volume on
                               which you want to put your PDA log
                               data sets.
   ```

2. Allocate the minimum recommended storage for PDA log data sets: 20 cylinders.

   Substitute the values you have provided in step 1 of this work sheet, and run the JCL job shown in "Allocating the Problem Determination Aid (PDA) Log Data Sets" on page 387 to allocate and catalog the PDA log data sets.

   If you allocated these data sets as SMS-managed data sets, they must be allocated on a specific volume and they must be associated with a storage class having the GUARANTEED SPACE attribute.

3. Allocate a generation data group (GDG) in which you can archive your site's trace history data.

   The following example defines the generation data group (GDG) name for the archived problem determination output data set. Substitute the applicable values you provided in step 1 of this work sheet, and run the JCL job shown in "Archiving the Problem Determination Aid (PDA) Log Data Sets" on page 388 to create a generation data group.

4. Develop a procedure to automatically copy your PDA log data sets to tape.

   The following example shows you how to copy the inactive trace data set to tape as a generation data set (GDS). Substitute the applicable values you have provided in step 1 of this work sheet, and run the JCL job shown in "Copying the Problem Determination Aid (PDA) Log Data Sets to Tape" on page 388 to automatically copy your PDA log data sets to tape.

Appendix F. Problem Determination Aid Log Data Set Size Work Sheet for Short-Term Trace History

Use the following work sheet to calculate the size of your PDA log data set (short term).

1. Fill in the following blanks with values for your installation.

```
_______  =  ?tracehours   -  The number of hours of trace history
                             you want to retain.
_______  =  ?UID          -  The high-level qualifier you want to
                             use for the PDA log data sets.
_______  =  ?HOSTID       -  The identifier for the processing unit
                             at your site.
_______  =  ?TRACEUNIT    -  The unit identifier for the device on
                             which you want to allocate the PDA log
                             data sets.
_______  =  ?TRACEVOL     -  The serial number for the volume on
                             which you want to put your PDA log
                             data sets.
```

2. Allocate the minimum recommended storage for PDA log data sets which is 20 cylinders.

   Substitute the values you used in step 1 of this work sheet, and run the JCL job shown in Figure 178 to allocate and catalog the PDA log data sets.

```
//ALLOPDO JOB MSGLEVEL=1,TYPRUN=HOLD
//STEP1   EXEC PGM=IEFBR14
//DD1     DD DSN=?UID..?HOSTID..RMMPDOX,DISP=(,CATLG),
//           UNIT=?TRACEUNIT.,VOL=SER=?TRACEVOL.,SPACE=(CYL,(20))
//DD2     DD DSN=?UID..?HOSTID..RMMPDOY,DISP=(,CATLG),
//           UNIT=?TRACEUNIT.,VOL=SER=?TRACEVOL.,SPACE=(CYL,(20))
```

Figure 178. JCL for Allocating and Cataloging PDA Log Data Sets

   If you have allocated these data sets as SMS-managed, they must be allocated on a specific volume and they must be associated with a storage class having the GUARANTEED SPACE attribute.

3. Measure the cylinders per hour trace history generation rate at your site.

   After one hour of processing (during a time of high DFSMSrmm activity), measure the amount of storage used to record that hour’s trace activity. Issue the following MODIFY command to swap the EDGPDOX and EDGPDOY data sets. After you have swapped these data sets, the EDGPDOY data set will be ready to measure and the EDGPDOX data set will be ready to receive additional trace data.

   ```
   F DFRMM,PDALOG=SWAP
   ```

   Use the information gathered in this step to calculate the cylinders per hour.

   ```
   Cylinders/hr = cylinders per hour of trace history
   ```

4. Calculate the total amount of cylinders required for your site’s trace history data.

   ```
   ((tracehours = _______) x (cylinders/hr = _______))  =  _______

     Total = total number of cylinders of trace data
   ```

5. Divide in half the total cylinders required for your short-term trace history interval. If the result is a fraction, round up to the next whole number.

   ```
          (Total =       )      =  _______
          ---------------
                2
   ```

This step provides the total number of cylinders to allocate for each data set.

Appendix G. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`www.ibm.com/servers/eserver/zseries/zos/bkserv/`

One exception is command syntax that is published in railroad track format; screen-readable copies of z/OS books with that syntax information are separately available in HTML zipped file form upon request to mhvrcfs@us.ibm.com.

Notices

Programming interface information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of DFSMSrmm.

Trademarks

Glossary

This glossary defines technical terms and abbreviations used in DFSMS documentation. If you do not find the term you are looking for, refer to the index of the appropriate DFSMS manual or view the *Glossary of Computing Terms* located at:

http://www.ibm.com/ibm/terminology/

This glossary includes terms and definitions from:

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published part of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.
- The *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

The following cross-reference is used in this glossary:

See: This refers the reader to (a) a related term, (b) a term that is the expanded form of an abbreviation or acronym, or (c) a synonym or more preferred term.

A

abend. Abnormal end of task.

AL. American National Standards Label.

AMODE. Addressing mode.

ANDVRS. An RMM ADDVRS TSO subcommand operand. See also*Using AND*.

ANSI. American National Standards Institute.

APAR. Authorized program analysis report.

API. Application Programming interface.

ASA. American Standards Association.

assigned date. The date that the volume is assigned to the current owner. Assigned date is not meaningful for a scratch volume.

AUL. ANSI and user header or trailer label.

automated tape library data server. A device consisting of robotic components, cartridge storage areas, tape subsystems, and controlling hardware and software, together with the set of tape volumes that reside in the library and can be mounted on the library tape drives. Contrast with *manual tape library*. See also *tape library*.

automatic cartridge loader. An optional feature of the 3480 Magnetic Tape Subsystem that allows preloading of multiple tape cartridges. This feature is standard in the 3490 Magnetic Tape Subsystem.

automatic recording. In DFSMSrmm, the process of recording information about a volume and the data sets on the volume in the DFSMSrmm control data set at open or close time.

availability. For a storage subsystem, the degree to which a data set or object can be accessed when requested by a user.

B

backup. The process of creating a copy of a data set or object to be used in case of accidental loss.

basic catalog structure (BCS). The name of the catalog structure in the catalog environment.

BCS. Basic catalog structure.

bin number. The specific shelf location where a volume resides in a storage location; equivalent to a rack number in the removable media library. See also *shelf location*.

BLP. Bypass label processing.

BTLS. Basic Tape Library Support.

built-in storage location. One of the Removable Media Manager defined storage locations: LOCAL, DISTANT, and REMOTE.

C

cache fast write. A storage control capability in which the data is written directly to cache without using nonvolatile storage. Cache fast write is useful for temporary data or data that is readily recreated, such as the sort work files created by DFSORT. Contrast with *DASD fast write.*

cartridge eject. For an IBM Total Storage Enterprise Automated Tape Library (3494), IBM TotalStorage Enterprise Automated Tape Library (3495), or a manual tape library, the act of physically removing a tape cartridge, usually under robot control, by placing it in an output station. The software logically removes the cartridge by deleting or updating the tape volume record in the tape configuration database. For a manual tape library, the act of logically removing a tape cartridge from the manual tape library by deleting or updating the tape volume record in the tape configuration database.

cartridge entry. For either an IBM Total Storage Enterprise Automated Tape Library (3494), IBM TotalStorage Enterprise Automated Tape Library (3495), or a manual tape library, the process of logically adding a tape cartridge to the library by creating or updating the tape volume record in the tape configuration database. The cartridge entry process includes the assignment of the cartridge to scratch or private category in the library.

Cartridge System Tape. The base tape cartridge media used with 3480 or 3490 Magnetic Tape Subsystems. Contrast with *Enhanced Capacity Cartridge System Tape*.

cell. A single cartridge location within an automated tape library dataserver. See also *rack number.*

circular file. A type of file that appends data until full. Then, starting at the beginning of the file, subsequent incoming data overwrites the data already there.

client. (1) A user. (2) A consumer of resources or services. (3) A functional unit that receives shared services from a server. (4) A system that is dependent on a server to provide it with programs or access to programs. (5) On a network, the computer requesting services or data from another computer.

client-server. (1) In TCP/IP, the model of interaction in distributed data processing in which a program at one site sends a request to a program at another site and waits for a response. The requesting program is called a client; the answering program is called a server. (2) A model of computer interaction in which a server provides resources for other systems on a network, and a client accesses those resources. See also *client, client-server relationship, server*.

client-server relationship. Any process that provides resources to other processes on a network is a *server*. Any process that employs these resources is a *client*. A machine can run client and server processes at the same time.

command line. On a display screen, a display line usually at the bottom of the screen in which only commands can be entered.

concurrent copy. A function to increase the accessibility of data by enabling you to make a consistent backup or copy of data concurrent with the usual application program processing.

confirmation panel. A DFSMSrmm panel that lets you tell DFSMSrmm to continue or stop a delete or release action. You specify whether or not you want to confirm delete or release requests in your dialog user options.

container. A receptacle in which one or more exported logical volumes can be stored. A stacked volume containing one or more logical volumes and residing outside a virtual tape server library is considered to be the container for those volumes.

container volume. See *container*.

control data set. A VSAM key-sequenced data set that contains the complete inventory of your removable media library, as well as the movement and retention policies you define. In the control data set DFSMSrmm records all changes made to the inventory, such as adding or deleting volumes.

control data set ID. A one-to-eight character identifier for the DFSMSrmm control data set used to ensure that, in a multi-system, multi-complex environment, the correct management functions are performed.

convenience input. The process of adding a small number of tape cartridges to the IBM TotalStorage Enterprise Automated Tape Library (3494) and the IBM TotalStorage Enterprise Automated Tape Library (3495) without interrupting operations, by inserting the cartridges directly into cells in a convenience input station.

convenience input/output station. A transfer station with combined tape cartridge input and output functions in the IBM TotalStorage Enterprise Automated Tape Libraries (3494) only.

convenience output. The process of removing a small number of tape cartridges from the IBM TotalStorage Enterprise Automated Tape Library (3494) or the IBM TotalStorage Enterprise Automated Tape Library (3495) without interrupting operations, by removing the cartridges directly from cells in a convenience input station.

convenience output station. A transfer station, used by the operator to remove tape cartridges from the automated tape library dataserver, which is accessible from outside the enclosure area.

conversion. In DFSMSrmm, the process of moving your removable media library inventory from another media management system to DFSMSrmm. DFSMSrmm manages the inventory and policies once you have converted it.

create date. Create date for a data set is the date that the data set is written to tape. Create date can also be the date a data set was read if it was created before DFSMSrmm is in use. Create date is updated each time a data set is replaced and not extended. Create date for volumes and other resources defined to DFSMSrmm is the date the resource is defined to DFSMSrmm or the date specified on the command as the create date.

D

DASD. Direct access storage device.

DASD fast write. An extended function of some models of the IBM 3990 Storage Control in which data is written concurrently to cache and nonvolatile storage and automatically scheduled for destaging to DASD. Both copies are retained in the storage control until the data is completely written to the DASD, providing data integrity equivalent to writing directly to the DASD. Use of DASD fast write for system-managed data sets is controlled by storage class attributes to improve performance. See also *dynamic cache management*. Contrast with *cache fast write*.

DASD volume. A DASD space identified by a common label and accessed by a set of related addresses. See also *volume, primary storage, migration level 1, migration level 2*.

data column. A vertical arrangement of identical data items, used on list panels to display an attribute, characteristic, or value of one or more objects.

data control block (DCB). A control block used by access method routines in storing and retrieving data.

data entry panel. A panel in which the user communicates with the system by filling in one or more fields.

Data Facility Storage Management Subsystem (DFSMS). An operating environment that helps automate and centralize the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and automatic class selection routine definitions.

Data Facility Sort. An IBM licensed program that is a high-speed data processing utility. DFSORT provides an efficient and flexible way to handle sorting, merging, and copying operations, as well as providing versatile data manipulation at the record, field, and bit level.

DCB. Data control block.

device. This term is used interchangeably with unit. You mount a tape on a unit or device, such as a 3490.

DFSMSdfp. A DFSMS functional component or base element of z/OS, that provides functions for storage management, data management, program management, device management, and distributed data access.

DFSMSdss. A DFSMS functional component or base element of z/OS, used to copy, move, dump, and restore data sets and volumes.

DFSMShsm. A DFSMS functional component or base element of z/OS, used for backing up and recovering data, and managing space on volumes in the storage hierarchy.

DFSMShsm-managed volume. (1) A primary storage volume, which is defined to DFSMShsm but which does not belong to a storage group. (2) A volume in a storage group, which is using DFSMShsm automatic dump, migration, or backup services. Contrast with *system-managed volume, DFSMSrmm-managed volume*.

DFSMShsm-owned volume. A storage volume on which DFSMShsm stores backup versions, dump copies, or migrated data sets.

DFSMSrmm. A DFSMS functional component or base element of z/OS, that manages removable media.

DFSMSrmm control data set. See *control data set*.

DFSMSrmm-managed volume. A tape volume that is defined to DFSMSrmm. Contrast with *system-managed volume, DFSMShsm-managed volume*.

disaster recovery. A procedure for copying and storing an installation's essential business data in a secure location, and for recovering that data in the event of a catastrophic problem. Compare with *vital records*.

DISTANT. A DFSMSrmm built-in storage location ID. See *built-in storage location*.

| **DNS.** Domain Name System.

| **Domain Name System.** In the Internet suite of
| protocols, the distributed database system used to map
| domain names to IP addresses.

dual copy. A high availability function made possible by nonvolatile storage in some models of the IBM 3990 Storage Control. Dual copy maintains two functionally identical copies of designated DASD volumes in the logical 3990 subsystem, and automatically updates both copies every time a write operation is issued to the dual copy logical volume.

dump class. A set of characteristics that describes how volume dumps are managed by DFSMShsm.

duplexing. The process of writing two sets of identical records in order to create a second copy of data.

dynamic cache management. A function that automatically determines which data sets will be cached based on the 3990 subsystem load, the characteristics of the data set, and the performance requirements defined by the storage administrator.

E

EHPCT. Extended High Performance Cartridge Tape.

eject. The process used to remove a tape volume from a system-managed library. For an automated tape library dataserver, the volume is removed from its cell location and moved to the output station. For a manual tape library, the volume is not moved, but the tape configuration database is updated to show that the volume no longer resides in the manual tape library.

empty bin. A bin that can accept a volume.

Enhanced Capacity Cartridge System Tape. Cartridge system tape with increased capacity that can only be used with 3490E Magnetic Tape Subsystems. Contrast with *Cartridge System Tape*.

entry panel. See *data entry panel*.

EREP. Environmental Record Editing and Printing program.

expanded output. The output produced by the DFSMSrmm application programming interface when you specify OUTPUT=FIELDS and EXPAND=YES. For those subcommands for which expanded output applies, your application program receives more variable data than for standard output.

expiration. The process by which data sets and volumes are identified as available for reuse. In DFSMSrmm, all volumes have an expiration date or retention period set for them either by vital record specification policy, by user-specified JCL when writing a data set to the volume, or by an installation default. When a volume reaches its expiration date or retention period, it becomes eligible for release.

expiration date. The date at which a file is no longer protected against automatic deletion by the system.

expiration processing. The process of inventory management that ensures expired volumes are released and carries out required release actions on those volumes.

export. The operation to remove one or more logical volumes from a virtual tape server library. First, the list of logical volumes to export must be written on an export list volume and then, the export operation itself must be initiated.

exported logical volume. A logical volume that has gone through the export process and now resides on a stacked volume outside a virtual tape server library.

export list volume. A virtual tape server logical volume containing the list of logical volumes to export.

extended bin support. Enhanced options for managing shelf locations in a storage location including optimized use of the number of bins.

extended extract data set file. A data set created using the DFSMSrmm EDGJRPT exec. The records within the data set combine data set and volume information into single records.

extended record. A record in the DFSMSrmm extract data set that is mapped by the EDGXREXT mapping macro. The record contains both data set and volume information.

external label. A label attached to the outside of a tape cartridge that is to be stored in an IBM 3494 Tape Library Dataserver or IBM 3495 Tape Library Dataserver. The label might contain the DFSMSrmm rack number of the tape volume.

extract data set. A data set that you use to generate reports.

extract data set record. A record in an extract data set that is mapped by a DFSMSrmm mapping macro.

F

field format. Field format is where the output consists of Structured Field Introducers and variable data rather than output in line format.

filtering. The process of selecting data sets based on specified criteria. These criteria consist of fully or partially-qualified data set names or of certain data set characteristics.

FIPS. Federal Information Processing Standard.

FMID. Function modification identifier.

FRR. Functional recovery routines.

G

generation data group (GDG). A collection of data sets kept in chronological order. Each data set is a generation data set.

generation data set (GDS). One generation of a generation data group.

generation number. The number of a generation within a generation data group. A zero represents the most current generation of the group, a negative integer

(-1) represents an older generation and, a positive integer (+1) represents a new generation that has not yet been cataloged.

GDG. Generation data group.

GDS. Generation data set.

giga (G). The information-industry meaning depends upon the context:

1. G = 1 073 741 824(2^{30}) for real and virtual storage.
2. G = 1 000 000 000 for disk storage capacity (for example, a 4 GB fixed disk).
3. G = 1 000 000 000 for transmission rates.

global resource serialization (GRS). A component of z/OS used for serializing use of system resources and for converting hardware reserves on DASD volumes to data set enqueues.

GPR. General purpose register.

GRS. Global resource serialization.

grouping. When creating a report, grouping sorts report output contents into separate groups (and separate pages) based upon field contents.

guaranteed space. A storage class attribute indicating the space is to be preallocated when a data set is created. If you specify explicit volume serial numbers, SMS honors them. If space to satisfy the allocation is not available on the user-specified volumes, the allocation fails.

H

hardware configuration definition (HCD). An interactive interface in MVS that enables an installation to define hardware configurations from a single point of control.

HCD. Hardware configuration definition.

high-capacity input station. A transfer station, used by the operator to add tape cartridges to the IBM TotalStorage Enterprise Automated Tape Library (3494) or the IBM TotalStorage Enterprise Automated Tape Library (3495), which is inside the enclosure area.

high capacity output station. A transfer station, used by the operator to remove tape cartridges from the automated tape library dataserver, which is inside the enclosure area.

home. See *home location.*

home location. For DFSMSrmm, the place where DFSMSrmm normally returns a volume when the volume is no longer retained by vital records processing.

HPCT. High Performance Cartridge Tape.

I

ICETOOL. The DFSORT multipurpose data processing and reporting utility.

ID. Identifier.

IDRC. Improved data recording capability.

import. The operation to enter previously exported logical volumes residing on a stacked volume into a virtual tape server library. First, the list of logical volumes to import must be written on an import list volume and the stacked volumes must be entered, and then, the import operation itself must be initiated.

import list volume. A virtual tape server logical volume containing the list of logical volumes to import. This list can contain individual logical volumes to import and/or it can contain a list of stacked volumes in which all logical volumes on the stacked volume are imported.

imported logical volume. An exported logical volume that has gone through the import process and can be referenced as a tape volume within a virtual tape server library. An imported logical volume originates from a stacked volume that went through the export process.

improved data recording capability (IDRC). A recording mode that can increase the effective cartridge data capacity and the effective data rate when enabled and used. IDRC is always enabled on the 3490E Magnetic Tape Subsystem.

installation defined storage location. A storage location defined using the LOCDEF command in the EDGRMMxx parmlib member.

Interactive Storage Management Facility (ISMF). The interactive interface of DFSMS that allows users and storage administrators access to the storage management functions.

Interactive Problem Control System (IPCS). A system facility that allows interactive problem analysis.

Interactive System Productivity Facility (ISPF). An IBM licensed program used to develop, test, and run interactive, panel-driven dialogs.

internal label. The internal label for standard label tapes is recorded in the VOL1 header label,
| magnetically recorded on the tape media.

| **Internet Protocol (IP).** The TCP/IP layer between the
| higher-level host-to-host protocol and the local network
| protocols. IP uses local area network protocols to carry
| packets in the form of diagrams to the next gateway or
| destination host.

in transit. A volume state where a volume must be moved from one location to another and DFSMSrmm believes that the move has started, but has not yet received confirmation that the move is complete. For a volume moving from a system-managed library, the move starts when the volume is ejected.

inuse bin. A bin that is occupied by a volume and into which no volume can be assigned.

inventory management. The regular tasks that need to be performed to maintain the control data set. See also *expiration processing*, *storage location management processing*, and *vital record processing*.

| **IP address.** The unique 32-bit address that specifies the location of each device or workstation in the Internet. For example, 9.67.97.103 is an IP address.

IPCS. Interactive Problem Control System.

IPL. Initial program load.

ISPF. Interactive System Productivity Facility.

ISMF. Interactive Storage Management Facility.

ISO. International Organization for Standardization.

J

JCL. Job control language.

JES2. Job entry subsystem 2.

JES3. Job entry subsystem 3.

JFCB. Job file control block.

journal. A sequential data set that contains a chronological record of changes made to the DFSMSrmm control data set. You use the journal when you need to reconstruct the DFSMSrmm control data set.

K

keyword. A predefined word that is used as an identifier.

kilo (K). The information-industry meaning depends upon the context:

1. K = 1024(2^{10}) for real and virtual storage.
2. K = 1000 for disk storage capacity (for example, a 4 KB fixed disk).
3. K = 1000 for transmission rates.

L

Library Control System. The Object Access Method component that controls optical and tape library operations and maintains configuration information.

line format. Line format is where text and variable data are formatted into lines suitable for displaying at a terminal or printing as printed documentation.

LOCAL. A DFSMSrmm built-in storage location ID. See *built-in storage location*.

location name. A name given to a place for removable media that DFSMSrmm manages. A location name can be the name of a system-managed library, a storage location name, or the location *SHELF*, identifying shelf space outside a system-managed library or storage locations.

logical volume. Logical volumes have a many-to-one association with physical tape media and are used indirectly by MVS applications. They reside in a Virtual Tape Server or on exported stacked volumes. Applications can access the data on these volumes only when they reside in a Virtual Tape Server which makes the data available via its tape volume cache or after the data has been copied to a physical volume through the use of special utilities.

low-on-scratch management. The process by which DFSMSrmm replenishes scratch volumes in a system-managed library when it detects that there are not enough available scratch volumes.

LSR. Local shared resource.

M

management class. (1) A named collection of management attributes describing the retention and backup characteristics for a group of data sets, or for a group of objects in an object storage hierarchy. For objects, the described characteristics also include class transition. (2) In DFSMSrmm, if assigned by ACS routine to system-managed tape volumes, management class can be used to identify a DFSMSrmm vital record specification.

manual cartridge entry processing. The process by which a volume is added to the tape configuration database when it is added to a manual tape library. DFSMSrmm can initiate this process.

manual mode. An operational mode where DFSMSrmm runs without recording volume usage or validating volumes. The DFSMSrmm TSO commands, ISPF dialog, and inventory management functions are all available in manual mode.

manual tape library. An installation-defined set of stand-alone tape drives and the set of tape volumes that can be mounted on those drives.

master system. The MVS system where the master DFSMSrmm control data set resides.

master volume. A private volume that contains data that is available for write processing based on the DFSMSrmm EDGRMMxx parmlib MASTEROVERWRITE operand.

media format. The type of volume, recording format and techniques used to create the data on the volume.

media library. Removable media library.

media management system. A program that helps you manage removable media. DFSMSrmm is a media management system.

media name. An up to 8 character value that describes the shape or type of removable media stored in a storage location. Examples of media name are: SQUARE, ROUND, CARTRDGE, 3480.

media type. A value that specifies the volume's media type. Media type can be specified as *, CST, ECCST, HPCT, or EHPCT.

MEDIA 1. Cartridge system tape.

MEDIA 2. Enhanced capacity cartridge system tape.

MEDIA 3. High performance cartridge tape.

MEDIA 4. Extended high performance cartridge tape

MEDIA 5. IBM TotalStorage Enterprise Tape Cartridge.

| **MEDIA 6.** IBM TotalStorage Enterprise WORM Tape Cartridge.

| **MEDIA 7.** IBM TotalStorage Enterprise Economy Tape Cartridge.

| **MEDIA 8.** IBM TotalStorage Enterprise Economy WORM Tape Cartridge.

mega (M). The information-industry meaning depends upon the context:

1. M = 1 048 576(2^{20}) for real and virtual storage.
2. M = 1 000 000 for disk storage capacity (for example, a 4 MB fixed disk).
3. M = 1 000 000 for transmission rates.

migration. The process of moving unused data to lower cost storage in order to make space for high-availability data. If you wish to use the data set, it must be recalled. See also *migration level 1, migration level 2*.

migration level 1. DFSMShsm-owned DASD volumes that contain data sets migrated from primary storage volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage, migration level 2*.

migration level 2. DFSMShsm-owned tape or DASD volumes that contain data sets migrated from primary storage volumes or from migration level 1 volumes. The data can be compressed. See also *storage hierarchy*. Contrast with *primary storage, migration level 1*.

moving-in volume. A volume for which a move into a bin has been started, but not yet confirmed.

moving-out volume. A volume for which a move out of a bin has been started, but not yet confirmed.

MVS image. A single occurrence of the MVS/ESA operating system that has the ability to process work.

N

name vital record specification. A vital record specification used to define additional retention and movement policy information for data sets or volumes.

NEXTVRS. An RMM ADDVRS TSO subcommand operand. See also *Using Next*.

NL. No label.

nonscratch volume. A volume that is not scratch, which means it has valid or unexpired data on it. Contrast with *scratch*.

NSL. Nonstandard label.

O

OAM. Object access method.

object. A named byte stream having no specific format or record orientation.

object access method (OAM). An access method that provides storage, retrieval, and storage hierarchy management for objects and provides storage and retrieval management for tape volumes contained in system-managed libraries.

OPC/ESA. Operations Planning and Control/Enterprise Systems Architecture.

optical volume. Storage space on an optical disk, identified by a volume label. See also *volume*.

optical disk. A disk that uses laser technology for data storage and retrieval.

option line. Command line.

owner. In DFSMSrmm, a person or group of persons defined as a DFSMSrmm user owning volumes. An owner is defined to DFSMSrmm through an owner ID.

owner ID. In DFSMSrmm, an identifier for DFSMSrmm users who own volumes.

P

parallel. During conversion, when you install DFSMSrmm concurrently with an existing media management system, it is called running in parallel.

partitioned data set (PDS). A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

PDS. Partitioned data set.

permanent data set. A user-named data set that is normally retained for longer than the duration of a job or interactive session. Contrast with *temporary data set*.

PF. Program function key.

physical stacked volume. See *stacked volume*.

physical volume. A volume that has a one-to-one association with physical tape media and which is used directly by MVS applications. It may reside in an automated tape library dataserver or be kept on shelf storage either at vault sites or within the data center where it can be mounted on stand-alone tape drives.

pool. A group of shelf locations in the removable media library whose rack numbers share a common prefix. The shelf locations are logically grouped so that the volumes stored there are easier to find and use.

pool ID. The identifier for a pool. You define pool IDs in parmlib member EDGRMMxx.

pooling. The process of arranging shelf locations in the removable media library into logical groups.

pool storage group. A type of storage group that contains system-managed DASD volumes. Pool storage groups allow groups of volumes to be managed as a
| single entity. See also *storage group*.

| **port.** (1) An access point for data entry or exit. (2) A
| receptacle on a device to which a cable for another
| device is attached.

primary space allocation. Amount of space requested by a user for a data set when it is created. Contrast with *secondary space allocation*.

primary storage. A DASD volume available to users for data allocation. The volumes in primary storage are called primary volumes. See also *storage hierarchy*. Contrast with *migration level 1, migration level 2*.

primary vital record specification. The first retention and movement policy that DFSMSrmm matches to a data set and volume used for disaster recovery and vital record purposes. See also vital record specification and secondary vital record specification.

private tape volume. A volume assigned to specific individuals or functions.

protect mode. In protect mode, DFSMSrmm validates all volume requests.

pseudo-generation data group. A collection of data sets, using the same data set name pattern, to be managed like a generation data group. The ¬ masking character is used in DFSMSrmm to identify the characters in the pattern that change with each generation.

PSW. Program status word.

PTF. Program temporary fix.

pull list. A list of scratch volumes to be pulled from the library for use.

PUT. Program update tape.

R

RACF. Resource Access Control Facility.

rack number. A six-character identifier that corresponds to a specific volume's shelf location in the installation's removable media library, and is the identifier used on the external label of the volume to identify it. The rack number identifies the pool and the external volume serial number for a volume residing in an automated tape library dataserver. The rack number identifies the pool, the external volume serial, and shelf location number for a volume not residing in an automated tape library dataserver. The rack number is not written by the tape drive. It exists as an entry in the DFSMSrmm control data set and on the external label of the tape. See also *shelf location.*

rack pool. A group of shelves that contains volumes that are generally read-only.

ready to scratch. This describes the condition where a volume is eligible for scratch processing while it resides in a storage location. Since no other release actions are required, the volume can be returned to scratch directly from the storage location.

recording format. For a tape volume, the format of the data on the tape; for example, 18 tracks or 36 tracks.

record-only mode. The operating mode where DFSMSrmm records information about volumes as you use them, but does not validate or reject volumes.

recovery. The process of rebuilding data after it has been damaged or destroyed, often by using a backup copy of the data or by reapplying transactions recorded in a journal.

relative start generation. Relative start generation zero is the latest generation of a tape. Relative start generation -1 is the previous generation of that tape. Relative start generation -2 is the generation before the previous one.

REMOTE. A DFSMSrmm built-in storage location ID. See also *built-in storage location*.

removable media. See also *volume*.

removable media library. The volumes that are available for immediate use, and the shelves where they could reside.

report. Data that has been selected and extracted according to the reporting tool, the type of report desired, and the formatting criteria.

reporting tool. A REXX exec that builds control statements to enable you to create reports using a reporting utility.

report type. A data source and how it is mapped.

Resource Access Control Facility (RACF). An IBM licensed program that provides for access control by identifying and verifying the users to the system; authorizing access to protected resources; logging the detected unauthorized attempts to enter the system; and logging the detected accesses to protected resources.

Resource Group. A collection of structured fields that describe the attributes of a resource such as a volume.

Restructured Extended Executor (REXX) Language. A general-purpose, high-level programming language, particularly suitable for EXEC procedures or programs for personal computing.

retention date. Retention date can be the date that a data set or volume is retained by a vital record specification or the date of the inventory management run when the data set or volume is no longer retained by a vital record specification.

retention period. The time for which DFSMSrmm retains a volume or data set before considering it for release. You can retain a data set or volume as part of disaster recovery or vital records management. You set a retention period through a vital record specification that overrides a data set's expiration date.

retention type. The types of retention for which DFSMSrmm retains a volume or data set before considering it for release. The retention types for data sets are BYDAYSCYCLE, CYCLES, DAYS, EXTRADAYS, LASTREFERENCEDAYS, UNTILEXPIRED, and WHILECATALOG. The retention types for volumes are DAYS and CYCLE.

REXX. Restructured Extended Executor Language.

RMF. Resource Measurement Facility.

RMM complex (RMMplex). One or more MVS images that share a common DFSMSrmm control data set.

RMODE. Residence mode.

S

SAF. System Authorization Facility.

scratch. The status of a tape volume that is available for general use, because the data on it is incorrect or is no longer needed. You request a scratch volume when you omit the volume serial number on a request for a tape volume mount.

scratch pool. The collection of tape volumes from which requests for scratch tapes can be satisfied. Contrast with *rack pool*.

scratch processing. The process for returning a volume to scratch status once it is no longer in use and has no outstanding release actions pending.

scratch tape. See *scratch volume*.

scratch volume. A tape volume that contains expired data only. See *scratch*.

SDB. Structured database.

SDSF. Spool display and search facility.

secondary space allocation. Amount of additional space requested by the user for a data set when primary space is full. Contrast with *primary space allocation*.

secondary vital record specification. The second retention and movement policy that DFSMSrmm matches to a data set and volume used for disaster recovery and vital records purposes. See also vital record specification and primary vital record
| specification.

| **server.** (1) A functional unit that provides shared
| services to workstations over a network; for example, a
| file server, a print server, a mail server. (2) On a
| network, the computer that contains the data or
| provides the facilities to be accessed by other
| computers in the network. (3) A program that handles
| protocol, queuing, routing, and other tasks necessary for
| data transfer between devices in a computer system.

SFI. Structured field introducer.

shelf. A place for storing removable media, such as tape and optical volumes, when they are not being written to or read.

shelf location. A single space on a shelf for storage of removable media. DFSMSrmm defines a shelf location in the removable media library by a rack number, and a shelf location in a storage location by a bin number. See also *rack number* and *bin number.*

shelf-management. Is the function provided to manage the placement of volumes in individual slots in a location. Shelf-management is provided for the removable media library using rack numbers. For storage locations it is optional as defined by the LOCDEF options in parmlib and uses bin numbers.

shelf-resident volume. A volume that resides in a non-system-managed tape library.

shelf space. See *shelf*.

SL. Standard label.

slot. See *shelf location*.

SMF. System management facility.

SMP/E. System Modification Program Extended.

SSI. Subsystem interface.

stacked volume. A volume that has a one-to-one association with physical tape media and which is used in a virtual tape server to store logical volumes. A stacked volume is not used by MVS applications but by the virtual tape server and its associated utilities. It may be removed from a virtual tape server to allow transportation of logical volumes to a vault or to another virtual tape server.

standard label. An IBM standard tape label.

standard output. The output produced by the DFSMSrmm application programming interface when you specify OUTPUT=LINES or EXPAND=NO with OUTPUT=FIELDS.

storage administrator. A person in the data processing center who is responsible for defining, implementing, and maintaining storage management policies.

storage class. A collection of storage attributes that identify performance goals and availability requirements, defined by the storage administrator, used to select a device that can meet those goals and requirements.

storage group. A collection of storage volumes and attributes, defined by the storage administrator. The collections can be a group of DASD volumes or tape volumes, or a group of DASD volumes and optical volumes treated as a single object storage hierarchy.

storage location. A location physically separate from the removable media library where volumes are stored for disaster recovery, backup, and vital records management.

(storage) location dominance. The priority used by DFSMSrmm to decide where to move a volume within the removable media library during vital record specification processing. It covers all the locations; SHELF, storage locations, and system-managed tape libraries.

storage location management processing. The process of inventory management that assigns a shelf location to volumes that have moved as a result of vital record processing. See also *vital record processing.*

stripe. In DFSMS, the portion of a striped data set, such as an extended format data set, that resides on one volume. The records in that portion are not always logically consecutive. The system distributes records among the stripes such that the volumes can be read from or written to simultaneously to gain better performance. Whether it is striped is not apparent to the application program.

striping. A software implementation of a disk array that distributes a data set across multiple volumes to improve performance.

structured field. Output from the DFSMSrmm application programming interface consisting of a Structured Field Introducer and output data.

structured field introducer (SFI). An 8-byte entity that either introduces the beginning of a group of data or introduces output data that immediately follows the introducer.

subsystem. A special MVS task that provides services and functions to other MVS users. Requests for service are made to the subsystem through a standard MVS facility known as the subsystem interface (SSI). Standard MVS subsystems are the master subsystem and the job entry subsystems JES2 and JES3.

subsystem interface (SSI). The means by which system routines request services of the master subsystem, a job entry subsystem, or other subsystems defined to the subsystem interface.

SUL. IBM standard and user header or trailer label.

SVC. Supervisor call.

system-managed storage. Storage managed by the Storage Management Subsystem. SMS attempts to deliver required services for availability, performance, and space to applications. See also *system-managed storage environment*.

DFSMS environment. An environment that helps automate and centralize the management of storage.

This is achieved through a combination of hardware, software, and policies. In the DFSMS environment for MVS, this function is provided by DFSMS, DFSORT, and RACF. See also *system-managed storage*.

system-managed tape library. A collection of tape volumes and tape devices, defined in the tape configuration database. A system-managed tape library can be automated or manual. See also *tape library.*

system-managed volume. A DASD, optical, or tape volume that belongs to a storage group. Contrast with *DFSMShsm-managed volume, DFSMSrmm-managed volume*.

system programmer. A programmer who plans, generates, maintains, extends, and controls the use of an operating system and applications with the aim of improving overall productivity of an installation.

T

tape configuration database (TCDB). One or more volume catalogs used to maintain records of system-managed tape libraries and tape volumes.

tape librarian. The person who manages the tape library. This person is a specialized storage administrator.

tape library. A set of equipment and facilities that support an installation's tape environment. This can include tape storage racks, a set of tape drives, and a set of related tape volumes mounted on those drives. See also *system-managed tape library, automated tape library data server*.

Tape Library Control System (TLCS). IBM program offering 5785-EAW. DFSMSrmm replaces TLCS.

tape library dataserver. A hardware device that maintains the tape inventory that is associated with a set of tape drives. An automated tape library dataserver also manages the mounting, removal, and storage of tapes. An automated tape library dataserver that supports system-managed storage of tape volumes. The IBM automated tape library dataservers include the IBM 3494 Tape Library Dataserver and the IBM 3495 Tape Library Dataserver.

tape storage group. A type of storage group that contains system-managed private tape volumes. The tape storage group definition specifies the system-managed tape libraries that can contain tape volumes. See also *storage group*.

tape subsystem. A magnetic tape subsystem consisting of a controller and devices, which allows for the storage of user data on tape cartridges. Examples of tape subsystems include the IBM 3490 and 3490E Magnetic Tape Subsystems.

tape volume. A tape volume is the recording space on a single tape cartridge or reel. See also *volume*.

TCDB. Tape configuration database.

temporary data set. An uncataloged data set whose name begins with & or &&, that is normally used only for the duration of a job or interactive session. Contrast with *permanent data set*.

tera (T). The information-industry meaning depends upon the context:

1. T = 1 099 511 627 776(2^{40}) for real and virtual storage.
2. T = 1 000 000 000 000 for disk storage capacity (for example, 4 TB of DASD storage).
3. T = 1 000 000 000 000 for transmission rates.

TLCS. Tape Library Control System.

Transmission Control Protocol (TCP). A stream communication protocol that includes error recovery and flow control.

Transmission Control Protocol/Internet Protocol (TCP/IP). The two fundamental protocols of the Internet protocol suite. The abbreviation TCP/IP is frequently used to refer to this protocol suite. TCP/IP provides for the reliable transfer of data, while IP transmits the data through the network in the form of datagrams. Users can send mail, transfer files across the network, or execute commands on other systems.

TSO. Time Sharing Option.

U

Until Expired. Allows the use of vital record specification policies for managing retention in a location as long as the volume expiration date has not been reached.

use attribute. (1) The attribute assigned to a DAD volume that controls when the volume can be used to allocate new data sets; use attributes are *public*, *private*, and *storage*. (2) For system-managed tape volumes, use attributes are *scratch* and *private*.

user volume. A volume assigned to a user, that can contain any data and can be rewritten as many times as the user wishes until the volume expires.

using AND. A method for linking DFSMSrmm vital record specifications to create chains of vital record specifications. DFSMSrmm applies policies in chains using AND only when all the retention criteria are true.

using NEXT. A method for linking DFSMSrmm vital record specifications to create chains of vital record specifications. DFSMSrmm applies policies in chains using NEXT one vital record at a time.

V

virtual export. A method of exporting a volume by marking a volume as exported by using the DFSMSrmm subcommands.

virtual input/output (VIO) storage group. A type of storage group that allocates data sets to paging storage, which simulates a DASD volume. VIO storage groups do not contain any actual DASD volumes. See also *storage group*.

virtual tape server (VTS). This subsystem, integrated into the IBM TotalStorage Enterprise Automated Tape Library (3494) or the IBM TotalStorage Enterprise Automated Tape Library (3495), combines the random access and high performance characteristics of DASD with outboard hierarchical storage management and virtual tape devices and tape volumes.

vital record group. A set of data sets with the same name that matches to the same DFSMSrmm vital record specification.

vital record processing. The process of inventory management that determines which data sets and volumes DFSMSrmm should retain and whether a volume needs to move. These volumes and data sets have been assigned a vital record specification.

vital records. A data set or volume maintained for meeting an externally-imposed retention requirement, such as a legal requirement. Compare with *disaster recovery*.

vital record specification. Policies defined to manage the retention and movement of data sets and volumes used for disaster recovery and vital records purposes.

vital record specification management value. A one-to-eight character name defined by your installation and used to assign management and retention values to tape data sets. The vital record management value can be any value you chose to create a match between a vital record specification and data sets and volumes in your installation. By matching the vital record specifications to the data set or volumes, DFSMSrmm applies the retention and movement policies you define in the vital record specifications. During inventory management VRSEL processing, DFSMSrmm selects the correct, best matching vital record specification for a tape data set or volume.

VOLSER. Volume serial number.

volume. The storage space on DASD, tape, or optical devices, which is identified by a volume label. See also *DASD volume*, *logical volume*, *optical volume*, *stacked volume*, and *tape volume*.

volume catalog. See *tape configuration database*.

volume expiration date. The date the volume should expire based on the highest expiration date of the data sets that reside on the volume.

volume serial number (VOLSER). (1) An identification number in a volume label that is assigned when a volume is prepared for use on the system. For standard label volumes, the volume serial number is the VOL1 label of the volume. For no label volumes, the volume serial number is the name the user assigns to the volume. (2) In DFSMSrmm, volume serial numbers do not have to match rack numbers.

VTS. Virtual tape server.

W

warning mode. The operating mode in which DFSMSrmm validates volumes as you use them, but issues warning messages when it discovers errors instead of rejecting volumes.

| **world-wide identifier (WWID).** Often used in z/OS software as the abbreviation for the world-wide unique cartridge identifier (WWCID). See also *world-wide unique cartridge identifier*

| **world-wide unique cartridge identifier (WWCID).** A permanent identifier associated with a specific tape cartridge, typically stored on the tape itself and the non-volatile cartridge memory.

| **Write Once, Read Many (WORM).** A technology to allow data to be written once to storage media. After that, data is permanent and cannot be altered, but can be read any number of times.

write-to-operator (WTO). An optional user-coded service that allows a message to be written to the system console operator informing the operator of errors and unusual system conditions that may need to be corrected.

WTO. Write-to-operator.

| **WWCID.** See world-wide unique cartridge identifier.

| **WWID.** See world-wide identifier.

Index

Readers' Comments — We'd Like to Hear from You

z/OS
DFSMSrmm Implementation
and Customization Guide

Publication No. SC26-7405-04

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SC26-7405-04

Cut or Fold
Along Line

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

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400

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

SC26-7405-04

Cut or Fold
Along Line

Program Number: 5694-A01

Printed in USA

SC26-7405-04