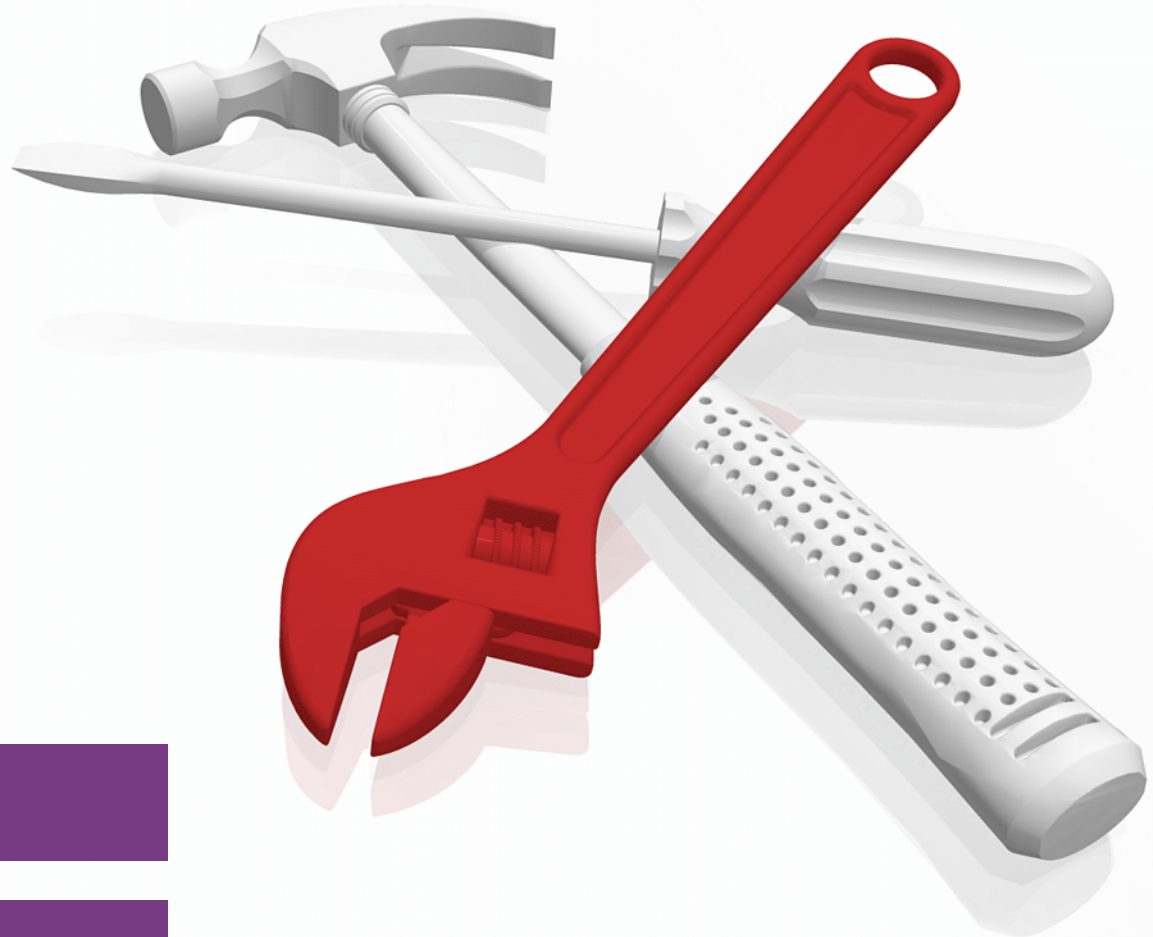


# What is New in DFSMSrmm

Ryan Bouchard  
Larry Coyne  
Parker Mathewson  
Michael Scott  
Samuel Smith  
Karim Walji



**IBM Z**

**Storage**





IBM Redbooks

**What is New in DFSMSrmm**

December 2023

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.

w

**Second Edition (December 2023)**

This edition applies to Version 2, Release 5, Modification 0 of z/OS DFSMSrmm (product number 5650-ZOS).

**© Copyright International Business Machines Corporation 2022,2023. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
Authors .....	ix
Now you can become a published author, too! .....	x
Comments welcome .....	x
Stay connected to IBM Redbooks .....	xi
<b>Summary of changes</b> .....	xiii
December 2023, Second Edition .....	xiii
<b>Chapter 1. External Data Manager function</b> .....	1
1.1 Introduction to the External Data Manager function .....	2
1.2 TSO subcommand and ISPF panel support for EDM .....	3
1.2.1 REXX variable and Extract support for EDM .....	3
1.2.2 EDM-related APARs .....	4
<b>Chapter 2. Retention methods in DFSMSrmm and the use of the expiration date</b> .....	5
2.1 Comparison of Vital Record Selection and Expiration Date .....	6
2.2 Expiration date setting .....	7
2.2.1 Order of precedence when setting EXPDT and RETPD .....	7
2.2.2 WHILECATALOG processing in the EXPDT retention method .....	9
2.2.3 EDGRMMxx parmlib options for EXPDT processing .....	10
<b>Chapter 3. Introduction to the Defaults Table</b> .....	13
3.1 Overview .....	14
3.2 Defaults Table command operands .....	15
3.2.1 RETENTIONMETHOD .....	15
3.2.2 VRSELEXCLUDE .....	16
3.2.3 EDM and NOEDM .....	16
3.2.4 RETPD .....	16
3.2.5 VRSVALUE .....	17
3.2.6 RETAINBY .....	17
3.2.7 WHILECATALOG .....	18
3.2.8 LASTREF(extra_days) .....	18
3.3 EDGUX100 EXIT .....	19
<b>Chapter 4. Using SMS Management Class to assign tape expiration values</b> .....	21
4.1 Overview .....	22
4.2 Expiration attributes .....	22
4.2.1 MCATTR option of parmlib member EDGRMMxx .....	22
4.2.2 Changing attributes .....	24
4.2.3 Management class attributes .....	24
4.3 Data set management .....	26
<b>Chapter 5. Logical Write Once Read Many changes</b> .....	27
5.1 Overview .....	28
5.2 Adjusting a volume's expiration date .....	29
5.3 Managing data sets .....	29

5.3.1 Mirroring retention settings . . . . .	29
5.3.2 Using the WHILECATALOG function . . . . .	31
5.4 Keeping LWORM volumes permanently . . . . .	32
<b>Chapter 6. General Data Protection Regulation changes . . . . .</b>	<b>33</b>
6.1 Overview . . . . .	34
6.2 DSNEXPIRE command option . . . . .	34
6.3 FORCEEXPIRE command . . . . .	35
<b>Chapter 7. z/OSMF GUI plug-in for DFSMSrmm . . . . .</b>	<b>37</b>
7.1 Overview . . . . .	38
7.2 Default Table Policy Helper . . . . .	39
7.3 Screen Settings . . . . .	40
7.4 Translations . . . . .	40
<b>Chapter 8. ONLYIF system option for EDGRMMxx . . . . .</b>	<b>41</b>
8.1 Overview . . . . .	42
8.2 Sharing RMM parmlib members . . . . .	42
8.3 DEFAULT TABLE support . . . . .	44
8.4 Backup procedure example . . . . .	45
8.5 CATSYSID example . . . . .	46
<b>Chapter 9. CDSFULL command for EDGRMMxx . . . . .</b>	<b>47</b>
<b>Chapter 10. Global Resource Serialization recommendations for RMM data sets . . . . .</b>	<b>49</b>
10.1 RMM control data sets and journals . . . . .	50
10.2 GRS Settings . . . . .	50
<b>Chapter 11. Disaster Recovery and Disaster Recovery testing with the TS7700 . . . . .</b>	<b>53</b>
11.1 Overview . . . . .	54
11.2 DFSMSrmm parmlib options . . . . .	54
11.3 Volser range example . . . . .	54
11.4 Additional safeguards . . . . .	57
11.5 Handling scratch volumes for DR testing . . . . .	57
11.6 Reading DR volumes back on PROD . . . . .	62
11.7 Reading LWORM volumes during DR testing . . . . .	62
11.8 Limited Housekeeping function . . . . .	63
<b>Chapter 12. ePartitioning examples with OPENRULE and PRTITION parameters . . . . .</b>	<b>65</b>
12.1 Separating a TS7700 library between two LPARs or SYSPLEXES . . . . .	66
12.2 Customizing OPENRULE statements to allow cross LPAR read of volsers . . . . .	67
<b>Chapter 13. Expire Hold setting and available scratch count . . . . .</b>	<b>69</b>
TS7700 EXPIRE HOLD overview . . . . .	70
<b>Chapter 14. Common problems in DFSMSrmm . . . . .</b>	<b>73</b>
14.1 Reasons why a volume does not go to SCRATCH . . . . .	74
14.2 Long-running EDGHSKP jobs . . . . .	75
14.2.1 VRSEL . . . . .	75
14.2.2 EXPROC . . . . .	75
14.2.3 CATSYNCH . . . . .	75
<b>Chapter 15. DFSORT JCL for use with DFSMSrmm . . . . .</b>	<b>77</b>
15.1 Introduction . . . . .	78
15.2 Overview of DFSMSrmm data . . . . .	79

15.3 Example use cases . . . . .	81
15.3.1 Use case 1: Reporting tape volume usage . . . . .	81
15.3.2 Use Case 2: Rebuilding data set records for recovered volume . . . . .	84
15.3.3 Use Case 3: Data sets are assigned an incorrect management value and that value must be removed . . . . .	86
15.3.4 Use Case 4: Comparing EDGUTIL output with Extract. . . . .	88
15.3.5 Additional use case information . . . . .	92
15.3.6 Use Case 5: Generating DEFINE NONVSAM commands for recovered tape data sets . . . . .	93
<b>Appendix A. Using EDGRMMxx global defaults for assignment of retention attributes     case study . . . . .</b>	<b>103</b>
<b>Appendix B. Using Defaults Table for expiration attributes assignment case study</b>	<b>109</b>
<b>Appendix C. Using Management Class for assignment of expiration attributes case     study . . . . .</b>	<b>117</b>
<b>Appendix D. Using mixed retention methods VRSEL and EXPDT case study . . . . .</b>	<b>123</b>
<b>Appendix E. Using Management Class and Defaults Table together case study . . . . .</b>	<b>139</b>
<b>Appendix F. Using MCATTR(ALL) instead of MCATTR(VRSELXDI) case study . . . . .</b>	<b>147</b>
<b>Appendix G. Key takeaways from the case studies. . . . .</b>	<b>155</b>
<b>Related publications . . . . .</b>	<b>157</b>
IBM Redbooks . . . . .	157
Other publications . . . . .	157
Help from IBM . . . . .	157
Stay connected to IBM Redbooks . . . . .	157





# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

# Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

IBM®


IBM Spectrum®

IBM Z®

OMEGAMON®

RACF®

Redbooks®

Redbooks (logo) ®

z/OS®

The following terms are trademarks of other companies:

Other company, product, or service names may be trademarks or service marks of others.

# Preface

DFSMSrmm is an IBM® z/OS® feature that is a fully functioning tape management system to manage your removable media. In the last decade, many enhancements were made to DFSMSrmm. This IBM Redbooks publication is intended to help you configure and use the newer functions and features that are now available.

Discussion of the new features is included along with use cases. Hints and tips of various common DFSMSrmm problems and useful configuration and reporting JCL are also included.

This publication is intended as a supplement to *DFSMSrmm Primer*, SG24-5983, which is still the recommended starting point for any users new to DFSMSrmm.

## Authors

This book was produced by a team of specialists from around the world working with the IBM Redbooks, Tucson Center.

**Ryan Bouchard** is a Staff DFSMS and IBM z/OS Technical Support Engineer in the IBM Systems, Client Enablement, and Systems Assurance team. He earned his Bachelor of Sciences from the Department of Systems and Industrial Engineering in the College of Engineering at the University of Arizona, and has a minor in Electrical and Computer Engineering. Ryan has a patent that is issued in the computer sciences field.

**Larry Coyne** is a Project Leader at the IBM International Technical Support Organization, Tucson, Arizona, center. He has over 35 years of IBM experience, with 23 years in IBM storage software management. He holds degrees in Software Engineering from the University of Texas at El Paso and Project Management from George Washington University. His areas of expertise include client relationship management, quality assurance, development management, and support management for IBM storage management software.

**Parker Mathewson** is a Software Engineer in the DFSMS group of IBM. With over 7 years of experience on front-end and back-end technologies, he brings different aspects of experience to the z/OS mainframe teams. Parker holds a Bachelors of Science in Computer Science from University of Arizona. He has a passion work web technologies and has been working in the modernization efforts within DFSMS.

**Michael Scott** is a Senior Data Facility Storage Management Subsystem (DFSMS) Technical Support Engineer in the IBM Systems, Client Enablement, and Systems Assurance team. He has 26 years of experience in DFSMS technical support. He holds a Masters in Business Administration and a Bachelor of Sciences in Mathematics. Michael has 14 patents that were issued in the computer sciences field. He is also a DFSMS Technical Advocate and has traveled worldwide to teach debugging classes for IBM. Michael has also been a part of the TS7700 IBM Redbooks® team for the last nine years.

**Samuel Smith** is a Senior Staff Software Engineer with IBM Systems. Samuel has 20 years of experience in support of the Mainframe, DFSMS, and IBM z/OS. Samuel's expertise is spread across the DFSMSrmm, DFSORT, and EREP components. Samuel holds a Bachelor's degree in Computer Information Systems, a Minor degree in Business Management, and a Minor degree in Psychology. Samuel also earned a Certified IBM System Z Professional certificate from Marist College (an IBM Partner). Samuel reached an invention

plateau with three patents and three publications in the computer sciences field. Samuel has taken international assignments to some of the world's largest banks to assist in understanding DFSORT diagnostics so that systems can be tuned to maximize performance.

**Karim Walji** is a Senior DFSMS Technical Support Engineer in the IBM Systems, Client Enablement, and Systems Assurance team. He has 30 years of experience in DFSMS technical support. He has degrees in Computer Science and Business Administration.

Thanks to the following people for their contributions to this project:

Erika Dawson  
Sri Hari Kolusu  
**IBM Systems**

Thanks to Norbert Schlumberger.

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, IBM Redbooks  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on LinkedIn:  
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:  
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:  
<http://www.redbooks.ibm.com/rss.html>



# Summary of changes

This section describes the technical changes that are made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

## December 2023, Second Edition

This revision includes the following new and changed information.

### **New information**

- ▶ Added “Reading LWORM volumes during DR testing” on page 62
- ▶ Added “Limited Housekeeping function” on page 63
- ▶ Added “NODATE matching parameter for Defaults Table”. See 3.1, “Overview” on page 14
- ▶ Added 15.3.6, “Use Case 5: Generating DEFINE NONVSAM commands for recovered tape data sets” on page 93

### **Changed information**

- ▶ Minor updates and updated product links







# External Data Manager function

This chapter discusses the External Data Manager support in DFSMSrmm and includes the following topics:

- ▶ 1.1, “Introduction to the External Data Manager function” on page 2
- ▶ 1.2, “TSO subcommand and ISPF panel support for EDM” on page 3

## 1.1 Introduction to the External Data Manager function

Tapes that are created by programs that provide their own tape management are controlled by an external data manager (EDM) and are referred to as EDM-managed.

In V2R1, DFSMSrmm introduced EDM-managed support in APAR [OA52541](#). Example of EDM programs are DFSMSHsm (HSM) and DFSMSdfp Object Access Method (OAM). This support was created to prevent EDM-managed tapes from being released prematurely because of an incorrect specification of retention criteria or abends that can occur during processing. In V2R1, EDM programs can maintain management of their own tape inventory and control when their tapes expire.

The EDM-managed support is a volume-level attribute and indicates that the volume is to be marked for EDM management. EDM-managed support prevents the volume from being set as expired by RMM, even after the volume leaves retention. EDM programs such as HSM, OAM, and IBM Spectrum® Protect must specifically release a volume by using the EDGTVEXT or EDGDFHSM interfaces.

Other programs can use this feature by issuing commands to set EDM by using the RMM API, Program Name mask in the Defaults Table, or by issuing RMM TSO subcommands to set the EDM attribute flag. The EDM attribute flag is cleared when the tape is no longer needed.

At V2R3 and beyond, this function defaults to EDM(YES), even if it is not specified in the EDGRMMxx parmlib. The decision was made that defaulting this option to YES protects accidental loss of data. The EDM attribute is automatically applied under EDM(YES) when a data set is created by programs ANZSRVR, ANRSERV, ARCCTL, ARCWCTL, CBROAM, or DSMSERV and the expiration date that is specified is 1999/365, 1999/366, or no expiration date is specified. The attribute is automatically removed when the volume is released by way of the EDGTVEXT or EDGDFHSM interface and the volume is then allowed to expire normally.

If HSM has SETSYS TAPESECURITY(EXPIRATION), HSM passes the PERMANENT(99365) expiration date for migration tapes, backup tapes, and Aggregate backup and recovery support (ABARS) tapes. For DUMP tapes, HSM passes the RETPD from DUMPCLAS definition.

IBM Resource Access Control Facility (RACF®) is an option with the format SETSYS TAPESECURITY(RACF). If the RACF option is used, HSM does not set the expiration date on the tape for migration and backup tapes. Instead, default retention is used from EDGRMMxx parmlib member RETPD.

**Note:** TAPESECURITY(PASSWORD) is *not* supported in the SMS Tape Library environment, including libraries that are defined as Manual Tape Libraries (MTL).

For both TAPESECURITY settings, if EDM is turned on in EDGRMMxx parmlib, EDM=YES is set to retain the tapes in RMM until such time that HSM releases the tapes by way of EDGTVEXT/EDGDFHSM interface. Then, the tapes are placed in PENDING RELEASE and the EDM flag is set to NO.

HSM migration and backup tapes are released by way of RECYCLE. DUMP tapes are released by way of the DUMP CLEANUP process that runs during automatic dumps and ABARS by using the EXPIREBV ABARS command.

With the EDM(YES) option that is specified in EDGRMMxx parmlib, it is no longer required to manage HSM and OAM volumes with a vital record specification (VRS). Removing these rules reduces CPU consumption and run times of the EDGHSKP step that performs VRSEL processing.

## 1.2 TSO subcommand and ISPF panel support for EDM

TSO subcommand support for the RMM CHANGEVOLUME(RMM CV) command includes the ability to specify EDM or NOEDM to turn on and off the EDM attribute. Authorization requires CONTROL access to STGADMIN.EDG.MASTER.

The EDM operand is valid for nonscratch, nonpending release volumes only. The RETAINBY(SET) option does not consider the EDM attribute; instead, each volume must be set to EDM managed.

Resetting the EDM attribute for a volume with the RMM CV volser NOEDM operand when it is still under control of an EDM might cause data loss because DFSMSrmm can release the volume by using normal expiration policies or abend earlier than expected.

The recommendation is to allow eligible EDM programs to turn on and off the EDM attribute automatically and to use the RMM CHANGEVOLUME command only in specific cases where recovery or manual intervention is needed. This support is available at V2R4 and higher.

TSO subcommand support for the RMM SEARCHVOLUME(RMM SV) command also includes the ability to create a list of volumes that feature the EDM attribute that is specified or do not have the EDM attribute specified(NOEDM). Authorization requires READ access to STGADMIN.EDG.MASTER.

In addition, if COMMANDAUTH(DSN) is in effect, you need READ access to the first file data set name in the DATASET class. READ access is required if no first file is defined to DFSMSrmm, the volume is in master status or user status, and if TAPEVOL resource class is active.

The RMM ISPF panel for Search Volume includes the EDM attribute. This attribute can be turned on or off by using the Change Volume Details panel.

### 1.2.1 REXX variable and Extract support for EDM

In V2R4, support was added to include the EDM field in the EXTRACT records. For the extended records, a one-character field exists that is named XVEDM that features a value of Y(Yes) or N(No) when browsing the file at location 1769 decimal. In the nonextended records, the field RVEDM exists that features Y or N at location 1105 decimal.

In V2R4, a TSO subcommand EDM variable that can be used in REXX EDG@EDM was added for the LISTVOLUME VOL subcommand. Previously, this variable was available only for LISTCONTROL OPTION. EDG@DEFM also was added in V2R4 for LC DEFTABLE.

## 1.2.2 EDM-related APARs

The following EDM-related APARs are available:

- ▶ [OA52541](#): OA5541 NEW FUNCTION – EDM VOLUME ATTRIBUTE SUPPORT (V2R1, V2R2)
- ▶ [OA51654](#): TOLERATION (CO-EXISTENCE) FOR FUTURE NEW FUNCTION (V2R1, V2R2)
- ▶ [OA54827](#): EDM FLAG NOT TURNED OFF WHEN HSM RECYCLES A VOLUME (V2R1, V2R2)
- ▶ [OA53134](#): ADDITIONAL CHANGES REQUIRED FOR EDM SUPPORT
- ▶ [OA56417](#): DEFAULT BEHAVIOUR WHILECATALOG(UNTILEXPIRED) MAY EXPIRE UNCATALOGED DATASETS PREMATURELY
- ▶ [OA56852](#): NEW FUNCTION: SEARCH SUPPORT FOR EDM AND FOR THE GDPR RELATED OPTIONS FORCEEXPIRE AND DSNEXPIRE(BLOCK) (V2R2, V2R3, V2R4)



# Retention methods in DFSMSrmm and the use of the expiration date

This chapter discusses the two retention methods that are available in DFSMSrmm and how the expiration date is used with each of them. It includes the following topics:

- ▶ 2.1, “Comparison of Vital Record Selection and Expiration Date” on page 6
- ▶ 2.2, “Expiration date setting” on page 7

## 2.1 Comparison of Vital Record Selection and Expiration Date

Two retention methods are available for use with DFSMSrmm, Vital Record Selection (VRSEL), and Expiration Date (EXPDT). The VRSEL retention method uses a collection of vital record specifications (VRS) to calculate a retention date for each data set that is managed by DFSMSrmm.

When the VRSEL function of inventory management processing is run, each data set in the RMM control data set (CDS) that is contained on a volume that is specified as VRSEL managed is analyzed and a retention date is calculated. The longest retention date and expiration date and time of all the data sets on a volume define the retention and expiration date and time for the volume.

After a data set leaves vital record status, the retention date is cleared, and the expiration date then becomes the date on which the data set is eligible for expiration. Unless the expiry date ignore setting is specified in the Release Options of the primary VRS that is holding the volume from being deleted, the volume is expired on its expiration date and time after all VRS retained data sets on a volume have left vital record status.

On the volume, if any data sets exist that are not matched to a VRS during VRSEL, those data sets are retained by DFRMM. The volume is retained while there are data sets on the volume currently retained by a VRS policy. The volume is eligible to be scratched when the last VRS matched data set is expired. It is a best practice to define a default VRS policy that matches on any dsname(\*\*) to prevent the loss of data contained in unmatched data sets.

The unmatched data sets are released when the last formerly VRS retained data set reaches its expiration date. It is a best practice to define a default VRS policy that matches on any dsname (\*\*) to catch data sets that are not matched to a specific VRS.

This method of retention is the original retention method that is available in DFSMSrmm. When a data set satisfies the requirements of the VRS policy that it is retained under and is moved to release pending, it will no longer be processed during the VRSEL function of housekeeping unless the data set expiration date is updated to a future date. Even with a change of VRS policies, a data set that already left vital records retention is not selected and retained for a second cycle without a change that specifies a future expiration date.

If no VRS rule is applied to a data set, the volume it is on is moved to the scratch pool during the EXPROC function of inventory management when its expiration date and time is passed. Therefore, if a customer intends to expire all tape data sets on their provided expiration dates, they need not assign any retention method. This level of retention that is based on the expiration date alone is what is termed *simple expiration retention*. This level is different than the EXPDT retention method, which provides more functions but also centers around the use of the expiration date.

The second and newer method of retention in DFSMSrmm is the EXPDT retention method. This retention method allows data sets to be expired on the expiration date and time of the data set or the deletion of the catalog entry for the data set. The EXPDT retention method does not use the RETENTION DATE field and no value is present in this field for any data set that is managed by using EXPDT or the volumes on which they are contained.

In the VRSEL retention method, the expiration date was the second consideration after a data set left vital record status. In the EXPDT retention method, the expiration date (optionally along with catalog status) becomes the primary consideration for when a data set is expired. If a customer wanted to specify more criteria beyond simple expiration date retention, the EXPDT retention method can be used. It provides various other conditions under which the data set is expired.

One of the primary differences between these two retention methods is that VRS rules are applied to VRSEL-managed volumes each time the VRSEL function of DFSMSrmm housekeeping is run. The status of the EXPDT-managed volumes retention method is adjusted dynamically based on event notifications from OPEN/CLOSE processing or Catalog Address Space.

However, both retention methods rely on the EXPROC function of DFSMSrmm housekeeping. In VRSEL, the volumes move to Pending Release and then, after two cycles of EXPROC, to SCRATCH status. If SCRATCH IMMEDIATE is specified, then one EXPROC cycle is needed.

In EXPDT, only one cycle of EXPROC is required to move a volume to SCRATCH status. For the EXPDT retention method, the CPU cycles and clock time reduction are significant when compared to the VRSEL retention method. The difference is because VRSEL issues many locates to catalog and performs much sort processing. Neither locates or sorts are needed for volumes that are managed under the EXPDT retention method.

Regardless of which retention method is chosen, the retention method is assigned at the time a volume goes from SCRATCH to MASTER or USER status, which is when the first data set is written on the volume. The retention method remains the same for the lifecycle of the volumes unless changed specifically by using a command such as RMM CHANGEVOLUME RETENTIONMETHOD(EXPDT or VRSEL).

## 2.2 Expiration date setting

### 2.2.1 Order of precedence when setting EXPDT and RETPD

The expiration date setting can be obtained from various sources. The order of precedence for EXPDT and RETPD is provided in the following list that is sorted by highest precedence to lowest precedence:

The most common is the EXPDT or RETPD value in the JCL that is submitted to create a tape data set. This JCL-provided EXPDT or RETPD is the highest level of precedence for the expiration date. When a RETPD or EXPDT is passed in JCL, it is honored unless it is overridden in the EDGUX100 exit(JFCBXPDT).

Next in priority is any RETPD or EXPDT value that is passed in the ACS that is assigned a data class. If the customer uses a EDGRMMxx parmlib MCATTR setting other than NONE, the EXPIRE AFTER DATE/DAYS or the EXPIRE AFTER DAYS NON-USAGE setting is next on the priority list. If an EDGDEFxx default table is used, this place is next in the hierarchy where an expiration date can be set.

Also, the Defaults Table function includes the OVERRIDE parameter that allows an expiration date(RETPD) to be set that overrides a JCL-provided expiration date.

Next in priority is the separately specified defaults for EXPDT-managed GDG and non-GDG data sets.

The last item in the following priority list is the EDGRMMxx parmlib RETPD value:

- ▶ The DFSMSrmm installation exit EDGUX100
- ▶ The JCL DD statement that uses the EXPDT or RETPD keywords
- ▶ The EXPDT or RETPD in the DFSMS data class, if the data set is associated with a DFSMS data class

- ▶ The expiration attribute Expire after Date/Days or Expire after Days Non-Usage in the DFSMS management class, if the use is enabled by the DFSMSrmm MCATTR parmlib option and if the data set is associated with such a management class
- ▶ The retention period that is specified in the Defaults Table(DEFTABLE) by using the EDGDEFxx parmlib member
- ▶ Separate defaults that are specific to GDG or non-GDG data sets that are managed by the EXPDT retention method that is specified in the DFSMSrmm parmlib
- ▶ The default retention period RETPD that is specified in the DFSMSrmm parmlib as a global default

However, SVC99 can also be programmatically used to set expiration dates. This SVC99 provided expiration date overrides any determinations that are made by DFSMSrmm. Therefore, it overrides all other expiration date sources.

**Note:** At the time of this writing, a known issue exists with the expiration date setting where the SMS MC class values for EXPIRE AFTER DATE/DAYS or EXPIRE AFTER DAYS/NONUSAGE are being ignored if they are set less than EDGRMMxx RETPD or a value that is passed by Defaults Table ([OA61415](#), [OA60354](#), [OA63281](#), [OA63283](#)).

In the RMM LISTVOLUME output or the ISPF RMM panel display for a volume, a field is available that is directly under the expiration date and time that indicates what source the date was taken from:

<b>CMD</b>	Set by TSO subcommand
<b>CMD_DEF</b>	Default RETPD that is applied during subcommand processing
<b>CMD_VOLCAT</b>	EXPDT that is obtained from VOLCAT during subcommand processing
<b>OCE_JFCB</b>	EXPDT that is obtained from EXPDT/RETPD keywords or from Data Class that is applied during tape recording
<b>OCE_EXIT</b>	EDGUX100 that is updated the JFCB EXPDT during tape recording
<b>OCE_DEF</b>	Default RETPD that is applied during tape recording
<b>OCE_MAX</b>	MAXRETPD was used to reduce the requested EXPDT during tape recording
<b>OCE_VOLCAT</b>	EXPDT that is obtained from VOLCAT during tape recording
<b>LCS - EXPDT</b>	Obtained from VOLCAT for systems that are managed tapes when called from OAM installation exits
<b>LCS_DEF</b>	Default RETPD applied for system-managed tapes when called from OAM installation exits
<b>TVEXTPURGE</b>	Set as a result of TVEXTPURGE parmlib option
<b>CNVT</b>	Set during conversion by EDGCNVT
<b>EXPORT</b>	Set during export processing
<b>LASTREF</b>	Set because of added retention days after last read or write
<b>OCE_MCE</b>	EXPDT that is obtained from Management Class during tape recording
<b>CATRETPD</b>	Set as CATRETPD hours after creation time during tape recording
<b>CATLGDAYS</b>	EXPDT is Catalog Days after the data set is uncataloged
<b>DEFTABLE</b>	EXPDT set by using the Defaults Table in the EDGDEFxx parmlib member



The date that a data set was last referenced can be used to increase the expiration date of a data set. This value can be derived from the management class EXPIRE AFTER DAYS NON-USAGE value or the EDGRMMxx parmlib option LASTREF.

MAXRETPD can come into play with the expiration date setting. When a value is passed that is greater than the MAXRETPD value in the EDGRMMxx parmlib, the expiration date is reduced to be the current date, plus the MAXRETPD value.

## 2.2.2 WHILECATALOG processing in the EXPDT retention method

Under the VRSEL retention method, a WHILECATALOG = YES setting in a VRS policy retains the data set if the data set has a valid entry in an Integrated Catalog Facility (ICF) catalog. When the catalog entry for the data set is deleted, the catalog field in the data set entry in RMM will change to CATALOG = NO. In the next run of VRSEL, the data set will be moved to PENDING RELEASE if the expiration date is in the past or if IGNORY EXPIRYDATE = YES. Otherwise, the data set will be further retained until the expiration date is met.

With EXPDT, the WHILECATALOG function is differentiated into two parms, WHILECATALOG(ON) and WHILECATALOG(UNTILEXPIRED).

With WHILECATALOG(ON), a data set is kept if it has a catalog entry. After the catalog entry is deleted, the data set expires after it reaches its expiration date and time during EXPROC processing. However, when a data set entry that is retained under WHILECATALOG(ON) has its catalog entry deleted, the expiration date can be extended by the value of CATLGDAYS that is specified in EDGRMMxx parmlib if that date is greater than the current expiration date. CATLGDAYS defaults to 2 days if not specified. With WHILECATALOG(UNTILEXPIRED), the data set is kept if it has a catalog entry, but no later than its expiration date.

In the case that the catalog entry is deleted before its expiration date, under WHILECATALOG(UNTILEXPIRED) the expiration date is adjusted to the current date plus the value of CATLGDAYS that is specified in EDGRMMxx parmlib.

WHILECATALOG(ON) is an AND condition. The catalog entry must be deleted and the expiration date reached before the data set is expired.

WHILECATALOG(UNTILEXPIRED) is an OR condition. The data set is expired if its catalog entry is deleted or its expiration date is reached. For users converting from VRSEL to EXPDT, the WHILECATALOG(ON) selection under EXPDT is the closest equivalent of WHILECATALOG without UNTILEXPIRE in VRSEL.

WHILECATALOG(ON) and WHILECATALOG(UNTILEXPIRED) can be set in any of the following ways, which are respected in order of precedence:

- ▶ Management Class WHILECATALOG field on page 8 of MC class definitions
- ▶ Defaults Table assignment of WC(ON) or WC(UX)
- ▶ EDGRMMxx parmlib specification of RM(EXPDT) default actions when the attributes were not specified in any other higher precedence method.

## 2.2.3 EDGRMMxx parmlib options for EXPDT processing

All of the following options except for CATLGDAYS can also be set in the DEFAULT table or the SMS Management Class definitions:

### RETENTIONMETHOD

If no other source is used, then the global default for RETENTIONMETHOD (RM) is used to set a retention method for new tape volume sets. Other sources to set the retention method include Management Class or DEFAULT table. The retention methods are used when new tape volume sets are created during Open/Close/End-of-Volume (O/C/EOV) processing, or through DFSMSrmm commands.

A tape volume set can be a multi-volume set, or a single tape volume. EXPDT can be specified as EXPDT or EXPDT(options) where options can be applied to all data sets or specifically to GDG or non-GDG data sets. For example, RM(EXPDT(GDG(options))) applies only to GDG data sets, RM(EXPDT(NOGDG(options))) applies only to non-GDG data sets, and RM(EXPDT(options)) applies to GDG and non-GDG data sets.

### CATALOG DAYS

CATLGDAYS(ret\_days) specifies that when a data set that is specified with WHILECATALOG(ON) or WHILECATALOG(UNTILEXPIRED) becomes uncataloged, it is kept for at least ret\_days more days (ret\_days can be 0–93000 days). For example, CATLGDAYS(24) keeps data sets for 24 days. Set ret\_days to 0 to have data sets expire the moment that they are uncataloged.

The default is CATLGDAYS(2).

### RETAINBY

You now have the option of retaining volumes with the EXPDT retention method based on a single volume or volume set, or on a controlling first file. The *values* variable in parmlib OPTION RM(EXPDT(RETAINBY(*value*))) includes the following settings:

#### RETAINBY(VOLUME)

Where DFSMSrmm expires volumes in a multi-volume set at the volume level. Each volume has its own expiration date. RETAINBY(VOLUME) is the default value.

#### RETAINBY(SET)

Where DFSMSrmm expires volumes in a multi-volume set at the volume set level. All volumes in the set have the same expiration date, which is the maximum expiration date of all volumes unless it is changed by using the CHANGEVOLUME command.

#### RETAINBY(FIRSTFILE)

Where DFSMSrmm expires volumes in a multi-volume set at the volume set level. All volumes in the set have the same expiration date of the first file in the volume set. A single volume is treated as a volume set with only one volume in it.

You can also set the RETAINBY for a specific volume set by subcommand. The best practice is to use SET, VOLUME, and FIRSTFILE. RETAINBY(SET) better protects your multi-volume files.

### LASTREF

Data sets that are managed by the EXPDT retention method can be retained or expired based on the number of days since the data set was last referenced. This value is taken from the parmlib OPTION RM(EXPDT(LASTREF(*extra\_days*))) where *extra\_days* is a decimal

number 0–93000. The value must not exceed the maximum retention period (MAXRETPD) that is specified in the EDGRMMxx parmlib. An *extra\_days* value of 0 has the same effect as the NOLASTREF operand.

DFSMSrmm uses the data set LASTREF value to determine the data set expiration date. The extra days are added to the date of last reference. The data set expiration date is set to the maximum of the date that is calculated by using the data set LASTREF value and the date that results from applying the EXPDT, RETPD, or default RETPD. Any reference to the data set by a read or write operation changes the expiration date.

When a data set is added to DFSMSrmm on a volume that is managed by the EXPDT retention method and neither LASTREF nor NOLASTREF is specified for the data set, then DFSMSrmm uses the default LASTREF value in EDGRMMxx parmlib. If neither LASTREF nor NOLASTREF are specified in EDGRMMxx parmlib, NOLASTREF is used by default.

If the volume set is retained by SET or VOLUME, the LastReferenceDays data set attribute is kept equal for all files of a multi-volume data set. The latest LastReferenceDays update to a single file in a multi-volume data set is propagated to all files that belong to the same multi-volume data set.

However, for volume sets that are retained by FIRSTFILE, the LastReferenceDays data set attribute is set, but not equalized across the multi-volume data set because the expiration date depends only on the first file of the first volume, and its LastReferenceDays.

The LastReferenceDays value can also be set by using the ADDDATASET subcommand when the data set record is created. The LastReferenceDays value can be changed by using the CHANGEDATASET subcommand after the data set record was created.

## RETPD

RETPD(*ret\_days*) where *ret\_days* specifies the default retention period in days for data sets on volumes that are managed by the EXPDT retention method. Consider the following points:

- ▶ *ret\_days* can be a number 0–93000 or the keyword PERMANENT.
- ▶ RETPD(0) specifies that the default expiration date is the same as the creation date. Data sets with RETPD(0) and WHILECATALOG(ON) that are specified are managed by their catalog status only.
- ▶ RETPD(PERMANENT) specifies that the default expiration date for new data sets is to be 1999/365, which means that the data set is retained permanently. This specification is useful with the WHILECATALOG(UNTILEXPIRED) operand to create data sets that are kept only by their catalog status unless RETPD or EXPDT is specified in the JCL or the data class, which always overrides an EDGRMMxx parmlib member that is specified RETPD/EXPDT.
- ▶ If RETPD is not specified on the EXPDT operand, the default retention period that is specified in the global RETPD parmlib option (which is used for both retention methods) is used.

## WHILECATALOG

WHILECATALOG(*value*) specifies the default WHILECATALOG value for data sets on volumes that are managed by the EXPDT retention method. The following list describes the possible values:

- ▶ WHILECATALOG(OFF) specifies that new data sets that are on volumes that are managed by the EXPDT retention method have the WHILECATALOG attribute set to OFF by default. As a result, retention of these data sets depends only on their expiration date and not on their catalog status.

- ▶ **WHILECATALOG(ON)** specifies that new data sets that are on volumes that are managed by the EXPDT retention method have the WHILECATALOG attribute set to ON by default. As a result, these data sets are not expired if they are cataloged, nor are they expired before their expiration date.

When DFSMSrmm determines that a data set is no longer cataloged, DFSMSrmm looks at the current expiration date and can reset the expiration date of the data set to a date that is equal to the date that the data set was uncataloged plus the number of days that is specified by the CATLGDAYS option if this date is greater than the current expiration date.

The expiration date of new data sets under WC(ON) is set to the greater of the EXPDT value or (creation date + CATRETPD hours). The CATRETPD option is used to ensure that new data sets are not prematurely expired during housekeeping before the catalog entry is created.

- ▶ **WHILECATALOG(UNTILEXPIRED)** specifies that new data sets that are on volumes that are managed by the EXPDT retention method by default have the WHILECATALOG attribute set to UNTILEXPIRED. As a result, these data sets expire after they are uncataloged or if the expiration date is reached.

When DFSMSrmm determines that a data set is no longer cataloged, the DFSMSrmm resets the expiration date of the data set to a date. The new date is the date that the data set was uncataloged plus the number of days that is specified by the CATLGDAYS option. WHILECATALOG(UNTILEXPIRED) has no effect on data sets that are never cataloged.

The default is WHILECATALOG(OFF).

Most customers prefer to manage GDG data sets with WHILECATALOG(ON) so that the data sets are not expired before the ALTER ROLLOUT processing is done when a GDG reaches its LIMIT value.

If WHILECATALOG(ON) is used with non-GDG data sets, care must be taken that a reasonable and accurate expiration date is set so that data sets are not kept permanently unless wanted, which can lead to an unintentional buildup of older data sets that are not needed.

With WHILECATALOG(UNTILEXPIRED), the opposite condition is a concern in that inaccurate expiration dates might lead to premature data expiration.

Of the two settings, WHILECATALOG(ON) is the most conservative of the two choices. The customer can manage data retention by setting short expiration dates and ensuring that unneeded data set entries are deleted.



# Introduction to the Defaults Table

This chapter discusses the DFSMSrmm Defaults Table and includes the following topics:

- ▶ 3.1, “Overview” on page 14
- ▶ 3.2, “Defaults Table command operands” on page 15
- ▶ 3.3, “EDGUX100 EXIT” on page 19

## 3.1 Overview

Defaults Table support was introduced in V2R3 and is a predefined set of rules that apply to tape expiration attributes. These rules are contained in an EDGDEFxx parmlib member that is pointed to by the DEFTABLE option in the EDGRMMxx parmlib. During tape data set creation, these rules are used to set expiration attributes dynamically.

The Defaults Table can be easier to create and maintain. It is meant as a replacement for the EDGUX100 and EDGCRSX/UXTABLE functions in older releases of z/OS Both are still supported in current releases as of this writing.

A Defaults Table cannot be specified in EDGRMMxx parmlib while a UXTABLE is loaded. DFRMM issues WTOR EDG0217D and asks which must be used. EDGUX100(EDGCVR SX) without UXTABLE can be loaded with DEFTABLE. Although they can coexist, EDGUX100 has a higher priority than DEFTABLE.

When a data set is created, DFRMM searches the rules in the Defaults Table from the beginning until a matching entry is found. If the CONTINUE keyword is coded, the search continues for any other match. If that match also includes the CONTINUE keyword, searching continues until no match is found or a match is found without the CONTINUE keyword.

DFRMM remembers the attributes that are specified at each qualified match and applies all of the attributes that are contained in each of the matching rules. If two matches specify a different attribute or a different value for the same attribute, the last rule that matched is applied.

An entry qualifies for a match only when all three search criteria are satisfied: DSNAME Mask, Jobname Mask, and KEYDATE. The default for DSNAME Mask is \*\*, which matches for any data set rules without a dsname mask. The default for Jobname Mask is \*, which matches if no jobname mask is present. The default for KEYDATE is \* which matches if no key date is specified in the rule.

The following values can be used in the Defaults Table entries for matching:

- ▶ DSNAME is case-sensitive and cannot contain blanks, X'00', or the member names of a PDS/PDSE. Standard masking can be used as in other areas of RMM; that is, % as a place holder for a single character, \* for single qualifier, \*\* for multiple qualifiers, and when used at the end of the mask to ignore the remaining characters.
- ▶ KEYDATE is used to match against EXPDT values that are passed in through JCL. It is coded as yyddd where yy is the years 1900–1999 (coded as 00–99), and ddd is the Julian Date value 0–366.

NOKEYDATE means that the rule cannot include a KEYDATE value that is specified in any other rule in the EDGDEFxx entry for a rule to match it.

With a new function in the fix for APAR OA64125, a new keyword, NODATE, is added for DEFTABLE and can be used for matching. For more information, see [APAR OA64125](#).

NODATE means that the data set only matches if it does not have an expiration date(EXPDT) or retention period(RETPD) specified in the JCL or original expiry day assigned by using Data Class or Model DSCB.

- ▶ JOBNAME is used to match a rule to the job that created the data set. The % can be used to match any one character and \* to match any character string that is inside the job name mask.
- ▶ PGMNAME is used to match a rule to the program that created the data set. The same wildcards can be used with program name as with job name.

- ▶ POOL is useful for Manual Tape Libraries (MTLs) or stand-alone tape drive environments. It is used to append a scratch pool value to the write to operator (WTO) mount messages so that tape solutions can copy those messages and use them to satisfy PRIVAT tape mounts.

POOL is incompatible with KEYDATE, NOKEYDATE, and PGMNAME operands. The scratch pool prefix must also be defined in the EDGRMMxx parmlib by using the VLPOOL command.

**Note:** Scratch pools are *not* used in In a non-MTL system-managed tape environment. A common scratch pool per media type is used instead.

The scratch pool selection can be taken from the following sources, which are listed in priority order with the highest priority first:

- ▶ Storage group name as determined by the ACS routines for nonsystem managed libraries when SMSACS(YES) is in effect.
- ▶ POOL(pool\_mask) in the Defaults Table in EDGDEFxx.
- ▶ PL100\_POOL is returned by the EDG\_EXIT100 exit routine.
- ▶ Storage group name as determined by the ACS routines for MTLs. This is ignored if the EDG\_EXIT100 exit is in use and sets the PL100\_SET\_IGNORE\_SGNAME bit. This bit is set by default if you use an unmodified version of EDGUX100 or EDGCVRSX for your user exit.
- ▶ DFSMSrmm system-based scratch pooling.

## 3.2 Defaults Table command operands

This section includes a discussion of the command operands that can be coded in the Defaults Table to assign expiration attributes to data sets at creation time.

### 3.2.1 RETENTIONMETHOD

The RETENTIONMETHOD command operand is VRSEL or EXPDT. It is defined when the first data set is written to the volume or volume chain, but can be changed later by using the RMM CHANGE VOLUME command. VRSEL is the default and is the traditional retention policy that consists of Vital Record Specification rules that are applied during EDGHSKP VRSEL.

EXPDT is a newer retention method and is based on the expiration date of the data sets that are on a volume. They are set dynamically as data sets are written and optionally the status of the catalog entry for the data set. Volumes that are managed with EXPDT include the VRSELEXCLUDE attribute that is applied by default.

**Note:** When no retention method is applied, RMM retains data sets by default until they meet their expiration date. This issue can occur if VRSEL is the default retention method, but no VRS rule was coded that matches during VRSEL or when no Default Table matches are found. RETENTIONMETHOD can be abbreviated to RM.

The RETENTIONMETHOD for volumes can be taken from the following sources, which are listed in highest to lowest priority order:

- ▶ Retention Method in the Management Class if EDGRMMxx parmlib option MCATTR is a value other than NONE
- ▶ PL100\_RETENTIONMETHOD, which is returned by the EDG\_EXIT100 exit routine
- ▶ RETENTIONMETHOD in the Defaults Table in EDGDEFxx
- ▶ The default EDGRMMxx parmlib OPTION RM

### 3.2.2 VRSELEXCLUDE

VRSELEXCLUDE(YES) causes a data set to be excluded from VRSEL processing, which does not affect volumes that are managed by using retention method EXPDT because these data sets always include the VRSELEXCLUDE attribute set by default. This value can be changed by using the RMM CHANGEDATASET command for a single data set on the volume.

To exclude all data sets on the volume from VRSEL processing, use the RMM CHANGEVOLUME volser RM(EXPDT) command. VRSELEXCLUDE can be abbreviated to VX.

The VRSELEXCLUDE value for newly written data sets can be taken from the following sources, which are listed in priority order, with the highest priority first:

- ▶ Exclude from VRSEL in the management class when EDGRMMxx parmlib option MCATTR is a value other than NONE
- ▶ PL100\_SET\_VRSELEXCLUDE, which is returned by the EDG\_EXIT100 exit routine
- ▶ VRSELEXCLUDE(YES/NO) as specified in the Defaults Table

This option can be used to cause a data set and volume to be retained until its expiration date, which is the default when no retention criteria are assigned.

### 3.2.3 EDM and NOEDM

EDM and NOEDM are used to designate whether the volume is to be marked for external data manager management. EDM prevents the volume from being expired by RMM and requires external data managers, such as HSM and OAM, to specifically release the volume by using the EDGTVEXT or EDGDFHSM interfaces.

This attribute is also activated by the global parameter EDM(Y) in the EDGRMMxx parmlib member for the data set that is created by ANZSRVR, ANRSERV, ARCCTL, ARCWCTL, CBROAM, or DSMSEV. Having this parameter as an option to set in the Defaults Table allows more volumes to be marked as EDM managed. The use of NOEDM allows exceptions to be made to the list of programs that activate EDM automatically with the EDM(Y) global option for data sets that are created by ANZSRVR, ANRSERV, ARCCTL, ARCWCTL, CBROAM, and DSMSEV.

### 3.2.4 RETPD

RETPD specifies the number of days the data set is to be retained. The value range is 0–93000 and is added to the current date and time to calculate the new expiration date and time. If PERMANENT is specified, the expiration date is set to 12/31/1999 or 1999/365.



Adding **OVERRIDE** to the **RETPD** parm causes the **RETPD** value in the Defaults Table to override what is specified in the JCL.

The expiration date for newly written data sets can be taken from the following sources, which are listed in priority order, with the highest priority first:

- ▶ The value in the JFCB control block, which can come from the following three sources:
  - The **RETPD** or **EXPDT** value that is specified in the JCL.
  - The **RETPD** or **EXPDT** value that is specified in the **DATACLAS**.
  - The **EDG\_EXIT100** installation exit. The **RO** option in the **UXTABLE** defines whether the exit overrides the JCL value.
- ▶ Expire after Date/Days in the management class if the **EDGRMMxx** parmlib option **MCATTR** is a value other than **NONE**.
- ▶ **REPTD** in the Defaults Table.
- ▶ The default **EDGRMMxx** parmlib option **REPTD(value)**, superseded by **RM=EXPDT-specific options RM(EXPDT(RETPD(value)))** or **(RM(EXPDT(NOGDG(RETPD(value)))) and RM(EXPDT(GDG(RETPD(value))))).**

### 3.2.5 VRSVALUE

**VRSVALUE** specifies a Vital Record Specification (**VRS**) management value that is to be associated with the data set. A **VRS** management value is 1–8 characters other than blank, comma, and semicolon.

The **VRSVALUE** must be a single qualifier and can be changed by using the **RMM CHANGEDATASET** command. **VRSVALUE** can be specified as **MANAGEMENTVALUE**. **VRSVALUE** and **MANAGEMENTVALUE** are valid for **VRSEL** managed data sets only.

The **VRS** management value for newly written data sets can be taken from the following sources, which are listed in priority order, with the highest priority first:

- ▶ **PL100\_VRS**, which is returned by the **EDG\_EXIT100** exit routine
- ▶ **VRSVALUE** as specified in the Defaults Table

### 3.2.6 RETAINBY

**RETAINBY** in the Defaults Table applies to **RM(EXPDT)**-managed volumes only and specifies how **RMM** is to retain a volume or multi-volume set. Another **RETAINBY** parameter is available in the **EDGRMMxx** parmlib that pertains to **VRSEL** volumes.

The value **FIRSTFILE** specifies that the expiration date of the first file is used to set the expiration date of a single volume or a multi-volume set. The dates of other data sets on the volume or to a multi-volume set do not influence the volume expiration date.

The **SET** value specifies that all volumes in the set feature the same expiration date, which is derived from the highest expiration date of all the data sets on the volumes in the set.

**VOLUME** specifies that the expiration date of the volume is set for each volume separately and each file on a volume can increment the volume expiration date.

**Note:** **FIRSTFILE** is not usually recommended. **SET** or **VOLUME** must be used.

The RETAINBY value for volumes can be taken from the following sources, which are listed in order of highest to lowest priority:

- ▶ Volume Set Management Level in the Management Class if the EDGRMMxx parmlib option MCATTR is a value other than NONE
- ▶ RETAINBY in the Defaults Table in EDGDEFxx
- ▶ The default EDGRMMxx parmlib option RM(EXPDT(RETAINBY(value)))

### 3.2.7 WHILECATALOG

WHILECATALOG is a RM(EXPDT)-specific attribute that adds a dependency for retention on the status of the data sets catalog entry.

A value of WHILECATALOG(ON) specifies that the data set is expired only after the catalog entry for the data set is deleted and is passed its expiration date. The expiration date is set to its EXPDT value or the creation date plus CATRETPD value, whichever is later.

If WHILECATALOG is ON and the data set is cataloged, the data set expiration date is displayed with (KeptByCatlg) appended to the date. When a WC(ON) data set has its catalog entry deleted, the expiration date can be extended by the value of CATLGDAYS in EDGRMMxx parmlib if the resulting value is greater than the current expiration date.

A value of WHILECATALOG(UNTILEXPIRED) specifies that the data set is expired after its catalog entry is deleted, or it passed its expiration date. When such a data set has its catalog entry deleted, RMM resets the expiration date to the current date plus the value that is specified in EDGRMMxx parmlib option CATLGDAYS, which defaults to 2 if not specified.

If WHILECATALOG is UNTILEXPIRED and the data set is cataloged, the data set expiration date is displayed with (OrUncatlg) appended to the date.

The WHILECATALOG value for new data sets can be taken from the following sources, which are listed in order of highest to lowest priority:

- ▶ Retain While Cataloged in the Management Class if the EDGRMMxx parmlib option MCATTR is a value other than NONE
- ▶ WHILECATALOG in the Defaults Table in EDGDEFxx
- ▶ The default EDGRMMxx parmlib options:
  - RM(EXPDT(NOGDG(WHILECATALOG(value))))
  - RM(EXPDT(GDG(WHILECATALOG(value))))

WHILECATALOG(ON) can be abbreviated as WC(ON) and WHILECATALOG(UNTILEXPIRED) can be abbreviated as WC(UX).

### 3.2.8 LASTREF(extra\_days)

LASTREF(*extra\_days*) is an RM(EXPDT) attribute that uses the date that the data set was last referenced plus an additional 0–93000 days to determine whether the expiration date of the data set must be extended.

Any read or write reference to the data set causes DFRMM to determine whether the expiration date must be extended. A value of 0 means that the read or write reference to the data set does not cause the expiration date to be extended.

For a volume set that is retained by VOLUME or SET, DFSMSrmm ensures that the LASTREF(*extra\_days*) attribute is the same for all files of a multi-volume data set.

The LASTREF(*extra\_days*) value for new data sets can be taken from the following sources, which are listed in order of highest to lowest priority:

- ▶ Expire after Days Non-usage in the Management Class if MCATTR value is other than NONE
- ▶ LASTREF(*extra\_days*) in the Defaults Table in EDGDEFxx
- ▶ The default EDGRMMxx parmlib option RM(EXPDT(LASTREF(value)))

### 3.3 EDGUX100 EXIT

The IBM EDGUX100 EXIT function as offered is to look for special dates in the JCL allocating tape data sets and set specific VRS management values. The management values are associated with a set of defined VRS attributes to control expiration processing.

For example, D99000 often is associated with JCL-coded parameter EXPDT=1999/000 or EXPDT=99000. This VRS management value is then used to assign the WHILECATALOG attribute. This attribute specifies that a data set is expired when it has its catalog entry deleted. These special dates are called KEYDATES and can be coded into the Defaults Table so that the EDGUX100 exit does not need to be installed. This function in the EDGUX100 exit can be replaced by using a Defaults Table statement, as shown in Example 3-1.

*Example 3-1 Defaults Table statements*

---

```
DEFAULT KEYDATE(99000) VRSVAL(D99000)
DEFAULT KEYDATE(99365) VRSVAL(D99365)
DEFAULT KEYDATE(99366) VRSVAL(D99365)
```

---

These examples can be customized to match the VRS rules that are in place by altering the VRSVAL parameter. For example, many sites use M99000 instead of D99000.

Any value can be used for KEYDATE so that other dates (besides the dates that are shown in Example 3-1) can be accommodated without the need for the EDGUX100.

When system-managed tape is used, or for nonsystem managed tape with SMSACS(YES) in the EDGRMMxx parmlib, it is also possible to check for these special expiration dates in the ACS routines and assign a suitable management class with the expiration values that are wanted.

Assuming that the storage class for tape is called SC3490, the following code can check for EXPDT=1999000 in the JCL of the job that is allocating the tape data set and set a Management Class that is called D99000 that specifies WHILECATALOG or other attributes as suitable in your environment:

```
IF ((&STORCLAS = 'SC3490') AND (&EXPDT='1999000')) THEN
DO
SET &MGMTCLAS = 'D99000'
EXIT CODE(0)
END
```

If you use the EDGUX100 EXIT along with a UXTABLE in addition to special date processing, it can be converted into a Defaults Table.

In SYS1.SAMPLIB, a member EDGJDEF exists that contains JCL for a job that starts the EDGRDEF script to convert the UXTABLE into a Defaults Table.



# Using SMS Management Class to assign tape expiration values

This chapter discusses the implementation of Management Class support in DFSMSrmm. It includes the following topics:

- ▶ 4.1, “Overview” on page 22
- ▶ 4.2, “Expiration attributes” on page 22
- ▶ 4.3, “Data set management” on page 26

## 4.1 Overview

In V1R13, DFSMSrmm introduced Management Class support for tape expiration values. This support integrates with the new EXPDT Retention Method or can be used with the established VRSEL retention method.

ACS routines can be used to select an SMS management class that specifies retention method and catalog status dependency criteria. The SMS management class can also pass EXPIRE AFTER DATE/DAYS or EXPIRE AFTER DAYS NON-USAGE (LASTREF) values to set the expiration date of a data set.

Another benefit of Management Class support is that it allows the assignment of expiration policies by using a retention method without the need for user exits. The function was added to the EXPDT retention method and Management Class support (MCATTR) in each release since V1R13 and continues to evolve.

## 4.2 Expiration attributes

This section includes a discussion of the available expiration attributes.

### 4.2.1 MCATTR option of parmlib member EDGRMMxx

In the DFSMSrmm parmlib member EDGRMMxx, option MCATTR has the following values:

- ▶ ALL
- ▶ NONE
- ▶ VRSELXDI.

The default value is NONE. When the value is NONE, DFSMSrmm does not use the Management Class tape expiration attributes. It is still possible to use the EXPDT retention method.

**Note:** Retention method, EXPDT or VRSEL, along with other tape expiration attributes can be set in EDGRMMxx or Defaults Table in addition to the Management Class.

#### **ALL value**

With an MCATTR value of ALL, DFSMSrmm applies the following expiration attributes that are specified in the management class:

- ▶ Expire after Days Non-usage
- ▶ Expire after Date/Days
- ▶ Retention Method
- ▶ Volume Set Management Level
- ▶ Exclude from VRSEL
- ▶ Retain While Cataloged

**Note:** Implement the ALL value with care because tape-related management classes might unintentionally affect the expiration date.

The first two attributes are on the first page of the Management Class definitions. See Figure 4-1. Retention Limit is not yet supported in DFRMM as of V2R5 general availability.

```
Expiration Attributes
  Expire after Days Non-usage . . . :
  Expire after Date/Days . . . . . :
  Retention Limit . . . . . :
```

Figure 4-1 First page of the management class definition

The remainder of the tape expiration attributes are on the last page of the management class definitions. See Figure 4-2.

```
Tape Volume Attributes
  Retention Method . . . . . :
  Volume Set Management Level . . :

Tape Data Set Attributes
  Exclude from VRSEL . . . . . :
  Retain While Cataloged . . . . . :
```

Figure 4-2 Last page of the management class definitions

**VRSELXDI value**

With an MCATTR value of VRSELXDI, the attributes (as shown in Figure 4-1 and in Figure 4-2) are used for data sets and volumes that are assigned the EXPDT retention method but not for volumes that are assigned the VRSEL retention method. This choice is the most logical choice to use for migrating to the use of the management class attributes. It tolerates both retention methods simultaneously for different workloads or applications. It is also acceptable to use after all workloads are migrated to the EXPDT retention method.

The management class tape expiration attributes are applied during the OPEN for output for a new data set only, Changes that are made to them are not retroactively applied to existing data set and volumes.

## 4.2.2 Changing attributes

After these attributes are set, they can be changed by using the change volume command, RMM CV, and the change data set command, RMM CD. They are not reevaluated during each housekeeping run as VRSEL attributes are, which avoids the overhead that is incurred during traditional VRSEL processing.

During DISP=MOD,OLD, or SHR processing, the attributes are maintained as specified during the original data set open and allocation. You can change the management class that is assigned by using the RMM CV MANAGEMENTCLASS command; however, this change does not alter the original attributes that were assigned. Instead, only the management class name is altered.

## 4.2.3 Management class attributes

The following management class attributes are available:

### Expire after Date/Days

This field shows the expiration date or the number of days after creation before data sets in this management class expire. The following possible values can be specified:

- *yyyy/mm/dd* in which *yyyy* is the year, *mm* is the month, and *dd* is the day
- 0–93000 days
- NOLIMIT

This value is applied only if the retention period, RETPD, or expiration date, EXPDT, is not passed by using JCL or data class. This value takes precedence over values that are set in UXTABLE, DEFAULT TABLE (EDGDEFxx), or in the global default parameters that are specified in the DFSMSrmm parmlib member EDGRMMxx.

If this value is NOLIMIT, the expiration date is not set by the management class unless NOLIMIT is also specified for Expire After Days Non-usage, in which case the data set is retained permanently with the date 12/31/1999.

**Note:** The expiration date value set is limited to the EDGRMMxx parmlib value MAXRETPD.

### Expire after Days Non-usage

This field shows the number of days that data sets in this management class can go unaccessed before expiring. Values are specified as 1–93000 or NOLIMIT.

If this value is NOLIMIT, the LAST REF EXTRA DAYS attribute is not set from the management class unless Date/Days is also set to NOLIMIT, in which case the data set is retained permanently with the date 12/31/1999.

If both Expire after Days Non-usage and Expire after Date/Days fields feature NOLIMIT, the data set is retained permanently. If only one of these field includes a value of NOLIMIT, the other field is used to determine the expiration date or number of days until expiration.

If both fields specify a numerical value, the data set expires on the latter of the two dates. Expiration time and date can be changed by using the RMM CV EXPDT or RMM CV EXPTM commands.

**Note:** In any combinations of these values, the expiration date value set is still limited to the EDGRMMxx parmlib value MAXRETPD.



## Retention Method

This field specifies how new tapes that are written are to be managed by DFSMSrmm.

The retention method attribute for a volume is applied when the first data set is written to that volume. EXPDT retention allows expiration on a specific date and time, number of days since creation, number of days since last referenced, or based on catalog retention.

The VRSEL retention method uses vital record specifications to implement movement and retention policies through which a retention date is calculated whenever the VRSEL function of inventory management is run.

DFSMSrmm retains a volume that is based on this calculated retention date and on the volume expiration date, depending on whether the VRS release option EXPIRY DATE IGNORE is specified. This value can be changed by using the RMM CV RETENTIONMETHOD command.

## Volume Set Management Level

This field shows how DFSMSrmm manages a volume and volume set under the EXPDT retention method. The following values are available:

### Volume

The expiration date of a set is determined separately for each volume in the set. Unless changed by command, the expiration date is the maximum date of all the expiration dates of the data sets that are on the volume.

### Firstfile

The expiration of a volume is determined by the expiration date of the first data set that is written to the volume. All volumes in a set feature the same expiration date as the first data written to the set.

### Set

The expiration date of a volume is the maximum of all the data sets on the volumes in the set. All volumes in the set feature the same expiration date.

Set or Volume are the recommended choices. Firstfile is used only when all files on a volume must expire at the same time.

## Exclude from VRSEL

This field specifies whether tape data sets that were created on a VRSEL-managed volume must be excluded from VRSEL inventory management processing. A value of YES means the data sets on the volume are excluded from VRSEL processing. This attribute is automatically set for all data sets on a volume that are managed with the EXPDT retention method. It cannot be set if Retention Method EXPDT is also specified in the management class.

## Retain While Cataloged

This field specifies the WHILECATALOG attribute options for data sets on volumes that are managed by the EXPDT retention method. It does not apply if retention method VRSEL is specified in this management class. VRSEL use of WHILECATALOG is specified in the VRS rules, not in the management class.

The following values are available:

- |     |   |
|-----|---|
| OFF | The expiration of the volume depends on the expiration date of the volume and is the default setting if not specified.                                    |
| ON  | The data sets are kept if a catalog entry exists for them. After the catalog entry is deleted, the data set is kept until the expiration date is reached. |

When the catalog entry is deleted, RMM compares the expiration date to the current date plus the value of CATLGDAYS in the EDGRMMxx member (the default is 2 days if not specified) and uses whichever date retains the data sets the longest.

#### Untilexpired

The data sets are kept until the catalog entry is deleted or the expiration date is reached. When the catalog entry is deleted, RMM decreases the expiration date to the current date plus the value of CATLGDAYS in the EDGRMMxx member (the default is 2 days if not specified).

## 4.3 Data set management

When you specify EXPDT with WHILECATALOG(ON), the catalog entry governs expiration. For example, Generation Data Group (GDG) expiration is an area in which this method can be used.

When the generation data set (GDS) is moved out of the GDG base because it reached the limit value of that GDG, the data set is expired if the expiration date is passed. If the expiration date is not passed, the GDS expires on the expiration date.

The LIMIT value of the GDG then becomes the number of generations that are retained if EXPDT/RETPD is set to a shorter duration than the expected lifecycle of the data.

The use of WHILECATALOG(UNTILEXPIRED) allows for a method in which the expiration date is used as the upper limit for data set expiration. Non-GDG data sets can be managed so that when the catalog entry of the data set is deleted, or it reaches its expiration date, the data set is eligible for expiration. This method of expiration is similar to the way many customers handle data sets on DASD; that is, when the catalog entry is deleted, or the data set expiration is reached the DASD data set is scratched.

The use of the SMS Management Class support can be combined with Defaults Table assignments, but this combination can be confusing. Therefore, it can be easier to assign expiration values by using one of these methods rather than both.

SMS Management Class often has precedence over the Defaults Table assignments.



# Logical Write Once Read Many changes

This chapter discusses the implementation of LWORM retention values at the hardware level and Logical Write Once Read Many (LWORM) processing changes on the z/OS host.

This chapter includes the following topics:

- ▶ 5.1, “Overview” on page 28
- ▶ 5.2, “Adjusting a volume’s expiration date” on page 29
- ▶ 5.3, “Managing data sets” on page 29
- ▶ 5.4, “Keeping LWORM volumes permanently” on page 32

## 5.1 Overview

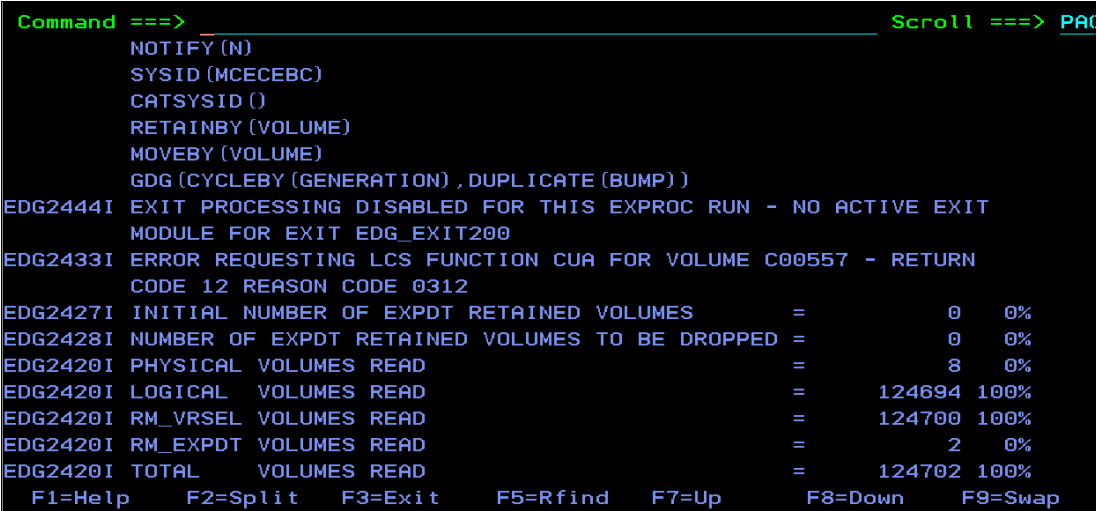
DFSMSrmm changed Logical Write Once Read Many (LWORM) processing in APARs [OA61055](#), [OA62765](#), and [OA64338](#). As a result of the change, you can recover an LWORM volume that is accidentally defined as scratch by first issuing the following command:

```
RMM CV volser STATUS(MASTER) EXPDT(futuredate)
```

After entering the command, use IDCAMS to redefine the catalog entries for the data sets on the LWORM tape. Because of the changes in the APARs, the WORM-related fields and the name of the first data set on the volume are preserved when LWORM volumes are returned to scratch.

In addition to these changes, the R5.2 code level for the TS7700 provides a feature that can prevent LWORM volumes from being returned to scratch. With this LWORM retention function enabled, TS7700 settings regarding LWORM retention can be set in the hardware Data Class to prevent premature return to scratch processing. This support effects scenarios in which specific commands are issued to return a volume to scratch. The support also effects scenarios in which the volume settings match the normal expiration criteria in the defined rules of the tape management system and becomes eligible for return to scratch processing.

If the conditions that are specified in the TS7700 settings for LWORM expiration were not met, any call to move the volume to SCRATCH fails with the message EDG2433I, as shown in Figure 5-1 and the volume is left in MASTER/PRIVATE host status and in the private category (XXXF) in the TS7700.



```
Command ==> _____ Scroll ==> PAU
NOTIFY (N)
SYSID (MCECEBC)
CATSYSID ()
RETAINBY (VOLUME)
MOVEBY (VOLUME)
GDG (CYCLEBY (GENERATION) , DUPLICATE (BUMP) )
EDG2444I EXIT PROCESSING DISABLED FOR THIS EXPROC RUN - NO ACTIVE EXIT
MODULE FOR EXIT EDG_EXIT200
EDG2433I ERROR REQUESTING LCS FUNCTION CUA FOR VOLUME C00557 - RETURN
CODE 12 REASON CODE 0312
EDG2427I INITIAL NUMBER OF EXPDT RETAINED VOLUMES           =           0    0%
EDG2428I NUMBER OF EXPDT RETAINED VOLUMES TO BE DROPPED   =           0    0%
EDG2420I PHYSICAL VOLUMES READ                             =           8    0%
EDG2420I LOGICAL VOLUMES READ                              =        124694 100%
EDG2420I RM_VRSEL VOLUMES READ                             =        124700 100%
EDG2420I RM_EXPDT VOLUMES READ                             =           2    0%
EDG2420I TOTAL VOLUMES READ                               =        124702 100%
F1=Help   F2=Split  F3=Exit   F5=Rfind  F7=Up    F8=Down  F9=Swap
```

Figure 5-1 Rejection of attempted move to scratch

Return code 12 and reason code 0312 are defined as “Requested function is incompatible with the library”. The code is issued by the TS7700 library back to the host directly when the attempt to change the category code is rejected. No method is available for overriding the TS7700 settings from the host. Also, the volume cannot change categories until the specified TS7700 settings are satisfied.

The volume remains available for read activity and can be appended to as normal for LWORM processing. No error condition is set on the host software side and the volume is not moved to the error category (XXXE) by the hardware. The reason for the attempt to move the volume by the Tape Management System must be analyzed and addressed.

## 5.2 Adjusting a volume's expiration date

In DFSMSrmm, the volume's expiration date can be adjusted to match the criteria that are specified in the TS7700 settings. If needed, the retention method can be changed to EXPDT by using the following TSO command:

```
RMM CV volser RM(EXPDT) EXPDT(YYYY/DDD)
```

The volume then moves correctly to SCRATCH during the return to scratch processing on or after the date specified whether the EXPDT value was set to the date that is allowed by the TS7700 LWORM settings or later. When this command is used, it might be wanted to set the expiration date value one more day past what is specified in the TS7700 settings. Doing so avoids a situation in which the return to scratch attempt occurs earlier in the day that the TS7700 timestamp for volume creation allows.

Normally, when a volume is created, this extra day is not needed because the timestamp on the host and the TS7700 for the expiration values are the same.

## 5.3 Managing data sets

This section describes how to manage data sets on LWORM volumes.

### 5.3.1 Mirroring retention settings

One method to manage data sets on LWORM volumes for which retention settings were specified for the TS7700 Data Class is to mirror those settings in the tape management system. In this way, the DFSMSrmm does not attempt to scratch the volumes before it is allowed by the TS7700 settings.

For example, in DFSMSrmm, you can specify the use of retention method EXPDT. You can then specify a number of days for retention that matches the FIXED DURATION value that is set in the TS7700 Data Class that is used for allocating LWORM volumes.

With SMS ACS routines and EDRMMxx parm MCATTR, the allocations of the data sets that are written to LWORM volumes can be driven to a Data Class and Management Class pair where the EXPIRE AFTER DATE/DAYS value in the SMS Management Class matches the required specifications for retention in the TS7700 Data Class.

Alternatively, the expiration date can be specified in the SMS Data Class or passed in the JCL rather than the Management Class. Tape expiration criteria can also be defined by using the DFSMSrmm Defaults Table to assign the same attributes that are available in the Management Class.

Figure 5-2 shows a user defining the configuration of Data Class D00YFIX2 to the TS7700, which indicates that the LWORM retention is a fixed duration of 7 days.

```

Please input the target data class: 1=ALL, 2=DEFAULT, 3=Individual data class
3

Please input the target data class name
D00YFIX2

/usr/vtd/persistent_data/vtd_grid/LWORM_RETENTION_D00YFIX2 will be added

Please specify the retention options individually:
Input Fixed Duration      : -1 (forever) or 0 (none) or X days (1 - 292800)?
7
Input Appl Managed Duration: -1 (forever) or 0 (none) or X days (1 - 292800)?
0
Input Retention Type      : 0 (FIXED) or 1 (HDR1)?
1
Input HDR1 sub-options   : NO_FIRST_HDR1_TREAT_AS_NO_DATE: Y(y) or N(n)?
N
Input HDR1 sub-options   : HONOR_ALL_SUBSEQUENT_HDR1S: Y(y) or N(n)?
Y
Input HDR1 sub-options   : NO_HDR1_MOD_USE_FIXED: Y(y) or N(n)?
N
Input HDR1 sub-options   : SUBSEQUENT_HDR1_HONOR_NO_DATE: Y(y) or N(n)?
N
Input HDR1 sub-options   : SUBSEQUENT_HDR1_HONOR_APP_MANAGED_DATE: Y(y) or N(n)?
N
Input HDR1 sub-options   : APPLY_FIXED_DURATION_FOR_HDR1_NO_DATE: Y(y) or N(n)?
Y
Input Return To Scratch  : HONOR_RETURN_TO_SCRATCH: Y(y) or N(n)?
N
Input RTS Use Fixed Dur  : USE_FIXED_DUR_ON_RETURN_TO_SCRATCH: Y(y) or N(n)?
Y

```

Figure 5-2 TS7700 DC fixed duration of 7

In Figure 5-3, the SMS Management class that is called MC00FIX2 shows a value of 7 in the EXPIRE AFTER DATE/DAYS field to match the fixed duration of 7 days that is specified in the TS7700 LWORM retention settings.

```

MANAGEMENT CLASS DEFINE                                     Page 1 of 8

SCDS Name . . . . . : SYS2.ITSCPLEX.DFSMS.SCDs
Management Class Name : MC00FIX2

To DEFINE Management Class, Specify:

Description ==> _____
                ==> _____

Expiration Attributes
  Expire after Days Non-usage . . . . . NOLIMIT      (1 to 93000 or NOLIMIT)
  Expire after Date/Days . . . . . 7                (0 to 93000, yyyy/mm/dd or
  NOLIMIT)

Retention Limit . . . . . NOLIMIT      (0 to 93000 or NOLIMIT)

```

Figure 5-3 SMS MC with 7 days fixed expiration

The SMS Management Class shows the EXPDT retention method and empty spaces for the WHILECATALOG option. See Figure 5-4.

```

MANAGEMENT CLASS DEFINE                                     Page 8 of 8
SCDS Name . . . . . : SYS2.ITSCPLEX.DFSMS.SCD
Management Class Name : MC00FIX2

To DEFINE Management Class, Specify:

Tape Volume Attributes
Retention Method . . . . . EXPDT           (VRSEL, EXPDT or blank)
Volume Set Management Level _____ (VOLUME, FIRSTFILE, SET or blank)

Tape Data Set Attributes
Exclude from VRSEL . . . . . _           (Y, N or blank)
Retain While Cataloged . . _____ (ON, OFF, UNTILEXPIRED or blank)

```

Figure 5-4 SMS MC with EXPDT and no WHILECATALOG value

By using this method, the data sets and the volumes they are on expire in DFSMSrmm on a specific date 7 days after creation, although that date can be extended by using the EXPIRE AFTER DAYS NON-USAGE value in addition to or in place of the EXPIRE AFTER DATE/DAYS if needed.

DFSMSrmm uses the longest of the two expiration dates that are determined by these two values.

### 5.3.2 Using the WHILECATALOG function

Another method to manage LWORM volumes in DFRMM is to make retention decisions that are based on the catalog entry for the data sets that are on the LWORM volume. This function is called WHILECATALOG function (WC). Two choices are available in the EXPDT retention method in DFSMSrmm, WHILECATALOG(EXPIRED) and WHILECATALOG(ON).

#### WHILECATALOG(EXPIRED)

WHILECATALOG(EXPIRED) can be abbreviated as WC(UX). WC(UX) retains the data sets on a volume until the expiration date is reached or the data sets on the volume are no longer cataloged. With WC(UX), the catalog entry that is being deleted or the expiration date being met can cause the data set and volume to expire.

WC(UX) is *not* a good choice for managing LWORM volumes. When the data sets are no longer cataloged, their expiration date is dynamically adjusted to be the current date plus the value of the CATLGDAYS parameter as specified in the EDGRMMxx parmlib member. The default of CATLGDAYS is 2.

Therefore, when the catalog entry of the last data set on a volume that is managed with WC(UX) is deleted, the expiration date for that data set and the volume is moved up. The new expiration date can be before the fixed duration that is specified in the TS7700 settings for the LWORM Data Class.

## WHILECATALOG(ON)

WHILECATALOG(ON) can be abbreviated as WC(ON). WC(ON) is a logical choice for managing data sets on LWORM volumes.

This function retains a data set, and the volume it is on, if the data set is cataloged and the expiration date is not passed.

With WC(ON), after the catalog entry is deleted, RMM compares the current expiration date and the value of CATLGDAYS and uses the greater of the two values to set the expiration date.

After that expiration date passes, the data set is not retained and the volume can be expired. The expiration date can be set by Data Class, Management Class, or JCL. If that date is equal to or greater than the fixed duration value for retaining LWORM volumes as specified in the TS7700, WC(ON) can be used effectively.

One option is the DFSMSrmm DEFAULT TABLE, which can be used with RETPD and OVERRIDE to overrule any JCL passed expiration date to ensure a value greater than the TS7700 fixed duration is used.

**Note:** The settings in the TS7700 for LWORM retention are a minimum value. The settings in the tape management system might keep a volume for longer than what is specified in the TS7700 Data Class.

## 5.4 Keeping LWORM volumes permanently

With the TS7700 LWORM retention function, it is possible that LWORM volumes are to be kept permanently. In such a case, DFSMSrmm can be used to keep a data set and volume permanently from the host perspective in any of the following ways:

- ▶ Pass an expiration date of 99365 or 99366 on the JCL or specified in Data Class
- ▶ Use MCATTR options in EDGRMMxx parmlib and assign a Management Class with EXPIRE AFTER DATE/DAYS and EXPIRE AFTER DAYS NON-USAGE set to NOLIMIT
- ▶ Assign the EDM(YES) attribute to a volume with RMM CV command or set by a qualified EDM program, such as DFHSM or OAM, which keeps the data set until the program turns off this attribute or a user issues the RMM CV NOEDM command
- ▶ Use the RMM CV *volser* HOLD command to keep a volume from being expired
- ▶ Use a Default Table entry to set RETPD(PERMANENT) for LWORM data sets that are based on job name, data set name, or program name

An older method of retention in DFSMSrmm can also be used to manage LWORM volumes. A VRS can be assigned to retain a data set and volumes permanently, retain for a specific number of days, or retain with a combination of days and catalog entry.

The VRS must not include the UNTIL EXPIRED option or IGNORE EXPIRY DATE option if it is to manage LWORM data with the TS7700 LWORM retention functions.





# General Data Protection Regulation changes

This chapter discusses the changes in DFSMSrmm to support the General Data Protection Regulation (GDPR) legislation and includes the following topics:

- ▶ 6.1, “Overview” on page 34
- ▶ 6.2, “DSNEXPIRE command option” on page 34
- ▶ 6.3, “FORCEEXPIRE command” on page 35

## 6.1 Overview

In V2R2, DFSMSrmm introduced support for the new European GDPR legislation in the fix for APARs [OA54871](#) and [OA55827](#). Because of the architecture of tape data, a data set cannot be deleted and removed from a tape as it can with disk-based data sets.

The intent of this new function was to block access to data sets on tape that expired but still physically exist on a tape. The effect is that DFSMSrmm prohibits read or write access to a data set that is no longer retained.

## 6.2 DSNEXPIRE command option

The format for the command is DSNEXPIRE(BLOCK | NONE).

For this support to be in effect, the DFSMSrmm parmlib member EDGRMMxx must be updated with DSNEXPIRE(BLOCK). This support blocks access to data sets that are expired after they are no longer retained as a vital record. Newly created data sets that are managed under the VRSEL retention method, but have not yet been processed during VRSEL, are unaffected.

Bypass Label Processing cannot be used to read or write expired data sets when DSNEXPIRE(BLOCK) is in effect.

If RMM is running, data sets that are expired are inaccessible. The following categories of data sets are considered expired:

- ▶ Data sets that are managed with the EXPDT retention method or are excluded from VRSEL processing that have an expiration date and time that are *before* the current date and time.
- ▶ Data sets that are managed with the VRSEL retention method, but were released from vital retention and have an expiration date and time that are *before* the current date and time.

Individual data set access can be blocked by issuing RMM CHANGEDATASET FORCEEXPIRE, even when DSNEXPIRE(BLOCK) is not in effect.

After this attribute is applied, ForceExpire appears next to their expiration date. The FORCEEXPIRE attribute can also be seen in the DFSMSrmm Data Set Details page in the DFSMSrmm ISPF windows.

If NONE is specified, the data is accessible even if the data set expired. If a user modifies a data set by using the RMM CHANGEDATASET dsname FORCEEXPIRE command, the data set becomes inaccessible, even if the data set was created with DSNEXPIRE(NONE).

When BLOCK is specified, expired data sets that are not retained by a vital record specification (VRS) cannot be accessed. However, newly created data sets are not blocked until the first run of EDGHSKP with VRSEL has a chance to retain them.

When BLOCK is specified, some data sets that are being retained cannot be blocked if the data set matches the following criteria:

- ▶ If a data set includes a HOLD attribute of YES
- ▶ If a data set has the EDM flag set to YES
- ▶ If a data set is cataloged and the WHILECATALOG operand is set to ON

When BLOCK is specified, data sets on a Not Labeled (NL) tape are blocked, even if the referenced data set features an incorrect name. This issue occurs because the data set matches the sequence number that is stored in the RMM control data set (CDS). Bypass label processing (BLP) cannot be used to read or write expired data sets when DSNEXPIRE(BLOCK) is set.

## 6.3 FORCEEXPIRE command

The format for this command is `RMM CHANGEDATASET dsname FORCEEXPIRE`.

As an alternative to setting DSNEXPIRE(BLOCK) in the EDGRMMxx parmlib member, customers can consider issuing the command `RMM CHANGEDATASET dsname FORCEEXPIRE` on a per data set basis.

When FORCEEXPIRE is set by using a CHANGEDATASET command, it is listed in the output of the RMM LISTDATSET command, next to the Expiration Date as '(ForceExpire)'.

If a data set is expired by using the CHANGEDATASET command, the following message is issued if a user tries to access the data set:

```
EDG4067I VOLUME volser REJECTED. DATASET BLOCKED BY FORCEEXPIRE
```

If a user must regain access to a data set against which FORCEEXPIRE was issued, the user must issue the following command:

```
RMM CHANGEDATASET dsname REMOVEEXPIRE FORCE
```





## **z/OSMF GUI plug-in for DFSMSrmm**

This chapter discusses the DFSMSrmm z/OSMF plug-in in the following sections:

- ▶ 7.1, “Overview” on page 38
- ▶ 7.2, “Default Table Policy Helper” on page 39
- ▶ 7.3, “Screen Settings” on page 40
- ▶ 7.4, “Translations” on page 40

## 7.1 Overview

The RMM z/OSMF plug-in is included in V2R4 DFSMSrmm, which includes fixes for APARs [OA59727](#) and [OA59499](#). The plug-in was enhanced with the fix for APAR [OA62705](#).

The interface into RMM on the z/OSMF platform provides a way to manage and interact with RMM and its data on IBM Z®. The plug-in provides a way to display traditional ISPF windows on a web browser.

You can use the plug-in to discover relationships, view metadata of multi-volume data sets, and browse active configuration for your Defaults Table in a web interface.

The web interface in its current state is not meant to replace the green-screen interface into your system library management. Instead, it allows for quick management without the need for an emulator connection for the user.

The RMM z/OSMF interface is intended for users who want to monitor their systems, gain quick insights, or create reports about filtered data. Query results can be downloaded in a .csv format by using a single click without the need for additional scripting. As adoption increases, the plan is to add more functions to the plug-in to help with managing libraries in a web interface.

Some of the recent features that were added are that the Default Table tab now includes a Policy Helper feature that displays default RMM policies that are assigned to data sets that were created with specified criteria. This feature is also available in the DataSets tab by right-clicking a data set row and selecting **Show Policies**.

## 7.2 Default Table Policy Helper

You can view the Default Table as shown in Figure 7-1 against any of the following conditions:

- ▶ Data set name

Specify a fully qualified data set name or a data set name mask with wildcards. For example, “\*\*\*” can be specified to indicate any data set name that is considered a match.

- ▶ Jobname

Specifies the name of a job that created the data set. The default is “\*”, which specifies that any jobname is considered a match.

- ▶ KEYDATE

Specifies the KEYDATE on the data set.

- ▶ Program Name

Specifies the name of a job step program that is running at the time that the data set was opened for output. The default is “\*”, which specifies that any program name is considered a match.

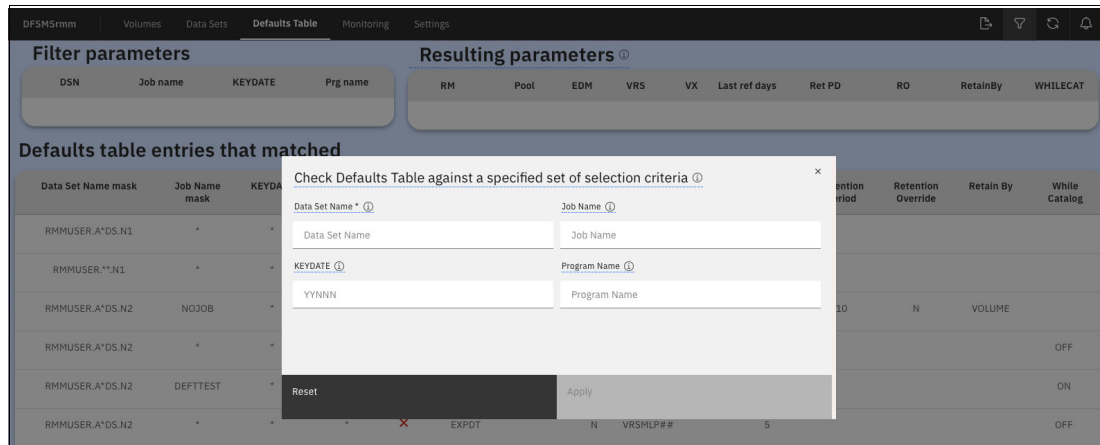


Figure 7-1 Default Table tab in GUI

For more information about this feature, see Chapter 22 of the [z/OS DFSMSrmm Implementation and Customization Guide](#).

For more information, see the following resources:

- ▶ [OA62705](#): New Function: z/OSMF RMM Plug-in Improvements
- ▶ [OA59727](#): New Function: z/OSMF RMM Plug-in Improvements
- ▶ [OA59499](#): New Function: z/OSMF RMM Plug-in

## 7.3 Screen Settings

User's can set settings on a per user and per screen basis as shown in Figure 7-2. The initial release of these settings allows users of version 3.1 to define which columns in a data table are visible without affecting the personal screen settings of other users. Also, only visible columns are exported to .csv when you use the export button. This does not affect how the Defaults Table Policy Helper determines default RMM policies that are assigned to Data Sets.

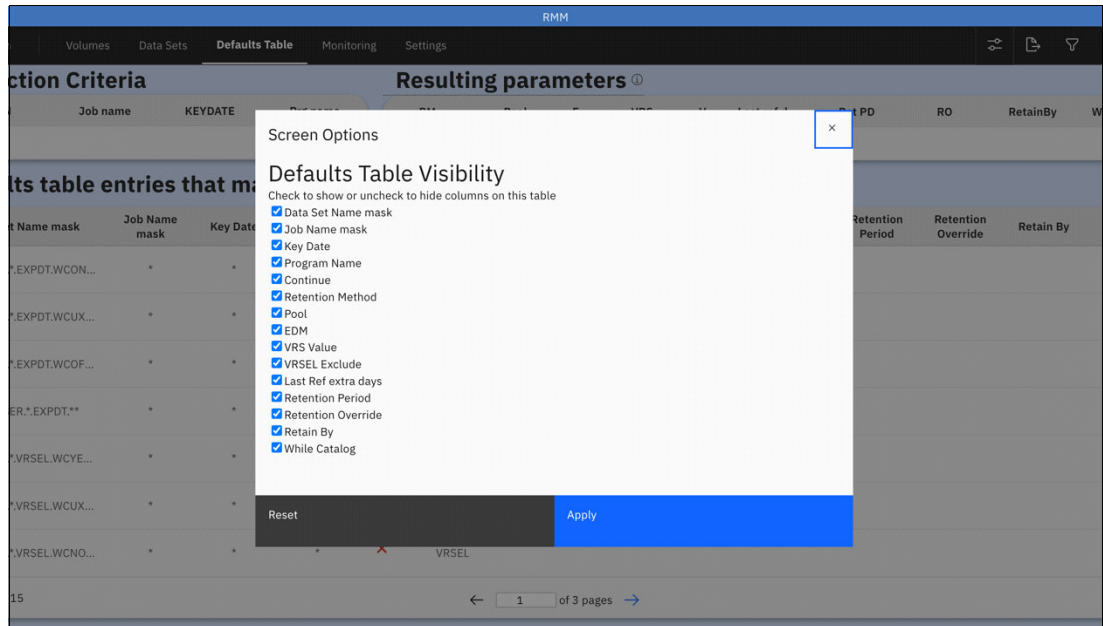


Figure 7-2 User's Screen Visibility Setting

## 7.4 Translations

Also, included in version 3.1, when you install the RMM z/OSMF plugin by using the z/OSMF "Import Manager" facility, you can select to view the interface in either Japanese or English. For detailed and up-to-date instructions in the plug-ins, see the readme file in the OMVS file system where the plugin is installed by SMP/E. At the time of writing, the plugin can be installed with only one language at a time. The language cannot be dynamically selected from the web interface itself. Translated text is also passed to the .csv export function, with all translated text from the data table that is also used in your data export.

When switching between languages, refresh the loaded assets by refreshing your browser between language packs. Otherwise, the web page might load incorrectly. As a reminder, it is recommended to use the z/OSMF plug-in in Chrome or Firefox.

Translation requests for additional languages (or additional enhancements) can be submitted through an IBM IDEA request by using the [IBM Ideas Portal](#).

Inconsistencies or errors found in translation can be reported to RMM by using traditional service portals or representatives.





# ONLYIF system option for EDGRMMxx

In this chapter, the ONLYIF system option is discussed along with examples of its use.

This chapter includes the following topics:

- ▶ 8.1, “Overview” on page 42
- ▶ 8.2, “Sharing RMM parmlib members” on page 42
- ▶ 8.3, “DEFAULT TABLE support” on page 44
- ▶ 8.4, “Backup procedure example” on page 458.5, “CATSYSID example” on page 46

## 8.1 Overview

In V2R4, the ONLYIF system option was introduced to be used in the DFSMSrmm parmlib members EDGRMMxx and EDGDEFxx. The ONLYIF system option cannot be used with earlier releases.

For releases prior to V2R4, you can use the system option called MEMBER. A combination of MEMBER and ONLYIF can be used in the same parmlib member. Such a configuration might be wanted if a sysplex exists that includes LPARs that are on earlier releases of z/OS than V2R4.

For more information about the MEMBER and ONLYIF system options, see Chapter 8: “Using the parmlib member EDGRMMxx” of the [DFSMSrmm Implementation and Customization Guide](#).

## 8.2 Sharing RMM parmlib members

The ONLYIF system option can make it easier to share RMM parmlib members among multiple instances of RMM in a SYSPLEX. The ONLYIF operand causes the parameter that follows the ONLYIF statement to be run only on the system that includes the same system name as specified in the ONLYIF command.

All other systems disregard the statements that follow the **ONLYIF** command. If duplicate operands are specified, DFSMSrmm uses the last value that was specified.

Multiple operands are possible with each ONLYIF statement. When you issue the ONLYIF statement, no continuation character is needed. However, when multiple operands are used inside an ONLYIF statement, the last operand on each line needs a continuation character except for the last line. See Example 8-1.

*Example 8-1 ONLYIF statement*

---

```
ONLYIF SYSNAME(ZOS24)
      OPTION JOURNALFULL(70) -
      DEFTABLE(00)
ONLYIF SYSNAME(PRD24)
      OPTION JOURNALFULL(75) -
      DEFTABLE(00)
ONLYIF SYSNAME(PRD23)
      LOCDEF LOCATION(OFFSITE2) TYPE(STORAGE) MEDIANAME(3480) -
      MANAGEMENTTYPE(NOINS) AUTOMOVE(YES)
```

---

**Note:** When DFSMSrmm sets a default value for a parameter that is under the OPTION statement, the ONLYIF system options must be after the global OPTION statements. Otherwise, the global defaults might override the ONLYIF system options. DFSMSrmm uses the last assignment of a value.

You can check the &SYSNAME system symbolic value by using the D SYMBOLS command. For example, the SYSNAME value is ZOS24 on the LPAR is shown in Figure 8-1.

```
00000290  D SYMBOLS
00000090  IEA007I STATIC SYSTEM SYMBOL VALUES 421
00000090  &SYSALVL.           = "2"
00000090  &SYSCLONE.         = "24"
00000090  &SYSNAME.          = "ZOS24"
00000090  &SYSOSLVL.         = "Z1020400"
00000090  &SYSPLEX.          = "ITSCPLEX"
00000090  &SYSR1.            = "Z24RS1"
00000090  &JESPARM.          = "JES2PARM"
00000090  &OMVS.              = "PS"
00000090  &PRISUBSY.         = "JES2"
00000090  &REL.               = "24"
00000090  &SMFID.             = "ZS24"
00000090  &SUFFIX.           = "PS"
00000090  &ZOSMFS.           = "CONNECT"
```

Figure 8-1 Defined System Symbols

Any ONLYIF SYSNAME commands that are in the EDGRMMxx or EDGDEFxx members on this LPAR are run if SYSNAME(ZOS24) is specified.

For example, the JOURNALFULL(70) and DEFTABLE(00) statements that are shown in Example 8-1 on page 42, are processed only on the LPAR that is shown in Figure 8-1 and *not* on any of the other LPARs that are sharing this EDGRMMxx member.

On ZOS24, DFSMSrmm includes a Defaults Table when it initializes and sets the JOURNALFULL setting to 70%. On LPAR PRD24, the same Defaults table is used, but the JOURNALFULL setting is 75%.

On LPAR PRD23, no Defaults Table is used because it is not supported at that level. Also, JOURNALFULL is not specified, but another location is defined named OFFSITE2, which is unknown to the other two LPARs.

As shown in Example 8-2, multiple ONLYIF statements are used for each operand. In this example, OPTION MEMBER is also used. It can be coded by using a system symbol, for example, OPTION MEMBER(&SYSCclone).

*Example 8-2 MULTIPLE ONLYIF*

---

```

OPTION  OPMODE(P)                /* PROTECT MODE          */ -
      EDM(YES)                   /* EXTERNALLY MANAGED    */ -
ONLYIF  SYSNAME(ZOS24)
      OPTION JOURNALFULL(70)
ONLYIF  SYSNAME(ZOS24)
      OPTION DEFTABLE(00)
ONLYIF  SYSNAME(ZOS24)
      OPTION MEMBER(24)

```

---

The symbol must resolve to two characters. For example, if the symbol is &SYSCclone=24, the member option uses EDGRMM24.

In this example, only the LPAR ZOS24 sets the JOURNALFULL value, initializes by using the DEFAULT TABLE that is described in EDGDEF00, and imports the contents of a secondary RMM parmlib member that is called EDGRMM24.

All LPARs then include OPMODE(P) and EDM(YES) in effect. However, any options that are specified in EDGRMM24 overwrite the options that are specified in this EDGRMMxx parmlib member. No other LPAR that is reading this EDGRMMxx parmlib member includes the three items that are listed under the ONLYIF statements.

## 8.3 DEFAULT TABLE support

ONLYIF can also be used with the DEFAULT TABLE support in DFSMSrmm. The same rules and functions apply. As shown in Example 8-3, which is an EDGDEFxx member, only the first data set mask for GDG data sets whose high-level qualifier begins with K are assigned on LPAR ZOS24. The remainder of the coded attributes applies on all LPARs that read this DEFAULT TABLE, including ZOS24.

*Example 8-3 DEFAULT TABLE*

---

```

ONLYIF  SYSNAME(ZOS24)
      DEFAULT DSN(K*.*.G%%V%%) WHILECATALOG(ON) RM(EXPDT)
DEFAULT DSN(TEST.RMXEXPDT.*.WCON)WHILECATALOG(ON) RM(EXPDT)
DEFAULT KEYDATE(99000) VRSVAL(D99000)
DEFAULT KEYDATE(99365) VRSVAL(D99365)
DEFAULT KEYDATE(99366) VRSVAL(D99365)

```

---

## 8.4 Backup procedure example

A good use of the ONLYIF option is how to submit a backup procedure to empty the journal data sets from multiple systems that are based on different JOURNALFULL thresholds on each system. In this way, if one of the LPARS is down, another LPAR can submit it for you.

The following best practice approach is described in the [z/OS DFSMSrmm Customization and Implementation Guide](#):

- ▶ Specify a value for JOURNALFULL operand on the OPTION command in parmlib as described in the section Defining system options: OPTION. If you do not specify a value, DFSMSrmm uses a value of 75%. Use only a single system to trigger a backup that uses the journal threshold.
- ▶ If multiple systems exist 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, you can specify a different threshold on each system if multiple systems exist.  
  
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 that 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 poor choice for automatic backup because DFSMSrmm checks only the journal threshold when a request is processed.

To implement the best practices, you can use ONLYIF in a single EDGRMMxx parmlib member that is used by z/OS V2R4 and newer systems in the sysplex. For example, if LPARS SYSA, SYSB, and SYSC are identified as SYSNAME, you can then code your EDGRMMxx as shown in Example 8-4.

*Example 8-4 EDGRMMxx for different JOURNALFULL thresholds*

---

```
OPTION  OPMODE(P)-
BACKUPPROC (DFRMMBKP)
...
ONLYIF  SYSNAME(SYSA)
        OPTION  JOURNALFULL(80)
ONLYIF  SYSNAME(SYSB)
        OPTION  JOURNALFULL(85)
ONLYIF  SYSNAME(SYSC)
        OPTION  JOURNALFULL(90)
...

```

---

As shown in Example 8-4, it is important to specify only the system-specific JOURNALFULL commands. If a duplicate JOURNALFULL command was found later in the parmlib, the system-specific values are ignored.

Another way to solve this issue is to have only one system call the backup procedure. You can then code your EDGRMMxx as shown in Example 8-5.

*Example 8-5 EDGRMMxx with only one system having BACKUPPROC*

---

```
OPTION OPMODE(P)
JOURNALFULL(80) -
...

ONLYIF SYSNAME(SYSA)
        OPTION BACKUPPROC(DFRMMBKP)
```

---

As shown in Example 8-5, it is important to specify only the single system-specific BACKUPPROC statement. That way, only that system can call the backup procedure.

## 8.5 CATSYSID example

In this example, two production systems, PRD1 and PRD2, share catalogs with each other, but not with test and development systems. However, you want to share one RMM control data set (CDS) and one EDGRMMxx between them. You can use the code example that is shown in Example 8-6.

*Example 8-6 Use of ONLYIF with CATSYSID*

---

```
ONLYIF SYSNAME(PRD1)
        OPTION
            CATSYSID(PRD1,PRD2) -
ONLYIF SYSNAME(PRD2)
        OPTION
            CATSYSID(PRD1,PRD2) -
ONLYIF SYSNAME(TEST)
        OPTION
            CATSYSID(TEST,DEV1,DEV2) -
ONLYIF SYSNAME(DEV1)
        OPTION
            CATSYSID(TEST,DEV1,DEV2) -
ONLYIF SYSNAME(DEV2)
        OPTION
            CATSYSID(TEST,DEV1,DEV2) -
```

---

By using this coding, EDGHSKP can be run on PRD1 and PRD2 to return production volumes to scratch and not affect any test or development systems. Conversely, EDGHSKP can be run on TEST, DEV1, or DEV2 and not affect any production system volumes.



# CDSFULL command for EDGRMMxx

This chapter includes a discussion of the CDSFULL operand for the DFSMSrmm EDGRMMxx parmlib member.

## Overview

The CDSFULL operand is new for z/OS V2R4 DFSMSrmm. CDSFULL is used to monitor the use of the space as a percentage in the RMM control data sets (CDSs). The syntax of this operand is CDSFULL(*nn*) where *nn* is a value of 0–99.

The percentage full calculation by RMM is determined by the High Used Relative Byte Address (HURBA) divided by the High Available Relative Byte Address (HARBA).

If the RMM CDS takes a Control Interval (CI) split such that this calculated percentage equals or exceeds the specified value, DFSMSrmm issues message EDG2120W. Whenever that DFSMSrmm starts thereafter, EDG2120W again is issued until the RMM CDS is REORGed (EDGBKUP with BACKUP(REORG)) or data is restored into a larger, newly allocated CDS. If you specify a value of 0, DFSMSrmm issues no warnings on that system.

The monitoring by DFSMSrmm does not consider the ability to take a secondary extent. Even if the CDS full percentage shows 100%, the CDS does not fail any insert of records if the CDS is defined with a secondary extent and room exists on the volume to take such an extent.

A best practice is to allocate the CDS in a single extent at least 20% larger than is needed to allow for growth and then to monitor with CDSFULL at a value of 90% or less. This configuration provides a grace period before the CDS fills up so that a REORG can be scheduled.

After a REORG, the CDS is compressed, as noted by an apparent decrease in the percentage of used space. However, the next several days show a rapid increase in the percentage full for the CDS files because record insertions cause Control Area (CA) and Control Interval (CI) splits.

You can expect a rapid increase in used space after a REORG, but the level stabilizes after a few days of normal processing. When allocated at the correct size, CDS use typically stabilizes at approximately 80% used. If CDS use stabilizes at a value over 90%, consider allocating a larger CDS.


For more information about increasing the size of the RMM CDS, see [Changing the size of the control data set and journal](#).

When you return volumes to scratch status in RMM, the data set records are not freed until that volume is reused. For that reason, deleting a backup data set does not increase available space in the RMM CDS. These records are retained if the expiration of the data is premature and must be recovered.

Although CA Reclaim helps, a need still exists for a periodic REORG to fully reclaim space in all partially used CAs.

For more information about a REORG of the RMM CDS, see [Reorganizing the control data set](#).





# Global Resource Serialization recommendations for RMM data sets

This chapter describes the recommended placement and settings for Global Resource Serialization (GRS) for DFSMSrmm-related data sets. It includes the following topics:

- ▶ 10.1, “RMM control data sets and journals” on page 50
- ▶ 10.2, “GRS Settings” on page 50

## 10.1 RMM control data sets and journals

To avoid deadlocks and contention, ensure that the volume where the DFSMSrmm control data set (CDS) is stored does not contain any other data sets on it and is not eligible for new data set allocations. This suggestion also pertains to the volume where the RMM journals are stored, which must be a separate volume from where the RMM CDS is stored.

One way to accomplish isolation of the CDS is by creating an SMS Storage Class and Storage Group that contains a single large volume. Then, use a filter list in the Automatic Class Selection (ACS) routines for the specific names of the RMM CDS data set to assign that storage class. That way, if the RMM CDS is processed by using BACKUP(REORG), as must be done occasionally to mitigate CA/CI splits even with CA RECLAIM, the CDS is allocated back onto the same volume.

To keep the RMM CDS isolated on this volume ensure that no other assignment occurs to this storage class.

For the RMM journals, it is a best practice to create an additional separate SMS storage class and storage group. However, this storage group can have many volumes, if wanted. Other RMM-related data sets for inventory management and reporting purposes can also be kept in this additional storage group.

To avoid contention on VTOC and VVDS, which might affect DFSMSrmm operations, exclude programs that interrogate DASD for reporting purposes from processing the volumes that contain the RMM CDS and RMM journals.

Also, DEFRAG does not need to be used nor does the RMM CDS need to be backed up outside DFSMSrmm because only backups that are created by DFSMSrmm utilities can be forward recovered by RMM.

## 10.2 GRS Settings

The main serialization resource for the RMM CDS is QNAME SYSZRMM RNAME MASTER.RESERVE and MASTER.RESERVE.cdsid. The handling of the SYSZRMM MASTER.RESERVE resource depends on how the SYSIGGV2 and SYSZVVDS resources are being treated.

If you are converting the SYSIGGV2 resource for the ICF Catalog that contains the RMM CDS, consider converting the SYSZRMM MASTER.RESERVE resource and SYSZVVDS resource for the specific volume on which the RMM CDS is stored.

If for any reason the ICF catalogs or the RMM CDS are shared across SYSPLEX boundaries, these resources must be left as reserves, which is the default setting. However, in that case, you can put SYSZRMM MASTER.RESERVE into the EXCL RNL to prevent the addition of a SYSTEMS enqueue, which is not beneficial across a sysplex boundary.

The resource SYSZRMM MASTER.RESERVE is managed as a hardware reserve by default. In some circumstances, this management can cause contention or deadlocks with Catalog and DFHSM.

If the DASD that contains the catalogs and the RMM CDS are shared only within the boundaries of a GRSPLEX, you can create entries in the SYSZRMM MASTER.RESERVE, SYSIGGV2, and SYSZVVDS resources can be converted from real hardware reserves by creating entries in the GRS CONVERSION RNL.

To avoid the following issues, it is considered a best practice to change from using reserves with a scope of system to using enqueues with a scope of systems:

- ▶ Adding an extent to the RMM CDS requires updates to the VVDS. Because of the timing of events, a lockout can result if SYSZVVDS uses reserves, and RMM does not. Therefore, use the same method for SYSZRMM MASTER.RESERVE as SYSZVVDS.
- ▶ Another problem relates to one system that is holding an enqueue on SYSIGGV2 for a catalog request and requires access to the RMM CDS volume. However, RMM holds a reserve on the CDS volume and issues a catalog request, which results in a lockout condition.

In general, it is a best practice to convert the RESERVE to a SYSTEMS enqueue when not sharing DASD across a sysplex boundary. In a GRS STAR configuration, sharing by way of global enqueues results in better performance and less contention or deadlocks.

Type GENERIC must be used to pick up the MASTER.RESERVE and the MASTER.RESERVE.cdsid resources, as shown in the following example:

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

For IBM OMEGAMON®, customers see PTF UJ08619 for APAR [OA63258](#), which provides a disablement function for RMM CDS data sets so that OMEGAMON for Storage does not process the RMM CDS and contribute to contention or deadlocks.





# Disaster Recovery and Disaster Recovery testing with the TS7700

This chapter describes disaster recovery (DR) and DR testing with DFSMSrmm. This chapter is written assuming that an IBM SMSTAPE library in the TS7700 family is being used, but much of the content also applies to non-IBM tape library processing.

This chapter includes the following topics:

- ▶ 11.1, “Overview” on page 54
- ▶ 11.2, “DFSMSrmm parmlib options” on page 54
- ▶ 11.3, “Volser range example” on page 54
- ▶ 11.4, “Additional safeguards” on page 57
- ▶ 11.5, “Handling scratch volumes for DR testing” on page 57
- ▶ 11.6, “Reading DR volumes back on PROD” on page 62
- ▶ 11.7, “Reading LWORM volumes during DR testing” on page 62
- ▶ 11.8, “Limited Housekeeping function” on page 63

## 11.1 Overview

When you consider how DFRMM can play a role in disaster recovery (DR) and the testing of DR procedures, remember the following concepts:

- ▶ How to configure a remote or DR site for reading tape volumes that were created in production while also protecting those volumes from being overwritten during a DR test.
- ▶ To simulate a batch workload at a DR site, a range of tapes must be made available to create data sets. Action must be taken at the end of each DR test cycle to leave the environment ready for future DR tests.

## 11.2 DFSMSrmm parmlib options

The DFSMSrmm parmlib options OPENRULE and PRTITION, as described in Chapter 12, “ePartitioning examples with OPENRULE and PRTITION parameters” on page 65, are the rules that are created for partitioning your environment to read and protect tapes that are owned by another LPAR.

PRTITION limits entries and returns of cartridges to scratch processing, and can prevent tapes from being added to the RMM control data set (CDS) when read. OPENRULE allows or disallows the opening of a tape volume for reading (INPUT) or writing (OUTPUT).

## 11.3 Volser range example

In the following examples, any volser that matches DR\* is used for scratch mounts at the DR. Any other is treated as a read-only tape because of VOLUME(\*). The OPENRULE and PRTITION statements that are presented here are in the EDGRMMxx parmlib member that is used with the DR host LPAR. These examples assume that only SMS tape is being used. More customization is required if non-SMS tape is used.

This first set of rules in Example 11-1 allows the DR Host to read any volume that is defined in the RMM CDS, but allow only write access to volumes that begin with prefix DR. These rules keep the DR host from overwriting any data that belongs to another host that is sharing the tape environment. If volumes that belong to other LPARs must be prevented from being read, the second rule can be changed to INPUT(REJECT).

*Example 11-1 Preventing data overwrites of non-DR volumes*

---

```
OPENRULE VOLUME(DR*) TYPE(RMM) OUTPUT(ACCEPT) INPUT(ACCEPT)
OPENRULE VOLUME(*) TYPE(RMM) OUTPUT(REJECT) INPUT(ACCEPT)
```

---

This second set of rules shown in Example 11-2 allows cartridge entry processing and return to scratch(EXPROC) functions against only volumes that are prefixed with DR and disallows such actions on any other volume ranges, even if RMM is aware of them. This method is one of the ways to prevent a housekeeping job that is run during a DR TEST from returning volumes to scratch that are owned by other LPARs. This issue is especially important when catalogs and the RMM CDSs are not fully shared on common DASD with the DR host.

*Example 11-2 Prevent return to scratch of non-DR volumes*

---

```
PRRTITION VOLUME(DR*) TYPE(RMM) SMT(ACCEPT) NOSMT(IGNORE)
PRRTITION VOLUME(*) TYPE(RMM) SMT(IGNORE) NOSMT(IGNORE)
```

---

The RMM CDS and catalogs that are used during a DR test are typically point-in-time copies and not updated with production. Therefore, the use of this information for expiration processing can cause data loss.

The cartridge entry limiting ensures that volumes that are defined with only DR\* must be moved from insert category to scratch category by the DR host LPAR. The purpose is that when those volumes are defined or inserted, they are not accidentally added into the production scratch pool, assuming unique DEVSUPxx categories are in use as recommended:

This third set of rules that are shown in Example 11-3 disallows any read/write/cartridge entry or EXPROC against volumes that are not defined in the RMM CDS. If you want to read a volume not in the RMM CDS, but not have it added to the RMM CDS and managed by RMM, the OPENRULE can be changed to INPUT(ACCEPT) and suitable IBM RACF rule put into place. This function might be needed to read a tape from another LPAR or site, as shown in the following example:

*Example 11-3 Prevent usage of volumes not defined in RMM CDS*

---

```
OPENRULE VOLUME(*) TYPE(NORMM) OUTPUT(REJECT) INPUT(REJECT)
PRRTITION VOLUME(*) TYPE(NORMM) SMT(IGNORE) NOSMT(IGNORE)
```

---

Any other host that shares the library or TS7700 Grid must use a similar pattern. For example, if two other LPARs called PROD and TEST host use this library and use volume prefix P\* for PROD and volume prefix T\* for test.

As shown in Example 11-4, add the following rules to EDGRMMxx parmlib member on PROD.

*Example 11-4 Adding rules on LPAR PROD*

---

```
OPENRULE VOLUME(P*) TYPE(RMM) OUTPUT(ACCEPT) INPUT(ACCEPT)
OPENRULE VOLUME(*) TYPE(RMM) OUTPUT(REJECT) INPUT(ACCEPT)
PRRTITION VOLUME(P*) TYPE(RMM) SMT(ACCEPT) NOSMT(IGNORE)
PRRTITION VOLUME(*) TYPE(RMM) SMT(IGNORE) NOSMT(IGNORE)
```

---

The rules in Example 11-4 allow read/write/cartridge entry and EXPROC fully to volumes that begin with P, and the ability to read but not write or scratch any other volume range that is defined to the RMM CDS.

Example 11-5 shows the third set of rules that disallows any read/write/cartridge entry or EXPROC against volumes that are not defined in the RMM CDS.

*Example 11-5 Disallow read/write/cartridge entry or EXPROC against volumes*

---

```
OPENRULE VOLUME(*) TYPE(NORMM) OUTPUT(REJECT) INPUT(REJECT)
PRRTITION VOLUME(*) TYPE(NORMM) SMT(IGNORE) NOSMT(IGNORE)
```

---

Similarly, Example 11-6 shows code that you can include on TEST to EDGRMMxx parmlib member.

*Example 11-6 Adding rules on LPAR TEST*

---

```
OPENRULE VOLUME(T*) TYPE(RMM) OUTPUT(ACCEPT) INPUT(ACCEPT)
OPENRULE VOLUME(*) TYPE(RMM) OUTPUT(REJECT) INPUT(ACCEPT)
PRRTITION VOLUME(T*) TYPE(RMM) SMT(ACCEPT) NOSMT(IGNORE)
PRRTITION VOLUME(*) TYPE(RMM) SMT(IGNORE) NOSMT(IGNORE)
OPENRULE VOLUME(*) TYPE(NORMM) OUTPUT(REJECT) INPUT(REJECT)
PRRTITION VOLUME(*) TYPE(NORMM) SMT(IGNORE) NOSMT(IGNORE)
```

---

When a new volume is defined or inserted to the library and a broadcast is sent out to all attached LPARS, OAM calls RMM, and it uses the volsr prefix to check the PRRTITION rules in EDGRMMxx parmlib member. This broadcast is only processed on LPARS where the OAM address space is active and the CBRUXENT exit is enabled. Another method to control cartridge entry is to issue LI DISABLE,CBRUXENT on all systems except on the LPAR on which you want to assign the scratch category to volsers inserted into the TS7700. After the test, this exit can be enabled with LI RESET,CBRUXENT. It is a good practice to leave it disabled except when adding new volumes. Volumes that are inserted with the exit disabled are left in an insert category but can then be processed by issuing LI RESET,CBRUXENT on the appropriate host LPAR where they will be used. This exit stays disabled across IPL and must be explicitly re-enabled by using the LI RESET,CBRUXENT command.

With the previous example rules in place, the following volume prefixes are defined:

DR	it is moved to the DR scratch category
P	it is moved to the PROD scratch category
T	it is moved to the TEST scratch category

In this way, volume ranges are then mapped to each LPAR. Also, they each maintain a unique scratch pool, and no LPAR expires data that is managed by another LPAR.



## 11.4 Additional safeguards

As an extra safeguard to the suitable partitioning and separation of the volumes to be managed by each host, customization is available in the EDGHSKP housekeeping job stream to limit the return to scratch processing function to a specific volume, volume list, or range of volumes.

Example 11-7 shows code that you can include into the step that is running EXPROC.

*Example 11-7 Code to include when running EXPROC*

---

```
//EXPROC EXEC PGM=EDGHSKP,PARM='EXPROC',REGION=4096K
//MESSAGE DD DISP=SHR,DSN=DFRMM.HSKP.MESSAGE
//REPORT DD DISP=SHR,DSN=DFRMM.HSKP.REPORT
//ACTIVITY DD DISP=SHR,DSN=DFRMM.HSKP.ACTIVITY
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
EXPROC VOLUMERANGES('DR0000':'DR9999')
```

---

Only the EXPROC function and VRSEL must be run on a earlier version of CDS, which results in volumes moving to pending release in error. However, all housekeeping functions can be moved in one step. EXPROC does not have to be the only function that is started in this step for the previous SYSIN DD to effectively limit return to scratch to the DR range of volumes.

## 11.5 Handling scratch volumes for DR testing

If a copy is made of the RMM CDS and tape configuration database (TCDB) each time a DR test is performed, then scratch volumes can be reused or newly defined for each test.

In an environment where the library at the DR site is owned and operated by the customer rather than rented, the scratch pool can be defined once and then reused repeatedly. This configuration reduces the start time to begin DR testing. The volumes can be predefined in the RMM CDS and OAM TCDB by using the following procedure:

1. Create the volumes first in the TCDB with IDCAMS, as shown in Example 11-8.

*Example 11-8 Creating volumes in the TCDB*

---

```
//SYSIN DD *
CREATE VOLUMEENTRY -
  (NAME(Vvolser) - Volser after the first V for volume record
  LIBRARYNAME(libname) - Composite library name
  USEATTRIBUTE(SCRATCH) -
  MEDIATYPE(MEDIA*) - Change to MEDIA1 or MEDIA2
  LOCATION(LIBRARY))
/*
```

---

In Example 11-9, the JCL creates a placeholder volser in the TCDB that does not yet exist in the library, as shown by the D SMS,VOLUME(DR0000) command in Example 11-9.

*Example 11-9 Creating a placeholder volser*

---

```
D SMS,VOLUME(DR0000) F OAM,D,VOL,DR0000,L=IBMUSER-Z
RESPONSE=ZOS24
CBR1180I OAM tape volume status: 735
VOLUME MEDIA STORAGE LIBRARY USE W C SOFTWARE LIBRARY
TYPE GROUP NAME ATR P P ERR STAT CATEGORY
DR0000 MEDIA2 *SCRATCH* ZOSVTS1 S NOERROR NOTAVAIL
-----
RECORDING TECH: 36 TRACK COMPACTION: UNKNOWN
SPECIAL ATTRIBUTE: NONE ENTER/EJECT DATE:
CREATION DATE: 2021-08-04 EXPIRATION DATE:
LAST MOUNTED DATE: LAST WRITTEN DATE:
SHELF LOCATION:
OWNER:
LM SG: NOTAVAIL LM SC: NOTAVAIL LM MC: NOTAVAIL LM DC: NOTAVAIL
LM CATEGORY: 0000
```

---

These steps must be performed for each volume to be added to the scratch pool.

2. Add each volume to the RMM inventory by using TSO subcommands that are based on their status in the TCDB as shown in Example 11-10.

*Example 11-10 Adding a volume to the RMM inventory*

---

```
RMM AV volser COUNT(number of entries) INIT(N) STATUS(VOLCAT)
MEDIANAME(medianame) -
MEDIATYPE(ECCST or CST) LOCATION(your libname)
```

---

Example 11-11 shows the same commands submitted as a batch job.

*Example 11-11 Adding a volume to the RMM inventory as a batch job*

---

```
//TSOBATCH EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
    RMM AV DR0000 COUNT(100) INIT(N) STATUS(VOLCAT) MEDIANAME(VTS1MD2) -
    MEDIATYPE(ECCST) LOCATION(ZOSVTS1)
```

---

The batch job in Example 11-11 on page 58 creates 100 “dummy” entries in the RMM CDS that look similar to the output of the TSO subcommand RMM LV *volser* in Example 11-12.

*Example 11-12 Results of batch job to create 100 placeholder entries in the RMM CDS*

---

```
Volume information:
Volume = DR0000 VOL1 = Rack = Owner =
Type = LOGICAL Stacked count = 0 Jobname =
Worldwide ID = WORM = N
Creation: Date = 08/04/2021 Time = 15:54:37 System ID = ZS24
Assign: Date = 08/04/2021 Time = 15:54:37 System ID =
User ID =
Expiration date = Original =
Expiration time = Datasets Kept By Catlg: 0
set by =
Retention date = Set retained = NO
Retention method= VRSEL
set by = CMD_DEF
retain by =
Data set name =
Volume status: EDM = N Hold = N File 1 Data set seq = 0
Status = SCRATCH Availability = Label = SL
Current label version = Required label version =
Media information: IBM
Density = * Type = ECCST Format = 36TRACK Compaction = *
```

---

When the OAM address space is started on the DR host or when the LI RESET, CBRUXENT command is issued, the previously defined virtual volumes in the DR cluster are moved from insert to scratch category. The volumes are assigned the category as defined in the DR host’s DEVSUPxx parmlib member for MEDIA2 scratch tapes

*Example 11-13 Moving volumes to scratch category*

---

```
RESPONSE=ZOS24
CBR1180I OAM tape volume status: 443
VOLUME MEDIA STORAGE LIBRARY USE W C SOFTWARE LIBRARY
TYPE GROUP NAME ATR P P ERR STAT CATEGORY
DR0000 MEDIA2 *SCRTCH* ZOSVTS1 S NOERROR SCRMED2
-----
RECORDING TECH: 36 TRACK COMPACTION: UNKNOWN
SPECIAL ATTRIBUTE: NONE ENTER/EJECT DATE: 2021-08-05
CREATION DATE: 2021-08-04 EXPIRATION DATE:
LAST MOUNTED DATE: LAST WRITTEN DATE:
SHELF LOCATION:
OWNER:
LM SG: LM SC: LM MC: LM DC:
LM CATEGORY: 1002
-----
Logical volume.
CBR3621I Enter request ignored by the cartridge entry installation
exit (CBRUXENT).
```

---

On any other system, if CBRUXENT exit is enabled and PRTITION is set up to deny DR\* volumes, which is normal, you see the following message:

```
CBR3620I Entry of volume DR0000 into library ZOSVTS1 failed.
```

In this situation, it is important to return the volumes to the scratch pool at the conclusion of the DR testing cycle so that their status in the TCDB and RMM CDS on the production system from which the copy is taken matches the status in the library.

If the library does not reflect scratch for the entire volser range, only part of that range can be mounted to create data sets. To clean up the volumes that were written to during the test on the DR test system, the RMM SEARCH VOLUME command can be used with the CLIST operand to set volumes to PENDING RELEASE. You do not need to run VRSEL because volumes in pending release are returned to scratch status by using a single job step that specifies only EXPROC function.

3. Example 11-14 shows the commands to find all the tapes that begin with DR\* that are listed in the RMM CDS in MASTER status and write them to a file by using DELETEVOLUME RELEASE commands to set them to PENDING RELEASE.

*Example 11-14 Find DR\* tapes and write them to a file*

---

```
//STEPA EXEC PGM=IKJEFT01
//SYSTSPRT DD DUMMY
//RMMCLIST DD DSN=h1q.RMMDV,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(20,20),RLSE),
// UNIT=SYSDA,LRECL=80,RECFM=FB
//SYSTSIN DD *
PROF NOMSGID
RMM SV VOLUME(DR*) OWNER(*) STATUS(MASTER) LIM(9999) -
CLIST('RMM DELETEVOLUME ',' RELEASE')
```

---

4. After the contents of the h1q.RMMDV file is confirmed as correct, you can run the commands shown in Example 11-15.

*Example 11-15 Set DR\* volumes to PENDING RELEASE*

---

```
//STEPB EXEC PGM=IKJEFT01,DYNAMNBR=60
//SYSTSPRT DD DSN=h1q.RMMDV.OUT,
// DISP=(NEW,CATLG,DELETE),
// SPACE=(CYL,(100,20),RLSE),
// UNIT=SYSDA,LRECL=80,RECFM=FB
//SYSTSIN DD DISP=SHR,
// DSN=h1q.RMMDV
```

---

Now the entire inventory of DR\* volumes should be in PENDING RELEASE or SCRATCH status. An EDGHSKP job can be run that moves only that range of volumes back into the scratch pool.

If for some reason this cleanup is not completed before the DR host is shut down, a list of volumes must be made from the library for volumes that are not in SCRATCH status. Also, the EDGSPLCS utility must be used to move them to the scratch pool.

If preferred, after the volume range is moved to SCRATCH, the command RMM DV *volser* FORCE EJECT can be used to delete the volume range from the RMM CDS, the OAM TCDB (assuming the EJECT default in ISMF for the library is set to PURGE), and the TS7700 library inventory at the same time.

However, in attempting to delete the volumes from the TS7700 library inventory, it is important to consider the influence of the Expire Hold Settings for the scratch category that is used for those volumes.

If the scratch category as defined in the TS7700 MI has the Expire Hold option selected, this category cannot be reused or have the back-end data deleted until the value that is specified in the Expire Hold Settings is satisfied.

Complete the following steps to examine this issue on the TS7700 MI:

1. Click **Virtual** → **Categories**.
2. Expand the scratch category by clicking the plus sign (+) on the left.
3. Highlight the category that is in use.
4. Select the Actions drop-down option **Modify Scratch Category** to open the window that is shown in Figure 11-1.

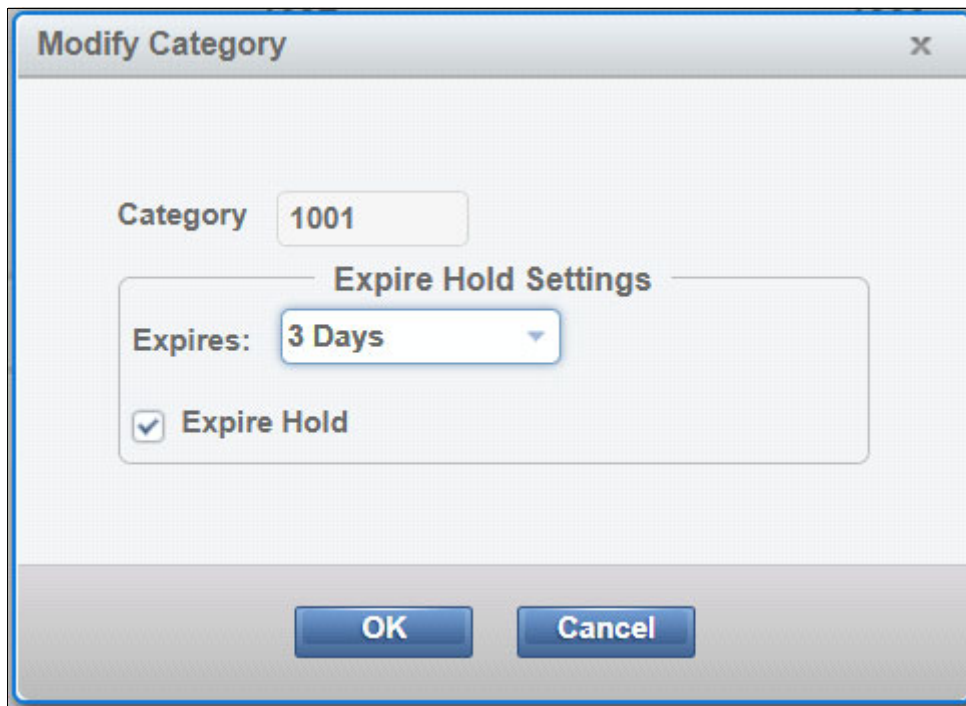


Figure 11-1 Expire Hold window

If the Expire Hold option is selected and you want to delete the volume inventory, then clear the checkbox and click **OK**. As a result, the grace period that is shown in the Expire Hold Settings field (see Figure 11-1) that was defined for that scratch category is not enforced.

It is recommended that this option is not selected for your DR or Test LPAR scratch category, which uses a unique DEVSUPxx parmlib member value, so as not to affect the production volumes expire delete settings.

This option is not selected by default when you create a scratch category. Also, it is not recommend to set it for Expire Hold in a DR or Test environment unless the data that is used during a test must be preserved.

If you attempt to eject a volume while it is in Expire Hold status, a failure message is issued with CBR3650I with CB3726I messages and error code 12. After the Expire Hold option

checkbox is cleared, because the RMM entry is deleted, reissue the EJECT command from ISMF to remove the volume from the library inventory.

Figure 11-2 shows a failure of the EJECT command and then success after the Expire Hold option is cleared.

```
CBR3650I Eject of volume B00470 from library ZOSVTS1 failed.
CBR3726I Function incompatible error code 12 from library ZOSVTS1 for volume B00470.
CBR3650I Eject of volume B00470 from library ZOSVTS1 failed.
CBR3726I Function incompatible error code 12 from library ZOSVTS1 for volume B00470.
SE 'CBR3650I Eject of volume B00470 from library ZOSVTS1 failed.',USER=(KARIM),LOGON
SE 'CBR3726I Function incompatible error code 12 from library ZOSVTS1 for volume B00470.',USER=(KARIM),LOGON
CBR3750I Message from library ZOSVTS1D: OP0409 The volume hold function was disabled for scratch category 1002.
CBR3010I Volume B00470 ejected from library ZOSVTS1. Place in shelf location ??????.
SE 'CBR3012I Volume B00470 ejected from library ZOSVTS1.',USER=(KARIM),LOGON
```

Figure 11-2 Eject Failure

## 11.6 Reading DR volumes back on PROD

If tape data is written during a DR test and you want to read this data back on the PROD LPAR, complete the following steps:

1. On the DR HOST, issue the TSO subcommand `RMM CV volser HOLD` to the volumes so that they do not go to SCRATCH with the rest of the DR volume range.
2. On PROD, use ISMF under `VOLUME>MOUNTABLE TAPE>` to list the volume. Then, use ALTER command to change the USE ATTRIBUTE of the volume from S to P and delete the storage group name of \*SCRATCH\*.

The volume then appears as MASTER in DFSMSrmm. The first file data set name might need to be changed or added unless BLP is being used with RMM DELETEDDATASET and ADDDATASET names.

After the data is moved from this tape, it can be deleted and redefined as a scratch tape in the RMM CDS with `RMM DV volser FORCE NOEJECT` and `IDCAMS DELETE VOLUMEENTRIES` commands.

## 11.7 Reading LWORM volumes during DR testing

LWORM volumes have two attributes that are used by the tape management system to ensure write integrity. The first is the World Wide ID (WWID) and the second is the Write Mount Count (WMC). During a DR test, it is possible that part of the test includes access to an LWORM volume that was previously written. If data was appended to the volume after a copy of the RMM CDS was taken, then the WMC would have been increased by a value of 1 or more from what is recorded in the RMM CDS. This will cause DFRMM to deny access to the data on the LWORM volume with error message EDG4056I.

The TSO subcommand `RMM CHANGEVOLUME volser WMC (nn)` can be used to clear this error condition and allow access to the volume. The value for *nn* can be an absolute value or an incremental value like +1.

This issue also pertains to restoring volumes to a specific time frame, such as when using IBM Cyberguard.

## 11.8 Limited Housekeeping function

With New function APAR [OA64283](#), support was added to limit EDGHSKP (housekeeping) functions using HSKP(LIMITED or ALL) in EDGRMMxx parmlib member. If LIMITED is set then inventory management functions CATSYNC,DSTORE,EXPROC and VRSEL are restricted as the CDS is not allowed to be updated during housekeeping. However, the following commands can be run:

- ▶ VRSEL(VERIFY)
- ▶ CATSYNCH(VERIFY)
- ▶ RPTEXT
- ▶ BACKUP

The default is HSKP(ALL) if the parameter is not specified in EDGRMMxx parmlib member.

Using HSKP(LIMITED) is useful during disaster recovery testing or when running a test in which you do not want to risk scratching production volumes. However, having limited housekeeping function engaged will prevent cleanup of any volumes written during the DR test and may not be suitable for all DR testing environments. Housekeeping can be unintentionally submitted by DFRMM even if you held your batch scheduling system in a situation when you are running low on scratches and EDGRMMxx SCRATCHPROC is specified or allowed to default. With HSKP(LIMITED) specified, housekeeping functions that are submitted from any source receive the following messages:


```
EDG6322I HOUSEKEEPING PARM SET TO LIMITED (HSKP=LIMITED), RESTRICTING->
CATSYNC,DSTORE,EXPROC,VRSEL
```

```
EDG6901I UTILITY EDGHSKP COMPLETED WITH RETURN CODE 4
```

You can verify the setting by issuing the command RMM LC ALL or RMM LC OPTION and viewing the value of the field Housekeeping.







# ePartitioning examples with **OPENRULE** and **PRTITION** parameters

This chapter includes examples of how the library partitioning commands in DFSMSrmm can be used.

This chapter includes the following topics:

- ▶ 12.1, “Separating a TS7700 library between two LPARs or SYSPLEXES” on page 66
- ▶ 12.2, “Customizing OPENRULE statements to allow cross LPAR read of volsers” on page 67

## 12.1 Separating a TS7700 library between two LPARs or SYSPLEXES

Partitioning a TS7700 library in DFSMSrmm is when you have two LPARs each with their own volume range that is used and managed independently.

In this example, the RMM control data set (CDS) and OAM TCDB structure are placed on shared DASD. You can also have a separate RMM CDS and OAM TCDB for each LPAR if the LPARS do not need to read each other's volumes. The intent is that both LPARs read and write data, manage the expiration of the data independently, have their own scratch pool, and perform return to scratch processing separately.

The first requirement for such an arrangement is that each LPAR has its own unique values that are coded in the DEVSUPxx parmlib member. In the TS7700, unique DEVSUPxx values on each LPAR and corresponding scratch categories establish separate scratch pools to be used by each system. If you use the same DEVSUPxx categories on both LPARs, then the scratch pool is shared.

The following statements assume that System A uses a range of volumes that start with A and that System B uses a range of volumes that start with B. Each LPAR cannot access the specific volume range of another LPAR. However, the LPARS do have an OPENRULE statement that allows for the read of foreign volumes that can be used to allow the other LPARs volumes.

System A statements that are to be added to EDGRMMxx parmlib on that system are shown in Example 12-1.

*Example 12-1 System A EDGRMMxx parmlib*

---

```
/* GLOBAL PARTITION RULE - IGNORE ALL VOLUMES WITH NO SPECIFIC RULE */
PRTITION VOLUME(*) TYPE(RMM) SMT(IGNORE) NOSMT(IGNORE)
PRTITION VOLUME(*) TYPE(NORMM) SMT(IGNORE) NOSMT(IGNORE)
/* ALLOW READ BUT NOT WRITE OF VOLUMES NOT DEFINED TO RMM ON THIS SYSTEM */
OPENRULE VOLUME(*) TYPE(NORMM) INPUT(ACCEPT) OUTPUT(REJECT)
/* THIS SYSPLEX OWNS A* SMSTAPE VOLUMES IF PREDEFINED */
PRTITION VOLUME(A*) TYPE(RMM) SMT(ACCEPT)
OPENRULE VOLUME(A*) TYPE(RMM) INPUT(ACCEPT) OUTPUT(ACCEPT)
```

---

The global partition rules (the first two PRTITION commands) lock down all volume ranges for cartridge entry and return-to-scratch processing (EXPROC). The first statement covers volumes that are defined to RMM, and the second statement covers volumes that are unknown to RMM. You can combine these statements with TYPE(ALL). The intent is to lock down all possible volume ranges by these statements and then open up specific volume ranges with additional PRTITION statements.

The advantage of having these global statements is that if a volume outside the intended ranges to be used on this system was defined to RMM in error, it would not have TCDB entries created during cartridge entry and would not be processed by RMM housekeeping and sent to the scratch pool in error.

The point of preventing the creation of the TCDB entries on this system outside of a specified range is that the system that creates the TCDB entries is the one that sets the category code when the volume is moved to the scratch pool. The category code determines which scratch pool is presented by the library during mount processing.

The OPENRULE statement that comes next allows for a foreign volume to be read, but not written. The Tape Management System marks as rejected any volume that is not defined to RMM on which a write attempt is made. A foreign volume that is read on this system is not added to the RMM CDS and is not processed during return to scratch.

The third set of statements is the opening up of the specific volume range that is predefined to RMM and allowed full read/write and return to scratch processing. With these statements in effect, any volume that begins with A that is inserted into the library (and already has an entry in the RMM CDS), creates a TCDB entry and is moved to the scratch pool for use on System A. If you do want to predefine the volume entries to RMM, you can specify TYPE(ALL) instead of TYPE(RMM).

During the EDGHSKP function EXPROC on this LPAR, only volumes that start with A that are defined to RMM can be returned to SCRATCH status. All other volume ranges are prevented from processing return to scratch requests by the global partition rule.

System B statements to be added to EDGRMMxx parmlib on that system are shown in Example 12-2.

*Example 12-2 System B EDGRMMxx parmlib*

---

```
/* GLOBAL PARTITION RULE - IGNORE ALL VOLUMES WITH NO SPECIFIC RULE */
PRITITION VOLUME(*) TYPE(RMM) SMT(IGNORE) NOSMT(IGNORE)
PRITITION VOLUME(*) TYPE(NORMM) SMT(IGNORE) NOSMT(IGNORE)
/* ALLOW READ BUT NOT WRITE OF VOLUMES NOT DEFINED TO RMM ON THIS SYSTEM */
OPENRULE VOLUME(*) TYPE(NORMM) INPUT(ACCEPT) OUTPUT(REJECT)
/* THIS SYSPLEX OWNS B* SMSTAPE VOLUMES IF PREDEFINED */
PRITITION VOLUME(B*) TYPE(RMM) SMT(ACCEPT)
OPENRULE VOLUME(B*) TYPE(RMM) INPUT(ACCEPT) OUTPUT(ACCEPT)
```

---

The statements that used on System B are the same, except for the volume range in the final PRITITION and OPENRULE statements. These statements limit System B to creating TCDB entries for volsers in the B range, return volsers to scratch in the B range, and read and write to the B range, but do allow foreign tapes to be read.

## 12.2 Customizing OPENRULE statements to allow cross LPAR read of volsers

To add to the previous example, this scenario defines System A as a production system and System B as a test or development system. In the scenario, System B can read the volser range in use by System B, but cannot overwrite or return to scratch the volumes owned by System A.

In this scenario, it is advantageous to have a common RMM CDS and OAM TCDB that are on shared DASD because data is shared between two systems rather than segregating them completely.

System A EDGRMMxx parmlib statements are unaffected by this change of intent and remain the same as in the first example, as shown in Example 12-3.

*Example 12-3 System A EDGRMMxx parmlib*

---

```
/* GLOBAL PARTITION RULE - IGNORE ALL VOLUMES WITH NO SPECIFIC RULE */
PRITION VOLUME(*) TYPE(RMM) SMT(IGNORE) NOSMT(IGNORE)
PRITION VOLUME(*) TYPE(NORMM) SMT(IGNORE) NOSMT(IGNORE)
/* ALLOW READ BUT NOT WRITE OF VOLUMES NOT DEFINED TO RMM ON THIS SYSTEM */
OPENRULE VOLUME(*) TYPE(NORMM) INPUT(ACCEPT) OUTPUT(REJECT)
/* THIS SYSPLEX OWNS A* SMSTAPE VOLUMES IF PREDEFINED */
PRITION VOLUME(A*) TYPE(RMM) SMT(ACCEPT)
OPENRULE VOLUME(A*) TYPE(RMM) INPUT(ACCEPT) OUTPUT(ACCEPT)
```

---

Again, these statements, limit this LPAR to adding A volumes to the OAM TCDB, returning to scratch only A volumes, and unfettered read/write access to only A volumes.

The changes affect the System B EDGRMMxx parmlib, as shown in Example 12-4.

*Example 12-4 System B EDGRMMxx parmlib*

---

```
/* GLOBAL PARTITION RULE - IGNORE ALL VOLUMES WITH NO SPECIFIC RULE */
PRITION VOLUME(*) TYPE(RMM) SMT(IGNORE) NOSMT(IGNORE)
PRITION VOLUME(*) TYPE(NORMM) SMT(IGNORE) NOSMT(IGNORE)
/* ALLOW READ BUT NOT WRITE OF VOLUMES NOT DEFINED TO RMM ON THIS SYSTEM */
OPENRULE VOLUME(*) TYPE(NORMM) INPUT(ACCEPT) OUTPUT(REJECT)
/* THIS SYSPLEX OWNS B* SMSTAPE VOLUMES IF PREDEFINED */
PRITION VOLUME(B*) TYPE(RMM) SMT(ACCEPT)
OPENRULE VOLUME(B*) TYPE(RMM) INPUT(ACCEPT) OUTPUT(ACCEPT)
/* ALLOW READ OF SYSTEM A VOLUMES BUT NOT WRITE */
OPENRULE VOLUME(A*) TYPE(RMM) INPUT(ACCEPT) OUPUT(REJECT) */
```

---

Notice that an OPENRULE statement is added as the last entry in the System B EDGRMMxx parmlib. Because a shared RMM CDS and shared OAM TCDB are used, this system can access the volume entries in those two sources for volumes that begin with A and volumes that begin with B.

With this configuration, volumes that start with A can be mounted and read on System B. Writes are still prevented because of OUTPUT(REJECT). Also, any run of EDGHSKP EXPROC on this system ignores volumes that are outside the A range because of the PRITION VOLUME(\*) TYPE(RMM) SMT(IGNORE) NOSMT(IGNORE) statement.

System B can now safely access data that is written on System A without any concern about overwriting that data or returning it to scratch. However, if the two systems show a common ICF catalog structure, then System B can delete the catalog entry or update LASTREF values by accessing the data. To prevent System B from making changes, the ICF catalogs other than the TCDB must not be shared between systems.



## **Expire Hold setting and available scratch count**

This chapter includes the discussion of the TS7700 EXPIRE HOLD setting and its effect on available scratch counts in the library.

## TS7700 EXPIRE HOLD overview

The TS7700 family of VTS products uses categories that are coded in the DEVSUPxx parmlib member. In the TS7700 MI, these categories are used to define the characteristics of the scratch pool that is used by each LPAR.

Each scratch category includes an EXPIRE TIME and an option that can be selected that is called EXPIRE HOLD. The EXPIRE HOLD option is used to ensure that a virtual volume that in the scratch pool cannot be reused or deleted before the specified grace period as the EXPIRE time passes.

After DFSMSrmm moves a volume into the scratch pool, the volume might appear as SCRATCH to the host. However, if the EXPIRE HOLD option is selected, the volume cannot be used as a SCRATCH volume by the TS7700. Therefore, it is often confusing as to how many SCRATCH volumes are available for use on the TS7700.

Some discrepancy often exists between the values that are reported for scratch counts from various sources (TS7700 MI, TCDB, and TMS). To obtain the number of usable SCRATCH volumes in the TS7700, issue the `D SMS,LIBRARY(libname),DETAIL` command.

In addition to providing other information about the designated tape library, this command reports the number of usable SCRATCH volumes. This value is obtained directly from the TS7700 and surfaced within the CBR1110I message.

When EXPIRE HOLD is in effect, the total number of SCRATCH volumes in the TCDB for that composite library can differ from the total number of usable SCRATCH volumes. The numbers can differ because volumes for which the EXPIRE time is not yet elapsed are not eligible to be selected to satisfy a SCRATCH mount request. For this reason, the most accurate source of scratch counts for a TS7700 always is the output from the `D SMS,LIBRARY(libname),DETAIL` command.

As an example, see the scratch counts for MEDIA and MEDIA2 in Figure 13-1 on page 71.

```

CBR110I OAM library status: 155
TAPE      LIB  DEVICE  TOT  ONL  AVL  TOTAL  EMPTY  SCRATCH  ON  OP
LIBRARY   TYP  TYPE    DRV  DRV  DRV  SLOTS  SLOTS  VOLS    Y  Y
ZOSVTS1  VCL  3957-V07 144  96  96    0      0     3679   Y  Y

-----

MEDIA          SCRATCH          SCRATCH          SCRATCH
TYPE           COUNT            THRESHOLD        CATEGORY
MEDIA1         51                25               1001
MEDIA2        3628             500              1002

-----

DISTRIBUTED LIBRARIES: ZOSVTS1D

-----

LIBRARY ID: 11111
OPERATIONAL STATE: AUTOMATED
ERROR CATEGORY SCRATCH COUNT:          6
CORRUPTED TOKEN VOLUME COUNT:         0

-----

Operation degraded due to unavailable hardware resource.
VTS operations degraded.
Library supports import/export.
Library supports outboard policy management.

```

Figure 13-1 SMS Lib Details Output

These values are the usable scratch counts that are contained in the categories that are used by the host LPAR that issued the D SMS,LIB(*libname*),DETAIL command. The scratch category values are to the far right of the display. If the TS7700 encounters a low on scratch or low on cache condition, you can modify the EXPIRE times or clear the Expire Hold option to help alleviate the condition. See Figure 13-2.

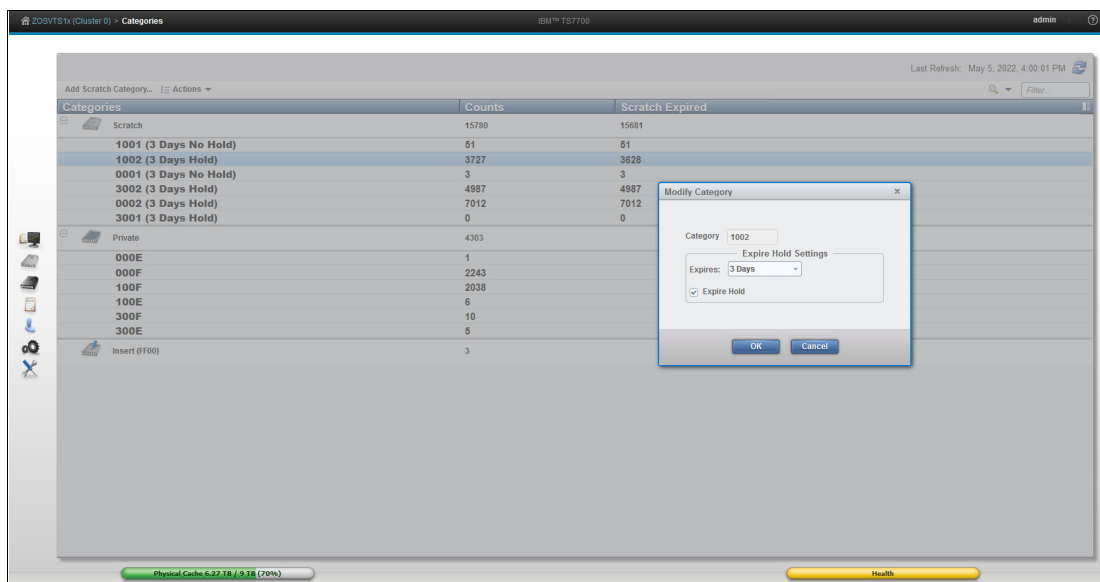


Figure 13-2 Expire Hold on TS7700 MI

Changes that are made to the Expire Time and the EXPIRE HOLD box are retroactively applied to the volumes in that scratch category. Changing these values allows excluded

volumes to be considered in the scratch counts, which you can verify with another issuance of the `D SMS,LIB(libname),DETAIL` command.





# Common problems in DFSMSrmm

This chapter discusses some common scenarios in DFSMSrmm and their solutions. It includes the following topics:

- ▶ 14.1, “Reasons why a volume does not go to SCRATCH” on page 74
- ▶ 14.2, “Long-running EDGHSKP jobs” on page 75

## 14.1 Reasons why a volume does not go to SCRATCH

By default, DFSMSrmm is intended to protect data from early expiration. The following list provides examples that might prevent a volume from returning to SCRATCH:

- ▶ CATSYSID is specified other than \* and the system to which this volser was assigned is not in the list.

If the system that moved the tape from SCRATCH to PRIVATE does not share catalogs with the system that is running the EXPROC function of DFSMSrmm, it cannot move the volume to SCRATCH. One resolution is to issue the RMM LV *volser* ALL command then check the system name to the far right of the ASSIGN field and run EXPROC on that system.

- ▶ The volume is not in its HOME location.

Again, an RMM LV *volser* ALL command can be issued to view information about where the volume is and what its HOME location is as shown at the bottom of the display.

To be moved to SCRATCH status, the value of CURRENT must be the same as HOME. You might need to issue the RMM CV *volser* HOME(CURRENT) command and then the RMM CV *volser* CONFIRMMOVE command to sync these two fields.

- ▶ The RETENTION DATE or EXPIRATION DATE is not yet passed.

Volumes that are under VRSEL retention have a calculated RETENTION DATE and an EXPIRATION DATE. Until both of these dates are passed, a VRSEL-retained volume cannot be moved to the SCRATCH pool.

If you want to move a VRSEL retain volume to SCRATCH manually, run the RMM DV *volser* RELEASE command. Although volumes under EXPDT retention do not have a RETENTION value, they might include a WHILECATALOG condition that must be satisfied, even if the EXPIRATION DATE is in the past.

In the RMM LV *volser* ALL output, check the EXPIRATION DATE field. If a notation exists that indicates (KEPTBYCATLG), one or more of the data sets on the volume must have their catalog entry deleted for this volume to be expired.

ISPF 3.4 and U for uncatalog can delete the catalog entries for the data sets and allow the volume to return to SCRATCH.

For more information, see [DFSMSrmm: Things to Check When RMM Fails to Return a Virtual Tape Volume to SCRATCH](#).

## 14.2 Long-running EDGHSKP jobs

The time to complete the inventory management process in DFSMSrmm is a function of the parameters that are passed in the EDGHSKP steps, and the number of steps that are required to complete the job.

The functions of EDGHSKP that normally take the longest to complete are VRSEL, EXPROC, and CATSYNCH. It can be beneficial to separate the different steps of EDGHSKP into their own steps for comparison of which steps are taking the longest. Consider the aspects of each of the three functions.

### 14.2.1 VRSEL

VRSEL unloads every record in the RMM control data set (CDS) and then uses a sort product to match all these records to the VRS rules that were defined. Both an increase in CDS entries and an increase in the number of rules that are used for comparisons can increase the workload on the sort product.

A sort product can work more efficiently if the data set masks in the VRS rules are more specific. Fewer rules can also reduce the workload. The newer retention method EXPDT does not need to match the VRS because EXPDT works dynamically to record changes to the RMM CDS. Any volumes that can be moved under the EXPDT retention method reduces the workload that is done during VRSEL.

Another aspect of the work done during VRSEL that can affect performance is the need to check the catalog status of data sets that are defined to the RMM CDS. A one-time run of function CATSYNCH to set the initial catalog status of the RMM-managed data sets usually suffices and thereafter, the VRSEL step does not need to issue catalog locates.

A change to VRS rules can add a large amount of work for one run of VRSEL. When VRS rules are changed, allow for at least one long-running EDGHSKP cycle as the new or changed VRS is applied to all of the eligible, managed data sets and volumes.

### 14.2.2 EXPROC

EXPROC is the mechanism that returns volume to SCRATCH in DFSMSrmm. EXPROC reviews the retention dates, availability flags, and expiration dates of volumes to determine whether they are suitable. EXPROC then issues commands to change volume eligibility for expiration from MASTER or USER to SCRATCH.

If many volumes must be changed from MASTER or USER to SCRATCH, the time to process all of them and return control to DFSMSrmm is lengthened. In a situation where a tape migration or tape environment cleanup is occurring, it is normal to expect a longer running EXPROC step during EDGHSKP processing.

### 14.2.3 CATSYNCH

CATSYNCH is a parameter that can be passed to EDGHSKP. This parameter synchronizes the data set records in the RMM CDS to the ICF Catalog entries in your user catalogs.

Because it must review every catalog in the catalog structure, CATSYNCH might take a long time to complete. After the RMM CDS is synchronized, the Catalog Address Space calls

RMM and updates the catalog status in the RMM CDS if the catalog status changes for any tape data set.

The intent is that CATSYNCH is run once or occasionally when needed and not in each EDGHSKP cycle. Some conditions must be considered to implement CATSYNCH. For more information, see [Implementing RMM Catalog Synchronization \(CATSYNCH\)](#).



# DFSORT JCL for use with DFSMSrmm

This chapter describes how to use the DFSMSrmm extract data set with the DFSORT product in several real-world use cases. It includes the following topics:

- ▶ 15.1, “Introduction” on page 78
- ▶ 15.2, “Overview of DFSMSrmm data” on page 79
- ▶ 15.3, “Example use cases” on page 81

## 15.1 Introduction

DFSMSrmm provides various options that can be used to report on managed tape data. Commonly used options include, but are not limited to, the following examples:

- ▶ DFSMSrmm report generator
- ▶ DFSMSrmm supplied EXECs
- ▶ DFSMSrmm inventory management reports
- ▶ DFSMSrmm supplied sample reports

These options can be used for reporting and analyzing tape data. For more information, see [DFSMSrmm Reporting manual](#).

These built-in capabilities can be augmented by knowing how to use DFSORT, which is IBM's high-performance sort, merge, copy, analysis, and reporting product. DFSORT, along with its multipurpose ICETOOL utility, can create reports from data, such as the DFSMSrmm extract data set, Activity File, and the System Management Facility (SMF) records.

For more information, see [z/OS 2.5: DFSORT Application Programming Guide](#).

**Note:** To view the JCL examples in this chapter, see [JCL examples for Chapter 15: DFSORT JCL for use with DFSMSrmm](#). You can use the text file format to copy, paste, and modify the JCL if you want to run the samples.

DFSORT can help resolve unexpected problems. Typically, RMM TSO subcommands, such as SEARCHVOLUME and SEARCHDATASET, feature a CLIST operand that can use the search results to generate a list of subcommands, such as ADDVOLUME, ADDDATASET, CHANGEVOLUME, and CHANGEDATASET, to help recovery or to correct discrepancies. However, some information is recorded in the RMM CDS that cannot be searched by using these subcommands.

Also, because the SEARCHVOLUME and SEARCHDATASET subcommands query the RMM CDS, these subcommands might not work in specific recovery situations, such as rebuilding a lost volume or data set records from a backup extract data set. In these situations, DFSORT can be used to generate a list of subcommands for corrective action instead.

## 15.2 Overview of DFSMSrmm data

Normal DFSMSrmm activity and inventory management generate data that can be used for various purposes. The DFSMSrmm Reporting manual documents the mapping macros that are available to be used as programming interfaces for users. These mappings detail the offset, length, and data type of each field within the Activity file, Extract data set records, and SMF records that are specific to DFSMSrmm.

You can use these mapping macros for reference when coding DFSORT or ICETOOL jobs. However, it can be more advantageous to use the DFSMSrmm provided symbols to create reports for DFSMSrmm managed data and resources.

These symbol mappings are available in SYS1.MACLIB after SMP/E APPLY processing as the following members:

EDGACTSY	Activity file symbols
EDGACXSY	Combined activity/extended Extract record symbol mapping
EDGEXTSY	Extract data set symbols
EDGSMFSY	SMF record symbols
EDGS42SY	SMF audit record type 42 subtype 22
EDGSRCSY	SMF record

By pointing the SYMNames DD statement directly to any of these members in DFSORT or ICETOOL jobs, you can use these symbols in place of offset, length, and data type in control statements. These members also include symbols for constants.

Alternatively, these members can be copied somewhere else and customized if suitable. For example, you might want to modify members with more constant symbols. After you finish the customization, you must point only the SYMNames DD to the member or data set to where the modified symbols mapping is found.

Example 15-1 shows how to display the symbols in the SYS1.MACLIB members as output to the job log (note that the SYMNOUT DD can be changed to point to an output data set or member).

*Example 15-1 DFSORT SYMBOLS JCL*

---

```
//SYMBOLS EXEC PGM=SORT
//SYSOUT DD SYSOUT=*
//SYMNOUT DD SYSOUT=*
//SYMNames DD DISP=SHR,DSN=SYS1.MACLIB(EDGS42SY)
// DD DISP=SHR,DSN=SYS1.MACLIB(EDGSRCSY)
//SORTIN DD *
//SORTOUT DD SYSOUT=*
//SYSIN DD *
        OPTION COPY
/*
```

---

An example SYMBOLS Job Output (from SYMNOUT DD) is shown in Figure 15-1.

```

----- ORIGINAL STATEMENTS FROM SYMNAME -----
*****
*
*   RMM Inventory Management SMF Audit Record type 42 subtype 22
*   DFSORT Symbol mapping
*
*****
* z/OS DFSMSrmm V1R10
*
*PROPRIETARY V3 STATEMENT
*LICENSED MATERIALS - PROPERTY OF IBM
*"RESTRICTED MATERIALS OF IBM"
*5694-A01
*COPYRIGHT 1993 2008 IBM CORP.
*STATUS = HDZ1A10
*END PROPRIETARY V3 STATEMENT
*
*****
* SEE "z/OS DFSMSrmm Reporting" FOR FIELD DETAILS ON RMM RECORDS
* SEE "DFSORT APG" FOR DETAILS OF USING SYMBOLS.
*****
* CHANGE ACTIVITY:
* $MZ=V1R10 ,1RA,070608,MB : SMF Forward Recovery @MZA
* $K0=K1A1214,1RA,070809,WS : hex representation of subtype @K0A
*****
*
*****
* Header for SMF record type 42
*****
SMF42,1,8463
SMF42RCL,=,2,BI      Record Length
SMF42SGD,*,2,BI     Segment Descriptor (RDW)
SMF42FLG,*,1,BI     System indicator flags
SMF42FSI,X'80'      When set=subsystem id follows system id
SMF42FSU,X'40'      When set = subtypes are used
SMF42FXA,X'04'      When set = MVS/XA (SMF enters)
SMF42FS2,X'02'      When set = VS2 (SMF enters)
SMF42FS1,X'01'      When set = VS1 (SMF enters)

```

Figure 15-1 DFSORT SYMBOLS



## 15.3 Example use cases

This section describes use cases where DFSORT can be used with DFSMSrmm data to create useful reports or perform other needed functions.

### 15.3.1 Use case 1: Reporting tape volume usage

This use case describes how to create a tape report for total tape volume usage in gigabytes.

In this example, you generate a report to estimate the total space that is used in your tape environment to assist with calculating client billing. This estimation includes the total space for each non-scratch tape, and a summation of all the total space that is used across all non-scratch tapes that are managed by DFSMSrmm.

Other tape volume information is requested within the report for more insights and includes the following information:

- ▶ First file on volume
- ▶ Jobname
- ▶ Assign or create date
- ▶ Volume expiration date
- ▶ Original expiration date
- ▶ Volume status
- ▶ Volume retention date

Only nonextended extract data set records are available for input to the reporting job.

Because of the requirements, the RVTUSE field is chosen from the volume record in the extract data set to estimate the total space that is used on tapes. Therefore, the report is relying on the volume usage (KB) field to estimate the total space use on tapes, which is the total amount of uncompressed data on the volume.

**Note:** RVTUSE can exceed the storage capacity of the physical tape if the data is stored in a library that uses a compressed format.

The sample JCL required in this use case is shown in Example 15-2 on page 82.

*Example 15-2 Sample JCL for Tape Volume Usage Report*

```

/*****
/*      Specify input nonextended extract data set name here
/*****
//      SET EXTRACT='your.non-extended-extract.dsn'
/*****
/*      Specify SPACE parms for temp data set large enough to
/*      hold records for all nonscratch tapes in report
/*****
//      SET TEMPPRI=x
//      SET TEMPSEC=y
//      SET TEMPSIZE='TRK | CYL'
/*****
//USECASE1 EXEC PGM=ICETOOL
//SYMNAMES DD DISP=SHR,DSN=SYS1.MACLIB(EDGEXTSY)
//          DD *
**** Reformatted type 'V' records
FMT-RDW,01,04,BI
FMT-RVVOLSER,*,06,CH
FMT-RVDSNAM1,*,44,CH
FMT-RVCRJOB,*,08,CH
FMT-RVASDATE,*,10,CH
FMT-RVEXPDT,*,10,CH
FMT-RVEXPDTO,*,10,CH
FMT-RVSTATUS,*,08,CH
FMT-RVRETDAT,*,10,CH
FMT-RVTUSE,*,16,FS
//TOOLMSG DD SYSOUT=* ICETOOL MESSAGES
//DFSMSG DD SYSOUT=* DFSORT MESSAGES
//SYMNOUT DD SYSOUT=* DFSORT Symbols with offsets
//EXTRACT DD DISP=SHR,DSN=&EXTRACT
//GETVOLS DD DSN=&&GV,DISP=(,PASS),
//          SPACE=(&TEMPSIZE,(&TEMPPRI,&TEMPSEC),RLSE)
//VOLRPT DD SYSOUT=*
//TOOLIN DD *      CONTROL STATEMENTS
COPY FROM(EXTRACT) TO(GETVOLS) USING(VOLS)
DISPLAY FROM(GETVOLS) LIST(VOLRPT)
TITLE('DFSMSrmm - Tape Usage Example Report')
DATE TIME PAGE
BLANK
HEADER('VOLSER') ON(FMT-RVVOLSER)
HEADER('DSNAME') ON(FMT-RVDSNAM1)
HEADER('JOBNAME') ON(FMT-RVCRJOB)
HEADER('AS/CR DATE') ON(FMT-RVASDATE)
HEADER('EXPDT') ON(FMT-RVEXPDT)
HEADER('JCL EXPDT') ON(FMT-RVEXPDTO)
HEADER('STATUS') ON(FMT-RVSTATUS)
HEADER('RET DATE') ON(FMT-RVRETDAT)
HEADER('USAGE') ON(FMT-RVTUSE,/MB,E'999999999.99999')
TOTAL('TOTAL TAPE USAGE IN GB: ')
/*
//VOLSCNTL DD *
OPTION VLSCMP,NOCHALT,LOCALE=NONE
INCLUDE COND=(EXTRACT_TYPID,EQ,RVTYPEID,AND, # TYPE 'V'
RVSTATUS,NE,C'SCRATCH') # Not scratch stat
OUTREC BUILD=(EXTRACT_RDW, # FMT-RDW
RVVOLSER, # FMT-RVVOLSER
RVDSNAM1, # FMT-RVDSNAM1
RVCRJOB, # FMT-RVCRJOB
RVASDATE, # FMT-RVASDATE
RVEXPDT, # FMT-RVEXPDT
RVEXPDTO, # FMT-RVEXPDTO
RVSTATUS, # FMT-RVSTATUS
RVRETDAT, # FMT-RVRETDAT
RVTUSE, # FMT-RVTUSE
MUL,+1000000,FS,LENGTH=16) # +6 DECIMAL PLACES
/*

```

A sample of the output from this JCL is shown in Figure 15-2.

```

DFSMsrm - Tape Usage Example Report          07/26/22          18:00:21          - 27 -
VOLSER  DSNAME          JOBNAME      AS/CR DATE      EXPDPT          JCL EXPDT      STATUS      RET DATE      USAGE
-----  -
V20006  KARIM.OFFSITE.GDG.G0001V00  KARIMOP      04/15/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20007  KARIM.OFFSITE.GDG.TWO.G0001V00  KARIMOP      04/15/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20008  KARIM.OFFSITE.GDG.NONBDG      KARIMOP      04/15/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20009  KARIM.OFFSITE.GDG.G0002V00  KARIMOP      04/15/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20010  KARIM.OFFSITE.GDG.G0003V00  KARIMOP      04/15/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20011  KARIM.TEST.D990002          KARIMOP      12/03/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20012  XXX.TEST                    KARIMOP      10/29/2015      12/31/1999      12/31/1999      MASTER      WHILECATLG      0000000000.000003
V20016  ABARSOUT.D.C01V0011        KARIMOP      12/14/2012      12/31/1999      12/31/1999      MASTER      WHILECATLG      0000000000.000396
V20017  ABARSOUT.D.C01V0011        KARIMOP      12/14/2012      12/31/1999      12/31/1999      MASTER      WHILECATLG      0000000000.000396
V20018  ABARSOUT.D.C01V0011        KARIMOP      12/14/2012      12/31/1999      12/31/1999      MASTER      WHILECATLG      0000000000.001037
V20019  ABARSOUT.D.C01V0011        KARIMOP      12/14/2012      12/31/1999      12/31/1999      MASTER      WHILECATLG      0000000000.001037
V20020  ABARSOUT.D.C01V0011        KARIMOP      12/14/2012      12/31/1999      12/31/1999      MASTER      WHILECATLG      0000000000.001037
V20021  ABARSOUT.D.C01V0011        KARIMOP      12/14/2012      12/31/1999      12/31/1999      MASTER      WHILECATLG      0000000000.001037
V20026  KARIM.TESTCART.DAT02      KARIMOP      11/23/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20027  KARIM.TESTCART.DAT03      KARIMOP      11/23/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20028  KARIM.TESTCART.DAT04      KARIMOP      11/23/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20029  KARIM.TESTCART.DAT05      KARIMOP      11/23/2020      00/00/1999      00/00/1999      MASTER      WHILECATLG      0000000000.000003
V20041  KARIM.NEXTVRS.TEST2      KARIMOP      11/22/2019      12/31/2019      12/31/2019      MASTER      WHILECATLG      0000000000.000003
V20057  KARIM.TEST.VRS.OFFSITE.TEST.TEST  KARIMOP      11/13/2014      12/31/1999      12/31/1999      MASTER      WHILECATLG      11/19/2014      0000000000.000003
V20061  OAM.PRIMARY.DATA          OAM          10/27/2014      12/31/1999      12/31/1999      MASTER      CYCL/99999      0000000000.000305
V20064  OAM.PRIMARY.DATA          OAM          10/27/2014      12/31/1999      12/31/1999      MASTER      CYCL/99999      0000000000.000003
V20065  OAM.BACKUP.DATA          OAM          10/27/2014      12/31/1999      12/31/1999      MASTER      CYCL/99999      0000000000.000610

TOTAL TAPE USAGE IN GB:          0000002192.784655

```

Figure 15-2 Sample Tape Volume Usage Report

This example includes the following notable DD statements:

- ▶ SYMNAMES points to SYS1.MACLIB(EDGEXTSY), which is the symbols mapping for the extract data set, or SYMNAMES points to inline DFSORT symbols where are used \* for position, which maps consecutive fields in your records without having to compute their positions.
- ▶ TOOLIN has all the main, inline control statements.
- ▶ Extract points to the input data, which is the nonextended extract data set records.
- ▶ GETVOLS points to a temporary data set that is used for copying volume information from the input extract data set.
- ▶ VOLSCNTL contains the inline control statements that are used when you are copying data from the extract data set to the GETVOLS temporary data set.
- ▶ VOLRPT points to the finalized report output.

The main control statements in the TOOLIN DD start the primary actions of the ICETOOL job. Stepping through each of the main control statements explains how this job operates.

The following main control statements are used:

- ▶ COPY FROM(Extract) TO(GETVOLS) USING(VOLS)

This statement copies data from Extract to GETVOLS by using the control statements in VOLSCNTL DD. Except for the main TOOLIN DD, each DD with control statements has a fixed suffix of CNTL. When using these control statement DDs, the CNTL suffix must be used.

VOLSCNTL performs the following functions:

- INCLUDEs only volume records, that is, records that start with V, and RVSTATUS (the volume status symbol from EDGEXTSY) that are not equal (NE) to SCRATCH.
- Reformats the output records with the wanted information in the OUTREC FIELDS statement. The symbols that are used are the inline symbols that are coded to map the reformatted records.
- The RVTUSE field is multiplied by 1 million to add six decimal places to increase the precision when converting from KB to GB at a later point within the job by using /MB.

► **DISPLAY FROM(GETVOLS) LIST(DRAFT)**

This statement displays the reformatted volume records from GETVOLS to DRAFT and performs the following functions:

- Creates a TITLE for the report.
- Prints a DATE and TIME on each PAGE of the report.
- Creates a HEADER for each column or field within the report, and again inline symbols for reformatted volumes are used.

The reformatted RVTUSE field is converted from KB to GB by using /MB, which multiplies the numeric field by 1024\*1024. The field is formatted to include the six extra decimal point places to account for the higher precision that is necessary to avoid rounding down the tapes with low KB usage.

### **15.3.2 Use Case 2: Rebuilding data set records for recovered volume**

This use case describes how to use JCL to rebuild an RMM CDS data set record for a volume that was recovered by using the information from a previously obtained RMM Extract.

In this scenario, tape volume B01138 is deleted from the RMM CDS and redefined. The data is still available on the tape after the volume is redefined.

However, part of the recovery process is to rebuild the data set records in the RMM CDS. You have an extract data set with extended data set records from your last run of RMM inventory management before the event that caused the loss or deletion of the volumes information from out of RMM.

The provided JCL sample shows how DFSORT can find and parse the data set information for the recovered volume from the input extended data set Extract records.

Although this simple case involves only one recovered volume, this JCL can be customized to rebuild the data set records for an entire range of volumes.

For example, instead of specifying only B01138 in the RXVOLSER,EQ,C'B01138' part of the INCLUDE statement, you can more broadly specify 9,1,CH,EQ,C'B' to rebuild the data set records for all the volumes in the B\* volser range.

The sample JCL required in this use case is shown in Example 15-3 on page 85.

*Example 15-3 Sample JCL to rebuild a data set record*

---

```

/*****
/*      Specify input extended extract data set and output
/*      data set that contains generated RMM ADDDATASET commands
/*****
//      SET EXTRACT='your.extended-extract.dsn'
//      SET OUTPUT='your.output.rmmad.clist'
/*****
/*      Specify space parms for output data set with generated
/*      RMM ADDDATASET CLIST
/*****
//      SET OUTPRI=x
//      SET OUTSEC=y
//      SET OUTSIZE='TRK | CYL'
/*****
//USECASE2 EXEC PGM=SORT
//SORTIN  DD  DISP=SHR,DSN=&EXTRACT
//SORTOUT DD  DSN=&OUTPUT,
//          DISP=(,CATLG,DELETE),
//          UNIT=SYSDA,
//          SPACE=(&OUTSIZE,(&OUTPRI,&OUTSEC),RLSE)
//SYSOUT  DD  SYSOUT=*
//SYMNAMES DD DISP=SHR,DSN=SYS1.MACLIB(EDGEXTSY) SYMBOLS FROM EXTRACT
//SYMNOUT DD  SYSOUT=* DFSORT SYMBOLS WITH OFFSETS
//SYSIN   DD  *
      INCLUDE COND=(EXTRACT_TYPID,EQ,RXTYPEID,AND,           # Data records
                    RXVOLSER,EQ,C'B01138')                  # volser is B01138
      SORT FIELDS=(RXDSNAME,A)                               # Dataset name
      OUTFIL VTOF,                                          # Fixed len output
      BUILD=(C' RMM AD ''',RXDSNAME,                        # RMM AD command
            C''',C' -',/,
            C' VOLUME(',RXVOLSER,
            C') FILESEQ(',RXDDSSEQ,
            C')',80:X)
/*

```

---

A sample of the output of this JCL written to the SORTOUT DD is shown in Figure 15-3.

```

***** Top of Data *****
RMM AD 'RBOUCHA.TEST.LWORM.TEST1
VOLUME(B01138) FILESEQ( 1)
***** Bottom of Data *****

```

*Figure 15-3 Command to rebuild a data set record in RMM CDS*

The example includes the following notable DD statements:

- ▶ SORTIN points to the input data, which is the extended extract data set records.
- ▶ SORTOUT point to the output data set.
- ▶ SYMNAMES points to SYS1.MACLIB(EDGEXTSY), which is the symbols mapping for the extract data set.
- ▶ SYSIN includes the main, inline control statements.

The main control statements in this DFSORT job are in the SYSIN DD and are performing the following functions:

- ▶ INCLUDE records with RXTYPEID (extended extract data set records that start with X, and contain volume and data set information for tapes) AND RXVOLSER (that is, the volser field from the EDGEXTSY symbols mapping) equal (EQ) to B01138.
- ▶ SORT FIELDS invokes a sort of the RXDSNAME field (that is, the data set name field from the EDGEXTSY symbols mapping) in ascending (A) order.
- ▶ Reformat the output records into RMM TSO ADDDATASET subcommands, in the following form:

```
RMM AD data_set_name VOLUME(volser) FILESEQ(data set sequence number)
```

In this case, it produces only subcommands for the recovered data sets that are in the B00001 volser.

### 15.3.3 Use Case 3: Data sets are assigned an incorrect management value and that value must be removed

This use case describes a method for removing a VRS management value for a range of data sets.

For this case, a range of tape data sets were assigned an incorrect VRS because of a typographical error.

The management value (VRSMV) assigned to these tapes does not include a corresponding VRS and thus are getting a short, default retention period instead of the retention period that corresponds to the correct management value.

You correct the typographical error and want to retroactively remove the management value from the tape data sets that were affected.

You have an Extract with extended data set records from the most recent EDGHSKP job. The JCL in Example 15-4 on page 87 shows how to parse all the data sets that were assigned the incorrect VRSMV. The JCL builds RMM TSO subcommands to remove the management value and to push out the retention period for a year while you determine the correct retention policy to specify for future tape data set allocations.

The sample JCL that is required in this use case is shown in Example 15-4.

*Example 15-4 JCL to remove management value from data set records*

---

```

//*****
//*      Specify input extended extract data set and output
//*      data set that contains generated RMM ADDDATASET commands
//*****
//      SET EXTRACT='your.extended-extract.dsn'
//      SET OUTPUT='your.output.rmmad.clist'
//*****
//*      Specify space parms for output data set with generated
//*      RMM ADDDATASET CLIST
//*****
//      SET OUTPRI=x
//      SET OUTSEC=y
//      SET OUTSIZE='TRK | CYL'
//*****
//USECASE3 EXEC PGM=SORT
//SORTIN  DD  DISP=SHR,DSN=&EXTRACT
//SORTOUT DD  DSN=&OUTPUT,
//          DISP=(,CATLG,DELETE),
//          UNIT=SYSDA,SPACE=(&OUTSIZE,(&OUTPRI,&OUTSEC),RLSE)
//SYSOUT  DD  SYSOUT=*
//SYMNAMES DD DISP=SHR,DSN=SYS1.MACLIB(EDGEXTSY) SYMBOLS FROM EXTRACT
//SYMNOUT DD  SYSOUT=* DFSORT SYMBOLS WITH OFFSETS
//SYSIN   DD  *
          OPTION VLSCMP
          INCLUDE COND=(EXTRACT_TYPID,EQ,RXTYPEID,AND,           # Data records
                        RXDVRVAL,EQ,C'VRSMV')                   # VRSVAL
          SORT FIELDS=(RXDSNAME,A)                                # Dataset name
          OUTFIL VTOF,                                           # Fixed len output
          BUILD=(C' RMM AD ''',RXDSNAME,                          # RMM AD command
                C''' -',/,
                C' VOLUME(',RXVOLSER,
                C') FILESEQ(',RXDDSNSEQ,
                C') NOMANAGEMENTVALUE',
                80:X)
/*

```

---

A sample of the output of this JCL written to the SORTOUT DD is shown in Figure 15-4.

```

RMM AD 'SCOTTMR.TEST.EXPDT.TEST1          * -
VOLUME(B03486) FILESEQ(    1) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST1          * -
VOLUME(B03514) FILESEQ(    1) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST1          * -
VOLUME(B03526) FILESEQ(    1) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST1          * -
VOLUME(B03612) FILESEQ(    1) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST1          * -
VOLUME(B04961) FILESEQ(    1) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B00916) FILESEQ(    2) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B01253) FILESEQ(    2) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B01608) FILESEQ(    2) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B01800) FILESEQ(    2) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B02040) FILESEQ(    2) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B02079) FILESEQ(    2) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B02746) FILESEQ(    2) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B03231) FILESEQ(    2) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B03253) FILESEQ(    2) NOMANAGEMENTVALUE
RMM AD 'SCOTTMR.TEST.EXPDT.TEST2          * -
VOLUME(B03279) FILESEQ(    2) NOMANAGEMENTVALUE

```

Figure 15-4 RMM CHANGEDATASET commands

This example includes the following notable DD statements:

- ▶ SORTIN points to the input data, which is the extended extract data set records
- ▶ SORTOUT point to the output data set
- ▶ SYMNames points to SYS1.MACLIB(EDGEXTSY), which is the symbols mapping for the extract data set
- ▶ SYSIN has all the main, inline control statements

The main control statements in this DFSORT job are in the SYSIN DD and perform the following functions:

- ▶ INCLUDE records with RXTYPEID (that is, extended extract data set records, which contain volume and data set information for tapes) AND RXDVRSVAL (that is, the VRS management value field from the EDGEXTSY symbols mapping) equal (EQ) to VRSMV.
- ▶ SORT FIELDS starts a sort of the RXDSNAME field (that is, the data set name field from the EDGEXTSY symbols mapping) in ascending (A) order.
- ▶ Reformat the output records into RMM TSO CHANGEDATASET subcommands, in the following form:

```
RMM CD dsn VOLUME(volser) FILESEQ(data set sequence number) NOMANAGEMENTVALUE
```

In this case, it produces subcommands to remove the management value for data sets that were accidentally assigned a VRS management value of VRSMV

### 15.3.4 Use Case 4: Comparing EDGUTIL output with Extract

This use case describes the use of DFSORT to create commands that are required to read EDGUTIL output and issue commands to RMM to change the volume status to MASTER and extend the expiration date.

In this case, EDGUTIL with PARM='VERIFY(SMSTAPE)' is run to check whether any discrepancies exist between the library, volume catalogs, and RMM CDS.



The output in the SYSPRINT lists numerous EDG6823I messages. Each message indicates that a system-managed volume includes a status that is inconsistent with the value in the volume catalog (VOLCAT) or Library Manager database (LMDB).

The status of the volume information of the affected tapes that is recorded in RMM CDS, which is SCRATCH in this example, is incorrect. The status of PRIVATE in the VOLCAT and LMDB is correct.

This example describes how to correct the status of the volumes that are within the RMM CDS.

**Note:** If you are unsure how this discrepancy occurred, then capture diagnostic information so that IBM can perform root cause analysis.

Also, if you extend the retention period to one year, you can determine the correct retention policies for these tapes and correct the root cause of the discrepancies.

The provided sample JCL in Example 15-5 on page 90 shows how to use a JOINKEYS DFSORT job to perform the following tasks:

- ▶ Cross-reference the EDGUTIL output with the extract data set to build RMM TSO subcommands to push out the retention date for a year
- ▶ Correct the status that is recorded in the RMM CDS
- ▶ Capture all the volume records in the extract data set for the volumes that are getting the EDG6823I message.

Example 15-5 DFSORT Joinkeys JCL

```

//*****
//*      Specify input nonextended extract data set, which is
//*      the edgutil verify job log, the output data set
//*      with generated RMM CHANGEVOLUME CLIST, and the data set
//*      with matching volume records from extract data set
//*****
//      SET EXTRACT='your.non-extended-extract.dsn'
//      SET JOBLLOG='your.input.edgutil.verify.msgs'
//      SET OUTPUT='your.output.rmmcv.clist'
//      SET MATCH='impacted.volume.extract.recs.for.support'
//*****
//*      Specify space parms for output and match data sets
//*****
//      SET OUTPRI=x1
//      SET OUTSEC=y1
//      SET OUTSIZE='TRK | CYL'
//      SET MATCHPRI=x2
//      SET MATCHSEC=y2
//      SET MATCHSIZ='TRK | CYL'
//*****
//USECASE4 EXEC PGM=SORT
//SYSOUT DD SYSOUT=*
//SYMNAMES DD DISP=SHR,DSN=SYS1.MACLIB(EDGEXTSY) SYMBOLS FROM EXTRACT
//      DD *
**** EDGUTIL VERIFY OUTPUT SYMBOLS
MSG-ID,06,08,CH
MSG-VOL,22,6
**** OTHER SYMBOLS
VOL-OFFSET-LEN,9,6
REC-START-END,1,1110
//SYMNOUT DD SYSOUT=* DFSORT SYMBOLS WITH OFFSETS
//IN1 DD DISP=SHR,DSN=&EXTRACT
//IN2 DD DISP=SHR,DSN=&JOBLLOG
//JNF1CNTL DD *
    INCLUDE COND=(EXTRACT_TYPID,EQ,RVTYPEID)
//JNF2CNTL DD *
    INCLUDE COND=(MSG-ID,EQ,C'EDG6823I')
//RMMCLIST DD DSN=&OUTPUT,
//      UNIT=SYSDA,DISP=(,CATLG,DELETE),
//      SPACE=(&OUTSIZE,(&OUTPRI,&OUTSEC))
//VOLRECS DD DSN=&MATCH,
//      UNIT=SYSDA,DISP=(,CATLG,DELETE),
//      SPACE=(&MATCHSIZ,(&MATCHPRI,&MATCHSEC))
//SYSIN DD *
JOINKEYS F1=IN1,FIELDS=(RVVOLSER,A),TYPE=V # Volser in extract
JOINKEYS F2=IN2,FIELDS=(MSG-VOL,A),TYPE=V # Volser in EDG6823I msg
REFORMAT FIELDS=(F1:REC-START-END) # Keep EDGRVEXT recs
OPTION COPY # for vols with msg
OUTFIL FNAMES=RMMCLIST,VTOF, # Fixed len output
BUILD=(C'RMM CV ',VOL-OFFSET-LEN, # RMM CV command
      C' STATUS(MASTER) RETPD(365)',
      80:X)
OUTFIL FNAMES=VOLRECS, # Write EDGRVEXT recs
OUTREC=(REC-START-END) # for vols with msg
/*

```

The contents that are written to the VOLRECS DD resemble the output that is shown in Figure 15-5.

```
***** Top of Data *****
S.V. B04660 ..... | ..... 07/27/20161402242622 04/15/202111222244K 2524 12/31/1999 IDRY 1 5192 9 1203V
***** Bottom of Data *****
```

Figure 15-5 VOLRECS DD contents

A sample of the output of this JCL written to the RMM CLIST DD is shown in Figure 15-6.

```
***** Top of Data *****
RMM CV B04660 STATUS(MASTER) RETPD(365)
***** Bottom of Data *****
```

Figure 15-6 RMM ChangeVolume command

This example includes the following notable DD statements:

- ▶ SYMNames points to SYS1.MACLIB(EDGEXTSY), which is the symbols mapping for the extract data set or to inline DFSORT symbols where \* is used for position, which maps consecutive fields in your records without having to compute their actual positions.
- ▶ IN1 points to the 1st input data, which is the extended extract data set records.
- ▶ IN2 points to the 2nd input data, which is a data set where the output from an EDGUTIL VERIFY(SMSTAPE) job run was saved (by using the SYSPRINT DD from the EDGUTIL job).
- ▶ JNF1CNTL contains inline control statements that are used to preprocess IN1 data.
- ▶ JNF2CNTL contains inline control statements that are used to preprocess IN2 data.
- ▶ RMMCLIST points to an output data set, which contains generated RMM CV subcommands.
- ▶ VOLRECS points to the output data set, which parses the full volume records from the input extract data set for all the affected tapes. This diagnostic information can be used in root cause analysis.
- ▶ SYSIN has all the main, inline control statements.

In this JOINKEYS DFSORT job, the control statements in JNF1CNTL and JNF2CNTL process the input records before the main control statements in the SYSIN. The control statements in each respective DD perform the following functions:

- ▶ JNF1CNTL
  - INCLUDE records that start with RVTYPEID (that is, volume records in extract data set)
- ▶ JNF2CNTL
  - INCLUDE records with message EDG6823I
- ▶ SYSIN
  - JOINKEYS effectively joins the records and sorts on the specified key fields (that is, the volser in this case), which for IN1 is the volser that is found at offset 9 for a length of 6 bytes and for IN2 is found at offset 18 for a length of 6 bytes.

Then, the REFORMAT FIELDS statement outputs records that include the entire volume record (bytes 1–1110) from the IN1 data, that is, the extract data set.

Output is directed to FNAMES (that is, the output file names), which are pointed to by the RMMCLIST and VOLRECS DDs.

- ▶ For RMMCLIST, it BUILDS the RMM TSO CHANGEVOLUME subcommands in the following form:

```
RMM CV volser STATUS(MASTER) RETPD(365)
```

- ▶ In this case, it produces subcommands to correct the status for all the volumes getting flagged with the EDG6823I message by EDGUTIL VERIFY processing.

For VOLRECS, the OUTREC is writing out the entire volume record from the extract data set. This information can help with root cause analysis.

### 15.3.5 Additional use case information

The Extract file is a variable length, blocked data set. A 4-byte RDW field at the start of each record must be accounted for when referencing fields within control statements.

Also, when any macro mapping is referenced, notice that the offsets start at byte 0, but fields in DFSORT control statements start at offset 1. That difference, along with the 4-byte RDW, requires 5 bytes to be added to any offset found within the macro mapping for the extract data set in the use cases covered within this chapter).

For example, although each volume record in the extract data set starts with the character “V” at offset 0 according to the macro mapping, any control statements that reference that field must be coded with a starting position of 5.

Consider offsets that are calculated as in 15.3.4, “Use Case 4: Comparing EDGUTIL output with Extract” on page 88. When coding the offsets for the inline DFSORT symbols that represent the MSD-ID and MSG-VOL fields in the EDGUTIL VERIFY messages, each field’s starting position required adding 5 bytes to any offset seen when browsing the input data set.

The reason for this addition is because the record formatting for the input data set was variable length, blocked ASA print control characters(RECFM=VBA). Therefore, along with adding 4 bytes for the RDW, another 1-byte carriage control must be factored into any offset, which is why five was added to the offset of any field within the VBA record formatted data set.

Also, all use cases that generate RMM TSO subcommands to a specified data set can be run by using a batch job that uses the sample JCL that is shown in Example 15-6.

*Example 15-6 Running RMM TSO commands*

---

```
/* SUBMIT TSO RMM SUBCOMMANDS BUILT IN PREVIOUS JOB STEP */
//RMMCMD EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*,LRECL=100
//SYSTSIN DD DISP=SHR,DSN=rmm.clist
/*
```

---

After the CLIST is ready for execution (after the generated subcommands are thoroughly reviewed), the corresponding data set name must be specified in the SYSTSIN DD.

Also, the JCL parameters in the sample JCL that is provided in this chapter might need to be updated before they are used. Most notably, the DSN and SPACE parameters in the provided examples might need to be changed to account for the size of the input data that is being fed into these sample jobs.

For your convenience, JCL symbols were set at the beginning of each sample job so that the input and output data set names for the DSN parameter can be specified, along with the primary size and secondary size. The unit for these sizes (for example, TRK or CYL) can be specified in the top section of each job before execution.

### 15.3.6 Use Case 5: Generating DEFINE NONVSAM commands for recovered tape data sets

This example describes a method of using DFSORT to generate DEFINE NONVSAM commands for recently recovered tape data sets that were cataloged before being prematurely released to the scratch pool. The input data is the extended extract data set records, which was generated after recovering a range of tape data sets that were prematurely scrapped. The tapes were prematurely scrapped because a missing EDGUX100 user exit did not assign the management value of "CATALOG" to the tapes that were written with EXPDT=1999/000, which is the key date for retention of tapes while they are cataloged. Therefore, the corresponding "CATALOG" VRS was not matched during VRSEL processing and the tapes went to scratch before they were uncataloged.

For this use case, the RMM CDS volume and data set records were rebuilt for the recovered tapes. Now that the tapes are recovered, rebuild the catalog entries for these tape data sets. That means the recovered tape data sets have the management value they were originally intended to have, such as 'CATALOG', and that also means they are no longer in scratch status.

VRSEL processing has not run yet because the catalog entries are not rebuilt yet. That means the matching VRS name does not show "CATALOG" for these impacted tape data sets. Because "CATALOG" is not shown, you can use that information to identify which tape data sets need the DEFINE NONVSAM command generated. Other identifying characteristics, such as a range of days in which these impacted tape data sets were created, can be used within the INCLUDE statements in the example JCL provided for this use case.

To verify that the correct tape data sets were included in the generated output, the sample ICETOOL job also writes an output report with a list of tape data sets in the generated DEFINE NONVSAM output. For multivolume tape data sets, the DEFINE NONVSAM commands must include the correct sequential order of volumes in the multivolume chain and the corresponding physical file sequence number for the tape data set on each volume. Therefore, the report also includes volume sequence and file sequence information for each data set for validating the generated DEFINE NONVSAM commands. The sample JCL required in this use case is shown in Example 15-7.

*Example 15-7* Generate commands for creating single and multi volume datasets

---

```

//*****
//* PURPOSE : GENERATE COMMANDS FOR CREATING SINGLE AND MULTI VOLUME   *
//*          DATASETS.                                                  *
//*                                                                 *
//* INPUT:                                                                 *
//*   -> SPECIFY EXTENDED EXTRACT DATA SET IN "EXTRACT" JCL SYMBOL,   *
//*       WHICH IS USED IN EXTINP DD                                     *
//*   -> SPECIFY MANAGEMENT VALUE THAT IMPACTED TAPES SHOULD HAVE IN  *
//*       "MV" JCL SYMBOL                                               *

```

```

//*      -> SPECIFY DEVICE TYPE THAT IMPACTED TAPES SHOULD HAVE IN      *
//*      -> "DEVTYPE" JCL SYMBOL                                          *
//*      -> SPECIFY SPACE PARMS FOR TEMP DATA SETS, LARGE ENOUGH TO    *
//*      APPROXIMATELY HOLD THE EXTENDED EXTRACT DATA SET RECORDS     *
//*      FOR IMPACTED TAPES                                             *
//*      OUTPUT:                                                         *
//*      -> CMDSV DD HAS GENERATED DEFINE CMDS FOR SINGLE VOL DATA SETS *
//*      -> CMDSM DD HAS GENERATED DEFINE CMDS FOR MULTIVOL DATA SETS  *
//*      NOTE: OUTPUT IS BEING WRITTEN TO SPOOL IN THIS EXAMPLE        *
//*****
// SET EXTRACT=YOUR.EXTENDED.EXTRACT.DSN
// SET MV='CATALOG'
// SET DEVTYPE='3490' # DO NOT USE ESOTERIC & INSTEAD USE DEVICE NUMBER
// SET MVDSNPRI=x      # IN CYLINDERS
// SET MVDSNSEC=y      # IN CYLINDERS
// SET RPRTPRI=x       # IN CYLINDERS
// SET RPRTSEC=y       # IN CYLINDERS
/*
//USECASE5 EXEC PGM=ICETOOL,PARM='JPO"&MV",JP1"&DEVTYPE"'
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//SYMNOUT DD SYSOUT=*
//SYMNAMES DD DISP=SHR,DSN=SYS1.MACLIB(EDGEXTSY) SYMBOLS FROM EXTRACT
// DD *
** TEMPORARY FIELD DEFINITIONS **
TEMP-RDW,01,04,BI
TEMP-DSN-FIELD,*,44,CH
TEMP-VOLSEQ-FIELD,*,04,CH
TEMP-VOL-FIELD,*,06,CH
TEMP-FSEQ-FIELD,*,05,CH
GRPID-DEFINE-CMDS,81,08,ZD
GRP-SEQ-INDICATOR,*,05,CH
  GRP-SEQ,=,04,ZD
  GRP-SEQ-ID,*,01,CH
NEW-VOLSEQ-FIELD,*,04,CH
TEMP-REC-INDICATOR,*,04,CH
SKIP,1
TEMP-REC,*,80,CH
  TEMP-REC-TYPEA,=,64,CH
  TEMP-REC-TYPEB,*,04,CH
  TEMP-REC-TYPEC,*,12,CH
SKIP,1
NEW-KEY,*,09,CH
SKIP,1
NEW-VOL,*,06,CH
//EXTINP DD DISP=SHR,DSN=&EXTRACT
//MVDSN DD DSN=&&T1,DISP=(,PASS),SPACE=(CYL,(&MVDSNPRI,&MVDSNSEC),RLSE)
//CMDSV DD SYSOUT=*
//CMDSM DD SYSOUT=*
//TOOLIN DD *
  SELECT FROM(EXTINP) TO(MVDSN) ON(TEMP-DSN-FIELD) ALLDUPS -
  DISCARD(CMDSV) USING(CTLO)
  SORT FROM(MVDSN) TO(CMDSM) USING(CTL1)
/*
//CTLOCNTL DD *

```

```

OPTION VLSCMP
*-----*
* INCLUDE ONLY THE RECENTLY RECOVERED TAPE VOLUMES *
*-----*
INCLUDE COND=(RXTYPE,EQ,C'X',AND,
              RXVSTATUS,NE,C'SCRATCH',AND,      # VOL STATUS!=SCRATCH
              RXDVRSVAL,EQ,JPO,AND,            # MV==YOUR_MV
              RXDVRSNAM,NE,JPO,AND,           # VRS!=MV
              RXDCAT,NE,C'Y')                 # CATALOG != Y

*-----*
* BUILD TEMP RECORDS WITH FIELDS NEEDED FOR DEFINE CMDS *
*-----*
INREC BUILD=(RXRDW,                          # RDW
              RXDSNAME,                        # DATASET NAME
              RXDVOLSEQ,                      # VOL SEQ
              RXVOLSER,                       # VOLSER
              RXDDSNSEQ)                     # DS SEQ

*-----*
* SORT ON DSNAME AND VOLUME SEQUENCE NUMBER IN TEMP RECORDS *
*-----*
SORT FIELDS=(TEMP-DSN-FIELD,A,                # DATASET NAME ASC
              TEMP-VOLSEQ-FIELD,D)            # VOL SEQ DESC

*-----*
* BUILDS GROUPING OF VARIOUS SECTIONS FOR DEFINE CMDS FOR MULTIVOL DS*
*-----*
OUTFIL FNAMES=MVDSN,                          # MULTIVOL OUTPUT

*-----*
* GENERATE GROUPID WITH THE TEMP RECORD DSNAME FIELD AS THE KEY *
*-----*
IFTHEN=(WHEN=GROUP,KEYBEGIN=(TEMP-DSN-FIELD), # GRPID FOR DSN
        PUSH=(GRPID-DEFINE-CMDS:ID=8,
              GRP-SEQ:SEQ=4)),

*-----*
* IF GENERATED GROUPID IS GREATER THAN 0, THEN REFORMAT TEMP RECORD *
*-----*
IFTHEN=(WHEN=(GRPID-DEFINE-CMDS,GT,0),
        BUILD=(TEMP-RDW,TEMP-DSN-FIELD,        # DSN RECORDS
                GRPID-DEFINE-CMDS:GRPID-DEFINE-CMDS,
                GRP-SEQ,C'A',4X,/,
                TEMP-RDW,TEMP-FSEQ-FIELD:TEMP-FSEQ-FIELD,
                GRPID-DEFINE-CMDS:GRPID-DEFINE-CMDS, # FILESEQ RECORD
                GRP-SEQ,
                C'B',TEMP-VOLSEQ-FIELD,/,
                TEMP-RDW,TEMP-VOL-FIELD:TEMP-VOL-FIELD, # VOLSER RECORDS
                GRPID-DEFINE-CMDS:GRPID-DEFINE-CMDS,
                GRP-SEQ,
                C'C',TEMP-VOLSEQ-FIELD))

*-----*
* BUILD DEFINE CMDS FOR SINGLE VOLUME DATA SETS WITH NEEDED FIELDS *

```

```

*-----*
OUTFIL FAMES=CMDSV,VTOF,                # SINGLE VOL CMDS
BUILD=(01:C' DEFINE NONVSAM',
      66:C'-',
      /,
*-----*
* LEAD + DSN + TRAIL (7+44+2) = 53 BYTE FIELD FOR JUSTIFY SHFT LEFT *
*-----*
      01:TEMP-DSN-FIELD,JFY=(SHIFT=LEFT,
        LEAD=C' NAME('',
        TRAIL=C''')',
        LENGTH=53),
      66:C'-',
      /,
      01:C' DEVICETYPES(',JP1,C') ',
      66:C'-',
      /,
*-----*
* LEAD + FILESEQ + TRAIL(7+4+2) = 12 BYTE FIELD FOR JUSTIFY SHFT LEFT*
*-----*
      01:TEMP-FSEQ-FIELD,JFY=(SHIFT=LEFT,
        LEAD=C' FSEQN(',
        TRAIL=C')',
        LENGTH=12),
      66:C'-',
      /,
*-----*
* LEAD + VOLSER + TRAIL (9+6+2) = 17 BYTE FIELD FOR JUSTIFY SHFT LEFT*
*-----*
      01:TEMP-VOL-FIELD,JFY=(SHIFT=LEFT,
        LEAD=C' VOLUMES(',
        TRAIL=C'))',
        LENGTH=17),
      80:X)
/*
//CTL1CNTL DD *
  OPTION VLSHRT
*-----*
* OMIT RECORDS WITH DUP DSNAME & GROUP ID (1ST PART OF DEFINE CMD) *
*-----*
      OMIT COND=(GRP-SEQ,GT,1,AND,                # OMIT DUP DSN
                GRP-SEQ-ID,EQ,C'A')              # RECORDS
*-----*
* OVERLAY INDICATOR W/ SYNTACTICALLY CORRECT ENDING TO CMD SECTION *
*-----*
      INREC OVERLAY=(TEMP-REC-INDICATOR:GRP-SEQ-INDICATOR,
        CHANGE=(4,C'0001B',C' ) -',          # CONTINUE CHAR
                C'0001C',C' ) )',          # ENDING CHAR
        NOMATCH=(C' - ')                    # CONTINUE CHAR
*-----*
* SORT ON GROUPID, THEN SEQ NUM FOR EACH REC IN GROUP, AND VOLSEQ *
*-----*
      SORT FIELDS=(GRPID-DEFINE-CMDS,A,                # GROUPID ASC

```



```

GRP-SEQ-ID,A, # RECID ASC
NEW-VOLSEQ-FIELD,A) # VOLSEQ ASC

```

```

*-----*
* CREATE A NEW GROUPID AND SEQ NUM FOR EACH REC IN GROUP AND VOLSEQ *
*-----*

```

```

OUTREC IFTHEN=(WHEN=INIT,
                OVERLAY=(NEW-KEY:GRPID-DEFINE-CMDS,GRP-SEQ-ID)),
IFTHEN=(WHEN=GROUP,KEYBEGIN=(NEW-KEY),
        PUSH=(GRP-SEQ:SEQ=4)),

```

```

*-----*
* BUILD DSNNAME PART OF DEFINE CMD WHEN INDICATOR == 0001A *
* LEAD + DSN + TRAIL (7+44+2) = 53 BYTE FIELD FOR JUSTIFY SHFT LEFT *
*-----*

```

```

IFTHEN=(WHEN=(GRP-SEQ-INDICATOR,EQ,C'0001A'),
        OVERLAY=(TEMP-REC-TYPEA:TEMP-DSN-FIELD,
                JFY=(SHIFT=LEFT,
                    LEAD=C' NAME('',
                    TRAIL=C'''),
                    LENGTH=53),
                TEMP-REC-TYPEB:TEMP-REC-INDICATOR,
                TEMP-REC-TYPEC:X)),

```

```

*-----*
* BUILD 1ST FILESEQ PART OF DEFINE CMD WHEN INDICATOR == 0001B *
* LEAD + FILESEQ (7+4) = 11 BYTE FIELD FOR JUSTIFY SHFT LEFT *
*-----*

```

```

IFTHEN=(WHEN=(GRP-SEQ-INDICATOR,EQ,C'0001B'),
        OVERLAY=(TEMP-REC-TYPEA:TEMP-FSEQ-FIELD,
                JFY=(SHIFT=LEFT,
                    LEAD=C' FSEQN(',
                    LENGTH=11),
                TEMP-REC-TYPEB:TEMP-REC-INDICATOR,
                TEMP-REC-TYPEC:X)),

```

```

*-----*
* REMAINING FILESEQ RECORDS WHEN GROUP SEQ > 1 & ID == B *
* LEAD + FILESEQ (7+4) = 11 BYTE FIELD FOR JUSTIFY SHFT LEFT *
*-----*

```

```

IFTHEN=(WHEN=(GRP-SEQ,GT,1,AND,GRP-SEQ-ID,EQ,C'B'),
        OVERLAY=(TEMP-REC-TYPEA:TEMP-FSEQ-FIELD,
                JFY=(SHIFT=LEFT,
                    LEAD=C' ',
                    LENGTH=11),
                TEMP-REC-TYPEB:TEMP-REC-INDICATOR,
                TEMP-REC-TYPEC:X)),

```

```

*-----*
* BUILD 1ST VOLUME PART OF DEFINE CMD WHEN INDICATOR == 0001C *
* LEAD + VOLSER (9+6) = 15 BYTE FIELD FOR JUSTIFY SHFT LEFT *
*-----*

```

```

IFTHEN=(WHEN=(GRP-SEQ-INDICATOR,EQ,C'0001C'),
        OVERLAY=(TEMP-REC-TYPEA:TEMP-VOL-FIELD,

```

```

                                JFY=(SHIFT=LEFT,
                                  LEAD=C' VOLUMES(',
                                  LENGTH=15),
                                TEMP-REC-TYPEB:TEMP-REC-INDICATOR,
                                TEMP-REC-TYPEC:X)),

*-----*
* REMAINING VOLUME RECORDS WHEN GROUP SEQ > 1 & ID == C *
* LEAD + VOLSER & CONT CH (9+8) = 17 BYTE FIELD FOR JUSTIFY SHFT LEFT*
*-----*
                                IFTHEN=(WHEN=(GRP-SEQ,GT,1,AND,GRP-SEQ-ID,EQ,C'C'),
                                  OVERLAY=(TEMP-REC-TYPEA:TEMP-VOL-FIELD,
                                              JFY=(SHIFT=LEFT,
                                                  LEAD=C'      ',
                                                  LENGTH=17),
                                              TEMP-REC-TYPEB:TEMP-REC-INDICATOR,
                                              TEMP-REC-TYPEC:X))

*-----*
* HEADER3 BUILDS 1ST PART OF DEFINE CMD FOR EACH SECTION/GROUP *
*-----*
                                OUTFIL REMOVECC,VTOF,
                                BUILD=(TEMP-REC),
                                SECTIONS=(GRPID-DEFINE-CMDS,
                                HEADER3=(' DEFINE NONVSAM',66:'-',/,
                                          ' (DEVICETYPES(',JP1,C') ',66:'-'),
                                TRAILER3=())

/*
//*****
//* PURPOSE : GENERATE REPORT OF TAPES TO BE CATALOGED. *
//* *
//* INPUT: *
//* -> EXTENDED EXTRACT DATA SET SPECIFIED EARLIER IN "EXTRACT" JCL *
//* SYMBOL, WHICH IS USED IN INPUT DD *
//* -> MANAGEMENT VALUE THAT IMPACTED TAPES SHOULD HAVE, WHICH WAS *
//* SPECIFIED EARLIER IN "MV" JCL SYMBOL *
//* OUTPUT: *
//* -> REPORT DD HAS INFO ON LIST OF TAPE DATA SETS THAT HAVE *
//* DEFINE CMDS GENERATED IN PREVIOUS JOB STEP (USECASE5) *
//* NOTE: OUTPUT IS BEING WRITTEN TO SPOOL IN THIS EXAMPLE *
//*****
//REPORT EXEC PGM=ICETOOL,PARM='JPO"&MV"'
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//INPUT DD DISP=SHR,DSN=&EXTRACT
//TEMPRPT DD DSN=&&RPRT,DISP=(,PASS),SPACE=(CYL,(&RPRTPRI,&RPRTSEC),RLSE)
//SYMNAMES DD DISP=SHR,DSN=SYS1.MACLIB(EDGEXTSY) SYMBOLS FROM EXTRACT
//*-----*
//REPORT DD SYSOUT=*
//*-----*
//TOOLIN DD *

*-----*
* TOOLIN STATEMENTS GENERATES REPORT OF TAPES TO BE CATALOGED *
*-----*

```

```

*   GENERATE A REPORT OF TAPES THAT NEED TO BE CATALOGED   *
*-----*
COPY FROM(INPUT) TO(TEMPRPT) USING(RPRT)
DISPLAY FROM(TEMPRPT) LIST(REPORT)           -
TITLE('DFSMSRMM - TAPE DATA SETS BEING RECATALOGED') -
DATE TIME PAGE                               -
BLANK                                         -
HEADER('DSNAME')          ON(RXDSNAME)       -
HEADER('VOLUME')         ON(RXVOLSER)        -
HEADER('VSEQ')           ON(RXDVOLSEQ,CH)    -
HEADER('FSEQ')           ON(RXDDSSEQ)        -
HEADER('VOL','CNT')      ON(RXVOLCNT)        -
HEADER('PREV','VOLUME') ON(RXVPVOL)         -
HEADER('NEXT','VOLUME') ON(RXVNVOL)         -
HEADER('CAT','STAT')    ON(RXDCAT)          -
/*
//RPRTCNTL DD *
  OPTION VLSCMP
  INCLUDE COND=(RXTYPE,EQ,C'X',AND,
               RXVSTATUS,NE,C'SCRATCH',AND, # VOL STATUS!=SCRATCH
               RXDVRSVAL,EQ,JPO,AND,       # MV==YOUR_MV
               RXDVRSNAM,NE,JPO,AND,       # VRS!=MV
               RXDCAT,NE,C'Y')             # CATALOG != Y

  SORT FIELDS=(RXDSNAME,A,
               RXDVOLSEQ,A)
/*

```

This example job has 2 job steps, USECASE5 and REPORT. Each step is discussed in the following sections.

## USECASE5

The USECASE5 job step includes the following notable DD statements:

- ▶ The input data for the first job step is the extended extract data set records, which are read in the EXTINP DD statement and specified within the JCL symbol “EXTRACT” at the top of the job. In a real-world scenario, the extended extract data set records are generated after recovering the tape volumes that were prematurely released to the scratch pool. The volumes were prematurely released because they were not assigned the correct management value for catalog retention when originally written.
- ▶ A temporary data set is specified in the MVDSN DD statement. At the beginning of the job, JCL symbols are defined for users to specify both the primary and secondary space parameters for this temporary data set.
- ▶ The SYMNames DDs are used to reference both the SYS1.MACLIB(EDGEXTSY) macro mapping for the extended extract data set and the hardcoding inline DFSORT symbols for reformatting various fields in later control statements.
- ▶ The output for the job step is written to the CMDSV and CMDSM DD statements.
  - CMDSV contains the output DEFINE NONVSAM commands to rebuild the catalog entries for single volume tape data sets.
  - CMDSM contains the output DEFINE NONVSAM commands to rebuild the catalog entries for multivolume tape data sets.
- ▶ TOOLIN has all the main, inline control statements for the ICETOOL job step.

- ▶ CTLOCNTL contains inline control statements that are used to process both single volume and multivolume tape data set records written to CMDSV and MVDSN respectively.
- ▶ CTL1CNTL contains inline control statements that are used to finish reformatting multivolume tape data set records written to CMDSM.

The following main control statements are used:

- ▶ SELECT FROM(EXTINP) TO(MVDSN) ON(TEMP-DSN-FIELD) ALLDUPS - DISCARD(CMDSV) USING(CTL0)
  - This copies data from the input extended extract records to both the single volume CMDSV DD, and the multivolume tape data set records are written to the temporary MVDSN DD by using the inline control statements in the CTLOCNTL DD.
  - The ALLDUPS routes records with duplicate keys ON the TEMP-DSN-FIELD to the MVDSN DD, that is, records with duplicate DSN, because these are multivolume data sets. DISCARD is routing nonduplicate records to the CMDSV DD, that is, records with a unique DSN because they are single volume data sets.
  - CTLOCNTL performs the following functions:
    - Includes only extended extract data set records with the conditions specified within the INCLUDE statement. The conditions are volume status not equal to “SCRATCH”, management value equal to a specified catalog control VRS, matching VRS name not equal to a specified catalog control VRS, and a catalog status not equal to “Y”.
    - INREC statement is building temporary records with fields that are needed for output of DEFINE NONVSAM commands.
    - The SORT statement is sorting on the DSN field in the temporary records in ascending order. The SORT statement also sorts the volume sequence number in the temporary records in descending order so that the last volume in a multivolume chain comes first and so that the last volser in a multivolume chain can be more easily identified.
    - OUTFIL for MVDSN uses IFTHEN statements to generate group identifiers for groups of records with the same DSN, that is, grouping the multivolume data sets. for the generation of DEFINE NONVSAM commands. Then it builds new temporary records where the DSN, FILESEQ, and VOLSER fields are split into their own records and marked with “A”, “B”, and “C”, respectively. The records are marked to identify each section for the corresponding parameters in the generated DEFINE NONVSAM commands later in the job.
    - OUTFIL for CMDSV builds the DEFINE NONVSAM commands for single volume data sets. There is no need to do grouping for these records because each single record has all the fields that are needed to generate the DEFINE NONVSAM commands for single volume data sets.
- ▶ SORT FROM(MVDSN) TO(CMDSM) USING(CTL1)
  - SORTs records from MVDSN to CMDSM USING the CTL1CNTL DD inline control statements.
  - CTL1CNTL performs the following functions:
    - OMIT statements omit records with duplicate DSN and group sequence identifier of “A”, which is the DSN portion of the DEFINE NONVSAM commands.
    - INREC statement overlays the records with the corresponding parameter terminators and continuation characters for each parameter section in the DEFINE NONVSAM commands.

- SORT statement sorts on the group identifier for the grouping of records for each DEFINE NONVSAM command (in ascending order), group sequence identifier (in ascending order), and volume sequence number (in ascending order so that the first volser in the volume chain now comes first).
- OUTREC contains a series of IFTHEN and OVERLAY statements that finish much of the record reformatting for the DEFINE NONVSAM commands.
- OUTFIL finishes building the DEFINE NONVSAM commands for the multivolume data sets.

## REPORT

The REPORT job step includes the following notable DD statements:

- ▶ The input data for this job step is also the extended extract data set records, which are read in the INPUT DD statement and specified in the JCL symbol “EXTRACT” at the top of the job. Again, in this scenario, the extended extract data set records are generated after recovering the tape volumes that were prematurely released to the scratch pool. The volumes were prematurely released because they were not assigned the correct management value when originally written.
- ▶ A temporary data set is specified in the TEMPRPRT DD statement. At the beginning of the job, JCL symbols are defined for users to specify both the primary and secondary space parameters for this temporary data set.
- ▶ The SYMNAMES DDs are used to reference the SYS1.MACLIB(EDGEXTSY) macro mapping for the extended extract data set.
- ▶ The output for the ICETOOL job is written to the REPORT DD statement.
  - REPORT outputs a list of the tape data sets getting a DEFINE NONVSAM command generated within the CMDSV and CMDSM output in the previous job step. The report details the DSN, volser, volume sequence number, file sequence number, total volume count in a multivolume set, previous volume in chain, next volume in chain, and catalog status.
- ▶ TOOLIN has all the main, inline control statements for the ICETOOL job step.
- ▶ RPRTCNTL contains inline control statements that are used to process the REPORT data.

The following main control statements are used:

- ▶ COPY FROM(INPUT) TO(TEMPRPRT) USING(RPRT)
  - This statement copies data from the input extended extract records to the temporary TEMPRPRT data set USING the control statements in RPRTCNTL DD.
  - RPRTCNTL performs the following functions:
    - Includes only extended extract data set records with the conditions that are specified within the INCLUDE statement. The conditions are volume status not equal to “SCRATCH”, management value equal to a specified catalog control VRS, matching VRS name not equal to a specified catalog control VRS, and a catalog status not equal to “Y”.
    - The SORT statement is sorting on the DSN field in the temporary records (in ascending order), and on the volume sequence number in the temporary records (in ascending order).

- ▶ DISPLAY FROM(TEMPRPRT) LIST(REPORT)
  - Displays the reformatted volume records from the temporary “TEMPRPRT” data set to the “REPORT” output
  - The DISPLAY performs the following functions:
    - Creates a TITLE for the report
    - Prints a DATE and TIME on each PAGE of the report
    - Creates a HEADER for each column or field within the report by using the symbols that are defined in the SYS1.MACLIB(EDGEXTSY) macro mapping



# Using EDGRMMxx global defaults for assignment of retention attributes case study

This appendix includes a discussion of the first in a series of case studies to help understand the newer features of DFSMSrmm. The case study illustrates the use of the EDGRMMxx parmlib member default statements to assign expiration criteria.

This appendix includes the following topics:

- ▶ “Introduction” on page 104
- ▶ “Case Study” on page 104
- ▶ “Summary” on page 108

## Introduction

A series of case studies are presented to better understand how the various options and new functions in DFSMSrmm work together.

The EXPDT retention method in DFSMSrmm can dynamically adjust expiration data rather than rely on the step in EDGHSKP that runs the VRSEL function. As more data sets leave the VRSEL environment, the processing time and elapsed time decreases in the daily EDGHSKP housekeeping cycle.

The implementation of expiration processing in DFRMM includes many different options. The cases in the appendices begin with the simplest case studies and progress to more complex cases such as in Appendix B, “Using Defaults Table for expiration attributes assignment case study” on page 109 and in Appendix F, “Using MCATTR(ALL) instead of MCATTR(VRSELXDI) case study” on page 147.

The conclusion of each case study includes a discussion of the various features that are demonstrated and some best practices tips and hints.

## Case Study

The first case study is a scenario in which the default retention method is set to EXPDT globally in the EDGRMMxx parmlib member along with the wanted subparameters.

In this case study, Generation Data Group (GDG) data sets are expired as their catalog entry is deleted, which occurs when they roll out of the base GDG. Non-GDG data sets are expired when they reach their expiration date or when their catalog entry are met first.

## Testing without the use of tape expiration values from the SMS MC

In this case study, the testing is done without the use of tape expiration values from the SMS MC, thus a default value in the EDGRMMxx parmlib member of MCATTR(NONE). Also, a Defaults Table is not specified and the EDGUX100/UXTABLE is not being used.

The EDGRMMxx parmlib member example that is shown Figure A-1 includes the relevant statements.

```
RETENTIONMETHOD(EXPDT(GDG(WHILECATALOG(ON) RETPD(1)) -  
NOGDG(WHILECATALOG(UNTILEXPIRED) RETPD(1)) RETAINBY(SET))) -
```

Figure A-1 EDGRMMxx parmlib member RM defaults



RMM LISTCONTROL(LC) ALL output lists the default expiration values, as shown in Figure A-2.

```
Exit status:
  EDG_EXIT100 = NONE

Retention period: Default = 5          Maximum = NOLIMIT
                  Catalog = 12       hours
Use of Management Class Attributes = NONE
Retention method: Default = EXPDT

RM(VRSEL) defaults:
  Retain by      = VOLUME
  Move By       = VOLUME
  VRS selection  = NEW
  VRS change     = INFO
  VRSMIN action  = FAIL
  VRSMIN count   = 1
  VRS job name   = 2
  GDG duplicate  = BUMP
  GDG cycle by   = GENERATION

RM(EXPDT) defaults:
  Retain by      = SET
  LASTREF extra days = 0
  Catalog Days   = 2
  GDG      : WHILECATALOG = ON
             RETPD       = 1
  Non-GDG: WHILECATALOG = UNTILEXPIRED
             RETPD       = 1

Defaults table definitions:
NO INSTALLATION DEFTABLE IS IN USE
```

Figure A-2 RMM LC ALL output

When a GDG is rolled out of the base, its catalog entry is deleted. When the attribute WHILECATALOG(ON) is used, the expiration criterion then becomes the expiration date that was set at data set creation.

In this case, the default for that expiration date setting RETPD is 1 day. Therefore, unless a JCL-provided or SMS Data Class value overrides it, the expiration date passes and the data set is expired in the next run of EDGHSKP EXPROC.

For a non-GDG data set, the WHILECATALOG(UNFILEXPIRED) attribute means that when the data set's catalog entry is deleted or the data set reaches its current expiration date, it is expired.

When the catalog entry is deleted, RMM adjusts the expiration date to the current date, plus the value of Catalog Days. Then, the data set expires during EDGHSKP EXPROC on or after that calculated date.

By using an IEBGENER job with no JCL-passed EXPDT or RETPD values, a GDG data set is created on a new volume. See Figure A-3 on page 106. The GDG data set entry shows the expiration date one day in the future, which set by = OCE\_DEF indicates came from the RETPD under RM(EXPDT) defaults.

It also features the value KeptByCat1g next to the expiration date and WHILECATALOG = ON indicates that when this data set's catalog entry is deleted and passed 9/23/2021, it is expired during EXPROC.

Also, VRSEL exclude = YES is assigned so that this data set is ineligible for processing during EDGHSKP VRSEL processing and no defined VRS rules affect expiration processing for this data set and the volume it is on. See Figure A-3 on page 106.

```

Data set name = KARIM.TEST.GDG.G0001V00
Volume       = B01661 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 09/22/2021 Create time = 14:03:20
Expiration date = 09/23/2021(KeptByCatlg)
Expir. time = 15:00:00
              set by      = OCE_DEF      Original expir.date =
LASTREF Extra Days = 0          WHILECATALOG      = ON
Management class = MCVTS1      VRS management value =
Storage group    = SGVTS1      VRS retention date   =
Storage class    = SC3490      VRS retained         = NO
Data class       = DC3490      Closed by Abend      = NO
                                   Deleted          = NO
VRSEL exclude   = YES          Catalog status       = YES
Primary VRS details:
  Name           =
  Job name       =              Type           =
  Subchain NAME =              Subchain start date =
Secondary VRS details:
  Value or class =
  Job name       =
  Subchain NAME =              Subchain start date =

```

Figure A-3 RMM LD output.

In Figure A-4, the volume to which the GDG data set was written takes its volume expiration attributes from the data set and specifies Datasets Kept By Catlg: 1, which indicates one catalog entry exists for a data set that is holding this volume from expiring.

Although the data set catalog entry is deleted, the expiration date must also be reached for the volume to be expired and returned to SCRATCH.

In the LISTVOLUME display that is shown in Figure A-4, Retention method = EXPDT is specified with retainby = SET, and both show the set by = OCE\_DEF value that indicates these attributes were taken from the EDGRMMxx parmlib member values.

```

Volume = B01661 VOL1 = Rack = Owner = KARIM
Type = LOGICAL Stacked count = 0 Jobname = KARIMOP
Worldwide ID = WORM = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign: Date = 09/22/2021 Time = 14:03:20 System ID = ZS24
User ID = KARIM
Expiration date = 09/23/2021(KeptByCatlg) Original =
Expiration time = 15:00:00 Datasets Kept By Catlg: 1 on Volume
set by = OCE_DEF
Retention date = Set retained = NO
Retention method= EXPDT
set by = OCE_DEF
retain by = SET
Data set name = KARIM.TEST.GDG.G0001V00

```

Figure A-4 RMM LV display

Because this volume is EXPDT managed, which is determined the first time a data set is written to the volume, no value for Retention date exists and none is assigned. because only the data sets on the volume include the VRSEL exclude = YES value, which is applied by default for all data sets on a volume that is managed by using EXPDT.

By using an IEBGENER job, a non-GDG data set is created with no JCL passed EXPDT or RETPD values. See Figure A-5.

```

Data set name = KARIM.TEST.NOGDG2
Volume       = B02029   Physical file sequence number = 1
Owner        = KARIM    Data set sequence = 1
Create date  = 09/22/2021 Create time = 14:16:51
Expiration date = 09/23/2021(OrUncatlg)
Expir. time = 15:00:00
              set by    = OCE_DEF      Original expir.date =
LASTREF Extra Days = 0          WHILECATALOG = UntilExpired
Management class = MCVTS1      VRS management value =
Storage group    = SGVTS1      VRS retention date =
Storage class    = SC3490      VRS retained = NO
Data class       = DC3490      Closed by Abend = NO
                                   Deleted = NO
                                   Catalog status = YES
VRSEL exclude   = YES
Primary VRS details:
  Name          =
  Job name      =              Type =
  Subchain NAME =              Subchain start date =
Secondary VRS details:
  Value or class =
  Job name      =
  Subchain NAME =              Subchain start date =

```

Figure A-5 RMM LD display NOGDG

In Figure A-5, the data set entry shows the expiration date one day in the future, which set by = OCE\_DEF indicates came from the RETPD under RM(EXPDT) defaults. It also has the value OrUncatlg next to the expiration date and WHILECATALOG = UNTILEXPIRED, which indicates that it is eligible for expiration when this data set passes its expiration date or has its catalog entry deleted.

If the data set has the catalog entry deleted *before* its expiration date of 9/23/2021, DFSMSrmm adjusts the expiration date to the current date plus the value of Catalog Days and then, expires during EDGHSKP EXPROC on that calculated date.

Because CATLGDAYS is not specified in the EDGRMMxx parmlib member statements, the default of 2 days is used, as seen in the RMM LC ALL output in Figure A-2 on page 105. Also, VRSEL exclude = YES is assigned so that this data set is ineligible for processing during EDGHSKP VRSEL processing, and no defined VRS rules affect expiration processing for this data set and the volume on which it is stored.

The volume to which this data set was written takes its volume expiration attributes from the data set. It specifies Datasets Kept By Catlg: 0, which indicates no data sets on this volume are held by catalog entry alone.

Also, as shown in Figure A-6, the Retention method = EXPDT, retainby = SET, and set by = OCE\_DEF values that show all of these attributes were taken from the EDGRMMxx parmlib member values. Because this volume is EXPDT-managed (which is determined the first time a data set is written to the volume), no value for Retention date exists and none are assigned because only data sets on the volume have the VRSEL exclude = YES value (see Figure A-6).

```

Volume = B02029   VOL1 =           Rack =           Owner   = KARIM
  Type = LOGICAL           Stacked count = 0           Jobname = KARIMOP
  Worldwide ID =           WORM      = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign:   Date = 09/22/2021 Time = 14:16:51 System ID = ZS24
                                           User ID  = KARIM
Expiration date = 09/23/2021(OrUncatlg) Original =
Expiration time = 15:00:00 Datasets Kept By Catlg: 0 on Volume
      set by = OCE_DEF
Retention date =           Set retained = NO
Retention method= EXPDT
      set by = OCE_DEF
      retain by = SET
Data set name = KARIM.TEST.NOGDG2

```

Figure A-6 RMM LV display NOGDG

## Summary

The setup for this case study relies on no ACS routine changes, no Defaults Table coding, and no EDGUX100/UXTABLE.

It handles data GDG data sets on tape much as data sets on SMS DASD are handled in that the data set is expired after the catalog entry is deleted.

For non-GDG data sets, the data is expired when its catalog entry is deleted or when it reaches the expiration date, a process that is similar to the DFHSM setting EXPIREDDATASETS(SCRATCH). This process partially achieves a level of device independence, so the user does not have to be concerned if the data set exists on tape or on DASD.

It also demonstrates the best practice of the use of RETAINBY (SET) so that all volumes in a volume chain are kept with the same expiration date and expire together.



# B

## **Using Defaults Table for expiration attributes assignment case study**

This chapter illustrates the use of the DFSMSrmm Default Table to assign expiration criteria. This case is the second in a series of case studies to help understand the newer features of DFSMSrmm.

In this case study, the EDGRMMxx parmlib member MCATTR is set to NONE so that the SMS Management Class is not used to assign tape expiration values.

Although no EDGUX100/UXTABLE exists, the Defaults Table is used to assign specific expiration attributes to the test data sets. The DFSMSrmm EDGRMMxx parmlib member example that is shown in Figure B-1 includes the retention method-related statements. However, the Defaults Table assignments overrides these statements.

```
RETENTIONMETHOD (EXPDT (GDG (WHILECATALOG (ON) RETPD (1) ) -
NOGDG (WHILECATALOG (UNTILEXPIRED) RETPD (1) ) RETAINBY (SET) ) ) -
DEFTABLE (T2) -
```

Figure B-1 EDGRMMxx parmlib member defaults

The Defaults Table that is used in this case study is shown in Figure B-2.

```
DEFAULT DSN(KARIM.TEST.DEFTABLE.**.G%V00) -
WHILECATALOG(ON) RM(EXPDT) RETPD(3)
DEFAULT DSN(KARIM.TEST.DEFTABLE.***) -
WHILECATALOG(UNTILEXPIRED) RM(EXPDT) RETPD(3)
DEFAULT DSN(KARIM.TEST.DEFTABLE.VRS.***) -
RM(VRSEL) VRSVALUE(D99000)
```

Figure B-2 EDGDEFxx contents

The statements in Figure B-1 generate the RMM LC ALL output that is shown in Figure B-3.

```
Exit status:
EDG_EXIT100 = NONE

Retention period: Default = 5          Maximum = NOLIMIT
                  Catalog = 12       hours
Use of Management Class Attributes = NONE
Retention method: Default = EXPDT

RM(VRSEL) defaults:          RM(EXPDT) defaults:
  Retain by                  = VOLUME          Retain by                  = SET
  Move By                    = VOLUME
  VRS selection              = NEW
  VRS change                 = INFO
  VRSMIN action              = FAIL
  VRSMIN count               = 1
  VRS job name               = 2
  GDG duplicate              = BUMP
  GDG cycle by               = GENERATION
  LASTREF extra days        = 0
  Catalog Days               = 2
  GDG : WHILECATALOG        = ON
  RETPD                      = 1
  Non-GDG: WHILECATALOG     = UNTILEXPIRED
  RETPD                      = 1

Defaults table definitions:
DEFTABLE is loaded from PARMLIB member EDGDEFT2, number of entries: 3
```

Figure B-3 RMM LISTCONTROL ALL

In Figure B-4, the contents of the EDGDEFxx member that is active with DFSMSrmm is displayed by issuing the RMM LISTCONTROL DEFTABLE command.

```

F DFRMM,CMD=LC DEFTABLE
DEFTABLE is loaded from PARMLIB member EDGDEFT2, number of entries: 3
DEFTABLE content:
-----
DSNAME mask: KARIM.TEST.DEFTABLE.**.G%V00                                CONTINUE:
JobName mask: *                PGMNAME: *                KeyDate: *
RETPD: 3      RO: N  EDM:  POOL:                RM: EXPDT
(VRSEL) VRSvalue:                VX:
(EXPDT) LastRef:                WHILECAT: ON  RetainBy:
-----
DSNAME mask: KARIM.TEST.DEFTABLE.**                                CONTINUE:
JobName mask: *                PGMNAME: *                KeyDate: *
RETPD: 3      RO: N  EDM:  POOL:                RM: EXPDT
(VRSEL) VRSvalue:                VX:
(EXPDT) LastRef:                WHILECAT: UX  RetainBy:
-----
DSNAME mask: KARIM.TEST.DEFTABLE.VRS.**                                CONTINUE:
JobName mask: *                PGMNAME: *                KeyDate: *
RETPD:        RO:    EDM:  POOL:                RM: VRSEL
(VRSEL) VRSvalue: D99000        VX:
(EXPDT) LastRef:                WHILECAT:                RetainBy:
-----

```

Figure B-4 RMM LC DEFTABLE output

The first entry in this table is defined so as to match a data set name that starts with KARIM.TEST.DEFTABLE, has any number of qualifiers in the middle. The first entry ends in a pattern that indicates it is part of a generation data group. It assigns a retention period (RETPD) of 3 days and applies the WHILECATALOG(ON) attribute.

A data set with the same data set pattern but without the generation information in the low-level qualifier does not match on this first table entry. Therefore, the search continues to the second entry.

The second table entry matches and the search stops and assigns RETPD 3 and WHILECATALOG(Untilexpired).

The third entry in this table checks for a data set name pattern that is similar to the first two entries but with VRS in the fourth qualifier. However, the code in this Default Table contains a logic error.

A data set that is allocated with KARIM.TEST.DEFTABLE.VRS matches the second entry on the table, stops parsing table values, and is assigned RM(EXPDT). However, the intention in this case is to match the third entry and to assign RM(VRSEL).

Resolve the error by adding CONTINUE to the second table entry so that the third table entry is also matched. CONTINUE is used so that multiple entries can match. The expiration attributes are taken from all matches that are assigned. Any conflicting assignments are taken from the last match. Therefore, adding CONTINUE on the second entry in the table allowed the third entry to match, and the RM attribute was correctly assigned to VRSEL.



As shown in Figure B-5, the result of the test without the CONTINUE parameter showed that the search was stopped after matching the second table entry. The data set was assigned EXPDT after matching the second table entry, so the volume was assigned RM(EXPDT) instead of VRSEL.

```

Volume = B02364   VOL1 =           Rack =           Owner   = KARIM
  Type = LOGICAL           Stacked count = 0           Jobname = KARIMOP
  Worldwide ID =           WORM      = N
Creation:  Date = 07/27/2016  Time = 14:32:24  System ID = ZS22
Assign:    Date = 09/22/2021  Time = 14:41:42  System ID = ZS24
                                           User ID   = KARIM
Expiration date = 09/25/2021(OrUncatlg)  Original =
Expiration time = 16:00:00  Datasets Kept By Catlg: 0      on Volume
      set by = DEFTABLE
Retention date =           Set retained = NO
Retention method= EXPDT
      set by = DEFTABLE
      retain by = SET
Data set name = KARIM.TEST.DEFTABLE.VRS.TEST

```

Figure B-5 CONTINUE needed

In Figure B-6, the result of the test with CONTINUE coded on the second table entry shows the data set and volume correctly being assigned RM(VRSEL). The set by field for the data set and volume shows DEFTABLE, VRSEL exclude = NO at data set level, and RM(EXPDT) at volume level as intended.

```

Data set name = KARIM.TEST.DEFTABLE.VRS.TEST2
Volume        = B02663 Physical file sequence number = 1
Owner         = KARIM  Data set sequence = 1
Create date   = 09/22/2021 Create time = 14:45:55
Expiration date      = 09/25/2021
Expir. time = 14:45:55
      set by      = DEFTABLE      Original expir.date =
LASTREF Extra Days  = 0           WHILECATALOG      = OFF
Management class   = MCVTS1      VRS management value = D99000
Storage group      = SGVTS1      VRS retention date   =
Storage class      = SC3490      VRS retained        = NO
Data class         = DC3490      Closed by Abend     = NO
                                           Deleted              = NO
VRSEL exclude     = NO           Catalog status      = YES

```

Figure B-6 CONTINUE used



Although the retainby = SET is listed under the set by = DEFTABLE, it was derived from the parameters in the EDGRMMxx parmlib member and is not specified in the Defaults Table that was coded for this use case. RETAINBY (SET) is a best practice to keep all volumes in a volume set that is held to the same expiration criteria.

Figure B-7 shows that CONTINUE RM(VRSEL) was assigned as intended.

```

Volume = B02663   VOL1 =           Rack =           Owner   = KARIM
Type = LOGICAL           Stacked count = 0           Jobname = KARIMOP
Worldwide ID =           WORM      = N
Creation: Date = 07/27/2016   Time = 14:32:24   System ID = ZS22
Assign:   Date = 09/22/2021   Time = 14:45:55   System ID = ZS24
                                           User ID  = KARIM
Expiration date = 09/25/2021           Original =
Expiration time = 14:45:55   Datasets Kept By Catlg: 0
      set by = DEFTABLE
Retention date =           Set retained = NO
Retention method= VRSEL
      set by = DEFTABLE
      retain by =
Data set name = KARIM.TEST.DEFTABLE.VRS.TEST2

```

Figure B-7 VRSEL assigned

Figure B-8 shows the coding of the corrected Defaults Table that includes the CONTINUE operand.

```

DEFAULT DSN(KARIM.TEST.DEFTABLE.**.G%%V00) -
  WHILECATALOG(ON) RM(EXPDT) RETPD(3)
DEFAULT DSN(KARIM.TEST.DEFTABLE.***) -
  WHILECATALOG(UNTILEXPIRED) RM(EXPDT) RETPD(3) CONTINUE
DEFAULT DSN(KARIM.TEST.DEFTABLE.VRS.***) -
  RM(VRSEL) VRSVALUE(D99000)

```

Figure B-8 Corrected DEFTABLE

The display that is shown in Figure B-9 of the corrected Defaults Table indicates CONTINUE: Y.

```
F DFRMM,CMD=LC DEFTABLE
DEFTABLE is loaded from PARMLIB member EDGDEFT2, number of entries: 3
DEFTABLE content:
-----
DSNAME mask: KARIM.TEST.DEFTABLE.**.G%%V00                                CONTINUE:
JobName mask: *                    PGMNAME: *                    KeyDate: *
RETPD: 3      RO: N  EDM:  POOL:                    RM: EXPDT
(VRSEL) VRSvalue:                    VX:
(EXPDT) LastRef:                    WHILECAT: ON  RetainBy:
-----
DSNAME mask: KARIM.TEST.DEFTABLE.**                                CONTINUE: Y
JobName mask: *                    PGMNAME: *                    KeyDate: *
RETPD: 3      RO: N  EDM:  POOL:                    RM: EXPDT
(VRSEL) VRSvalue:                    VX:
(EXPDT) LastRef:                    WHILECAT: UX  RetainBy:
-----
DSNAME mask: KARIM.TEST.DEFTABLE.VRS.**                                CONTINUE:
JobName mask: *                    PGMNAME: *                    KeyDate: *
RETPD:      RO:  EDM:  POOL:                    RM: VRSEL
(VRSEL) VRSvalue: D99000  VX:
(EXPDT) LastRef:                    WHILECAT:      RetainBy:
-----
```

Figure B-9 Display of corrected DEFTABLE

This test demonstrates that the way the Default Table works for matching is distinctly different from the way DFRMM applies VRS rules. In the VRSEL environment, the fully qualified dsname mask applies rather than the lesser qualified mask.

During EDGHSKP VRSEL processing, all of the VRS records that are defined in the RMM CDS are read into storage along with all data set records. SORT is then called to match the data set records to a VRS that starts with the OPEN and ABEND VRS rules, followed by the dsname mask rules most qualified to least qualified and finally, the \*\* rule if one is defined.

The Defaults Table processing works strictly in the order of the entries that are coded into the table. In this case study, the first table entry that matches causes the search to complete and the attributes from that entry that is assigned to the data set, unless the CONTINUE parameter is present on the specific entry that was first matched.

If CONTINUE is present, this search continues until all matches with CONTINUE are resolved. All attributes for each match are applied with the last match that overrides any conflicting attributes. Therefore, care must be taken to code the most qualified masks first, unless the combination of multiple matches is the wanted result.

This case study, much like the case study in Appendix A, "Using EDGRMMxx global defaults for assignment of retention attributes case study" on page 103, demonstrates a method that does not require ACS routine changes or a EDGUX100/UXTABLE and can be used to move off of a UXTABLE. It can also be used to move off of VRSEL onto the more efficient EXPDT retention method.

In its simplest form, it is easy to implement by changing the Defaults Table dsname masks to more generic values. It handles data GDG data sets on tape much as data sets on SMS DASD is handled in that the data set is expired after the catalog entry is deleted.

For the non-GDG data set, the data is expired when its catalog entry is deleted or when it reaches its expiration date, similar to the DFHSM setting EXPIREDDATASETS(SCRATCH). This method partially achieves a level of device independence, so the user does not need to be concerned about whether the data set exists on tape or DASD.





# Using Management Class for assignment of expiration attributes case study

This third case study discusses the use of the MCATTR setting VRSELXDI, which is useful in an environment that uses VRSEL and EXPDT retention methods. This setting can support a migration to EXPDT retention method or the permanent use of both retention methods in tandem for different tape workloads.

It can be used in a VRSEL environment, but most of the attributes in the Management Class (MC) apply only to the EXPDT retention method. You can also run with VRSELXDI in an EXPDT-only environment after the VRSEL data sets are migrated.

This chapter includes the following topics:

- ▶ “Introduction” on page 118
- ▶ “Summary” on page 122

# Introduction

In this case study, the EDGRMMxx parmlib member uses a default value of RM(VRSEL), but it is not used because MCATTR(VRSELXDI) is coded and the MC is used to assign Retention Method EXPDT.

The MC is also used to assign various attributes that are useful with the EXPDT retention method. No Defaults Table or UX100 is used for the three tests in this case study.

The results of running the RMM LC ALL command that confirms the use of MCATTR and the default of RM(VRSEL) are shown in Figure C-1.

```
Use of Management Class Attributes = VRSELXDI
Retention method: Default = VRSEL
```

Figure C-1 RMM LC ALL

In the first test in this case study, the EXPDT retention value WHILECATALOG ON (WC(ON)) is set.

In the second test, the EXPDT retention value of WHILECATALOG UNTILEXPIRED (WC(UX)) is set.

The first test in this case study is a simple case that uses values 10 and 10 for Date/Days and Days Non-usage, which are set on the first page of the MC (see Figure C-2).

```
Expiration Attributes
Expire after Days Non-usage . . : 10
Expire after Date/Days . . . . : 10
Retention Limit . . . . . : NOLIMIT
```

Figure C-2 MC definitions first page

On the last page of the MC class, RM(EXPDT) and WC(ON) are assigned, as shown in Figure C-3.

```
Tape Volume Attributes
Retention Method . . . . . : EXPDT
Volume Set Management Level . . :

Tape Data Set Attributes
Exclude from VRSEL . . . . . :
Retain While Cataloged . . . . : ON
```

Figure C-3 MC definitions last page

A simple IEBGENER job is used for data set allocation. The results are an expiration date set by OCE\_MC and a LASTREF value of 10, both are taken from the MC class along with the WHILECATALOG(ON) attribute.

This data set is retained until it is uncataloged and reaches its expiration date (KeptByCatlg). Any read/write of the data set extends the expiration date by 10 days.

The MC class did not specify any value for Volume Set Management Level, so a default from the EDGRMMxx parmlib member of RETAINBY(VOLUME) was assumed. It is a best practice to code RETAINBY(SET) in both MC class definitions and the EDGRMMxx parmlib defaults for the EXPDT retention method. These values are shown in Figure C-4.

```

Data set name = KARIM.MCATTR1E.TEST.WCON
Volume       = B03944 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 09/23/2021 Create time = 14:50:03
Expiration date = 10/03/2021(KeptByCatlg)
Expir. time = 16:00:00
      set by = OCE_MC Original expir.date = 10/03/2021
LASTREF Extra Days = 10 WHILECATALOG = ON
Management class = MCATTR1E VRS management value =
Storage group    = SGVTS1 VRS retention date =
Storage class    = SC3490 VRS retained = NO
Data class       = DC3490 Closed by Abend = NO
                  Deleted = NO
VRSEL exclude   = YES Catalog status = YES
Primary VRS details:
  Name =
  Job name = Type =
  Subchain NAME = Subchain start date =
Secondary VRS details:
  Value or class =
  Job name =
  Subchain NAME = Subchain start date =

```

Figure C-4 WC(ON) test LD display

The volume where the data set is stored takes the expiration date and attributes from the data set. WHILECATALOG(ON) sets the (KeptByCatlg) value next to the expiration date and the Datasets Kept By Catlg value.

After the catalog entry of the data set on this volume is deleted, the volume is expired during EXPROC if the expiration date was in the past. Otherwise, the volume is retained until the expiration date is reached.

The RMM LV volser ALL output for this volume is shown in (Figure C-5).

```

Volume = B03944    VOL1 =          Rack =          Owner = KARIM
Type = LOGICAL          Stacked count = 0          Jobname = KARIMOP
Worldwide ID =          WORM = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign:   Date = 09/23/2021 Time = 14:50:03 System ID = ZS24
        PAGE          3
User ID = KARIM
Expiration date = 10/03/2021(KeptByCatlg) Original =
Expiration time = 16:00:00 Datasets Kept By Catlg: 1 on Volume
        set by = OCE_MC
Retention date =          Set retained = NO
Retention method= EXPDT
        set by = MC_ATTR
        retain by = VOLUME
Data set name = KARIM.MCATTR1E.TEST.WCON

```

Figure C-5 WC(ON) test LV display

The MC for the second test in this case study is shown in Figure C-6. It is a simple case that uses only a value of 10 for Expire After Days Non-usage with no value for Expire after Date.

```

Expiration Attributes
Expire after Days Non-usage . . . : 10
Expire after Date/Days . . . . . : NOLIMIT
Retention Limit . . . . . : NOLIMIT

```

Figure C-6 MC settings first page

It also assigns settings Retention Method: EXPDT and Retain While Cataloged: UNTILEXPIRED, as shown in Figure C-7.

```

Tape Volume Attributes
Retention Method . . . . . : EXPDT
Volume Set Management Level . . . :

Tape Data Set Attributes
Exclude from VRSEL . . . . . :
Retain While Cataloged . . . . . : UNTILEXPIRED

```

Figure C-7 MC settings last page

An IEBGENER job is used for data set allocation. The data set shows an expiration date that is set by LASTREF, which is taken from the MC class along with the WHILECATALOG(UNFILEXPIRED) attribute.

The data set is retained until its catalog entry is deleted or its expiration date is reached. When the data set reaches its expiration date, it is expired regardless of catalog status.

If the data set is uncataloged before reaching the expiration date, the expiration date is adjusted to the current date plus the value for CATLGDAYS in EDGRMMxx parmlib and then, expired on that date(OrUncatlg).



Any read/write of the data set extends the expiration date by 10 days. These results are shown in the RMM LISTDATASET output in Figure C-8.

```

Data set name = KARIM.MCATTRK1.TEST.WCUEX
Volume       = B00888 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 09/23/2021 Create time = 14:53:50
Expiration date = 10/03/2021(0rUncatlg)
Expir. time = 14:53:50
              set by = LASTREF Original expir.date = 10/03/2021
LASTREF Extra Days = 10          WHILECATALOG = UntilExpired
Management class = MCATTRK1    VRS management value =
Storage group    = SGVTS1      VRS retention date =
Storage class    = SC3490      VRS retained = NO
Data class       = DC3490      Closed by Abend = NO
                                   Deleted = NO
VRSEL exclude   = YES         Catalog status = YES
Primary VRS details:
  Name =
  Job name = Type =
  Subchain NAME = Subchain start date =
Secondary VRS details:
  Value or class =
  Job name =
  Subchain NAME = Subchain start date =

```

Figure C-8 WC(UX) test LD display

The volume takes the expiration date and attributes from the data set. WHILECATALOG(UNTILEXPIRED) sets the (0rUncatlg) value that is next to the expiration date. In this case, the Datasets Kept By Catlg value is 0.

After the catalog entry of the data set on this volume is deleted or it reaches the expiration date, the volume is expired during EXPROC. If the catalog was deleted before it reached the expiration date, the expiration date is reset to the current date plus the value of CATLGDAYS in the EDGRMMxx parmlib. If not specified, the default of CATLGDAYS is 2. Then, it is expired during EXPROC processing when that date is reached.

The RMM LISTVOLUME volser ALL output for this volume is shown in Figure C-9.

```
Volume = B00888   VOL1 =           Rack =           Owner   = KARIM
Type = LOGICAL           Stacked count = 0           Jobname = KARIMOP
Worldwide ID =                               WORM    = N
Creation:  Date = 06/29/2016  Time = 13:59:36  System ID = ZS22
Assign:    Date = 09/23/2021  Time = 14:53:50  System ID = ZS24
                                           User ID  = KARIM
Expiration date = 10/03/2021(OrUncatlg)  Original =
Expiration time = 14:53:50  Datasets Kept By Catlg: 0      on Volume
      set by = LASTREF
Retention date =           Set retained = NO
Retention method= EXPDT
      set by = MC_ATTR
      retain by = VOLUME
Data set name = KARIM.MCATTRK1.TEST.WCUEX
```

Figure C-9 RMM LISTVOLUME volser ALL output

## Summary

The two tests in this case study demonstrate a similar implementation that is discussed in Appendix A, “Using EDGRMMxx global defaults for assignment of retention attributes case study” on page 103, and Appendix B, “Using Defaults Table for expiration attributes assignment case study” on page 109.

However, this case study added the function of the ACS routines to assign specific MC expiration values for varying needs by workload.

The settings that are used in the MC classes handle data GDG data sets on tape and data sets on SMS DASD in the same way: After the catalog entry is deleted, the data set is expired.

For non-GDG data sets, the data is expired when its catalog entry is deleted or when it reaches its expiration date, similar to the DFHSM setting EXPIREDDATASETS(SCRATCH). This method partially achieves a level of device independence. In this context, the user does not need to know whether the data set exists on tape or DASD.



## **Using mixed retention methods VRSEL and EXPDT case study**

This case study is a mixed case that shows the two data sets that are written in the third case study (see Appendix C, “Using Management Class for assignment of expiration attributes case study” on page 117) plus a third data set.

This chapter includes the following topics:

- ▶ “Introduction” on page 124
- ▶ “First test” on page 125
- ▶ “Second test” on page 130
- ▶ “Summary” on page 137

# Introduction

In this case study, all three data sets are written to the same volume under the MCATTR(VRSELXDI) setting in the EDGRMMxx parmlib.

The first data set includes Management Class (MC) settings for RETENTIONMETHOND (RM) AND WHILECATALOG (WC). The settings are RM(EXPDT) and WC(ON). The second data set includes the MC settings RM(EXPDT) WC(UX), and the third data set includes an MC settings NOLIMIT NOLIMIT and a blank setting for RM.

With no value specified for RM in the MC that is used for the third data set, it defaults to the EDGRMMxx parmlib default setting of RM(VRSEL).

When the job that is shown in Example D-1 is run, it creates the three data sets on the same volume.

## *Example D-1 IEBGENER JCL*

---

```
//STEP001 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=SHR,DSN=KARIM.A.CNTL(JOBCARD)
//SYSUT2 DD DSN=KARIM.MCATTRK1.TEST.WCUEX1,
// DISP=(,CATLG),LABEL=(1,SL),
// UNIT=ZOSVTS1,DCB=*.SYSUT1,VOL=(,RETAIN,,10)
//*****
//STEP002 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=SHR,DSN=KARIM.A.CNTL(JOBCARD)
//SYSUT2 DD DSN=KARIM.MCATTR1E.TEST.WCON2,
// DISP=(,CATLG),
// DCB=*.SYSUT1,VOL=(,RETAIN,,10,REF=*.STEP001.SYSUT2),
// LABEL=(2,SL)
//*****
//STEP003 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//*****
//SYSUT1 DD DISP=SHR,DSN=KARIM.A.CNTL(JOBCARD)
//SYSUT2 DD DSN=KARIM.TEST.VTS1.VRSEL,
// DISP=(,CATLG),
// DCB=*.SYSUT1,VOL=(,RETAIN,,10,REF=*.STEP001.SYSUT2),
// LABEL=(3,SL)
//*****
```

---

## First test

Per the first test in the third case study, Appendix C, “Using Management Class for assignment of expiration attributes case study” on page 117, the first data set shows an expiration date set by OCE\_MC and features a LASTREF value of 10. These values were taken from the first page of the MC class.

Any read/write of the data set extends the expiration date by 10 days. See Figure D-1.

```
Expiration Attributes
  Expire after Days Non-usage . . . : 10
  Expire after Date/Days . . . . . : 10
  Retention Limit . . . . . : NOLIMIT
```

Figure D-1 MC class first page

Figure D-2 shows the last page of the MC class assigning RM(EXPDT) and the WHILECATALOG(ON) attribute.

```
Tape Volume Attributes
  Retention Method . . . . . : EXPDT
  Volume Set Management Level . . :

Tape Data Set Attributes
  Exclude from VRSEL . . . . . :
  Retain While Cataloged . . . . : ON
```

Figure D-2 MC class last page

Figure D-3 shows that this data set is retained until its catalog entry is deleted and that is reached its expiration *date* (KeptByCatlg).

```

Data set name = KARIM.MCATTR1E.TEST.WCON2
Volume       = B03571 Physical file sequence number = 3
Owner        = KARIM Data set sequence = 2
Create date  = 09/23/2021 Create time = 15:01:16
Expiration date = 10/03/2021(KeptByCatlg)
Expir. time = 16:00:00
              set by = OCE_MC Original expir.date = 10/03/2021
LASTREF Extra Days = 10 WHILECATALOG = ON
Management class = MCATTR1E VRS management value =
Storage group    = SGVTS1 VRS retention date =
Storage class    = SC3490 VRS retained = NO
Data class       = DC3490 Closed by Abend = NO
                  Deleted = NO
                  Catalog status = YES
VRSEL exclude   = YES
Primary VRS details:
  Name           =
  Job name       = Type =
  Subchain NAME = Subchain start date =
Secondary VRS details:
  Value or class =
  Job name       =
  Subchain NAME = Subchain start date =
  
```

Figure D-3 WC(ON) LD display

The next data set to be allocated in the IEBGENER job uses an MC class with Expire after Days Non-usage value of 10 and no Expire after Date/Days, as shown in Figure D-4.

```

Expiration Attributes

Expire after Days Non-usage . . : 10
Expire after Date/Days . . . . : NOLIMIT
Retention Limit . . . . . : NOLIMIT
  
```

Figure D-4 MC class first page

Figure D-5 shows the assignment of RM(EXPDT) and WC(UX), per the second test in Appendix C, “Using Management Class for assignment of expiration attributes case study” on page 117.

```

Tape Volume Attributes
Retention Method . . . . . : EXPDT
Volume Set Management Level . :

Tape Data Set Attributes
Exclude from VRSEL . . . . . :
Retain While Cataloged . . . . : UNTILEXPIRED
  
```

Figure D-5 MC class final page

This second data set shows the expiration date that is set by LASTREF that was taken from the MC class along with the WHILECATALOG(UNTILEXPIRED) attribute.

When the data set reaches its expiration date, it is expired regardless of catalog status.

If the data set is uncataloged before it reaches the expiration date, the expiration date is adjusted to the current date plus the value for CATLGDAYS in EDGRMMxx parmlib and then, it is expired on that *date* (OrUncatlg).

Any read/write of the data set extends the expiration date by 10 days, as shown in Figure D-6.

```

Data set name = KARIM.MCATTRK1.TEST.WCUEX1
Volume       = B03571 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 09/23/2021 Create time = 15:01:15
Expiration date = 10/03/2021(OrUncatlg)
Expir. time = 15:01:15
      set by      = LASTREF      Original expir.date = 10/03/2021
LASTREF Extra Days = 10        WHILECATALOG      = UntilExpired
Management class = MCATTRK1    VRS management value =
Storage group    = SGVTS1      VRS retention date   =
Storage class    = SC3490      VRS retained         = NO
Data class       = DC3490      Closed by Abend      = NO
                                      Deleted                = NO
                                      Catalog status         = YES
VRSEL exclude   = YES
Primary VRS details:
  Name          =
  Job name      =              Type                =
  Subchain NAME =              Subchain start date =
Secondary VRS details:
  Value or class =
  Job name      =
  Subchain NAME =              Subchain start date =
  
```

Figure D-6 WC(UX) LD display

The third data set that is written to this volume demonstrates the typically unintended interaction of the SMS MC attributes and the two retention methods.

The first page of the MC class for this data set shows NOLIMIT for the three attributes. See Figure D-7.

```

Expiration Attributes

Expire after Days Non-usage . . : NOLIMIT
Expire after Date/Days . . . . : NOLIMIT
Retention Limit . . . . . : NOLIMIT
  
```

Figure D-7 MC class first page

The final page of the MC class does not specify any expiration values, as shown in Figure D-8.

```

Tape Volume Attributes
  Retention Method . . . . . :
  Volume Set Management Level . . :

Tape Data Set Attributes
  Exclude from VRSEL . . . . . :
  Retain While Cataloged . . . . :
  
```

Figure D-8 MC class last page

This data set is created under RM(VRSEL) because the MC settings do not include a value for the retention method and the EDGRMMxx parmlib does not specify RETENTIONMETHOD; therefore, it defaults to RM(VRSEL).

Because it is created on a volume that is managed by EXPDT, it is assigned the VRSELEXCLUDE (VRSEL exclude = YES) attribute, as do all data sets that are managed on an EXPDT volume. See Figure D-9.

```

Data set name = KARIM.TEST.VTS1.VRSEL
Volume       = B03571 Physical file sequence number = 3
Owner        = KARIM Data set sequence = 3
Create date  = 09/23/2021 Create time = 15:01:16
Expiration date = 12/31/1999 Expir. time = 15:01:16
  set by      = OCE_MC Original expir.date = 12/31/1999
LASTREF Extra Days = 0 WHILECATALOG = OFF
Management class = MCVTS1 VRS management value =
Storage group    = SGVTS1 VRS retention date =
Storage class    = SC3490 VRS retained = NO
Data class       = DC3490 Closed by Abend = NO
                Deleted = NO
VRSEL exclude    = YES Catalog status = YES
Primary VRS details:
  Name           =
  Job name       = Type =
  Subchain NAME = Subchain start date =
Secondary VRS details:
  Value or class =
  Job name       =
  Subchain NAME = Subchain start date =
  
```

Figure D-9 Mixed Retention LD display

Therefore, running VRSEL does not affect this data set or change any of its values. However, it does not take the expiration value from the EDGRMMxx parmlib defaults, and it does not set a retention date from the VRS rule that is applied later during VRSEL. Instead, it takes the expiration value from the MC settings of NOLIMIT NOLIMIT. This data set does not expire until manually released.



With these three data sets on the volume, the volume attributes are the ones that retain the volume the longest. The first data set that is written to the volume determines the RM that the volume is managed under, which is EXPDT in this case.

Although a data set is allocated with file sequence 3 on this volume that specified RM(VRSEL), this attribute is not changed. The first data set also includes the WHILECATALOG attribute that keeps the data set longer, WC(ON).

The second data set does not include any attributes that retain the data set the longest because WHILECATALOG(UNTILEXPIRED) is a shorter duration than WHILECATALOG(ON). WC(ON) is considered a longer duration because WC(UX) resets the expiration date to the current date plus CATLGDAYS but WC(ON) does not.

The third data set includes an expiration date that retains the data set the longest (12/31/1999). However, when you combine these attributes, you get an expiration date of 12/31/1999 with the addition of the (KeptByCatlg) qualifier.

In this case, the (KeptByCatlg) qualifier does not affect retention because 12/31/1999 is permanent retention. See Figure D-10.

```
Volume = B03571  VOL1 =          Rack =          Owner = KARIM
Type = LOGICAL          Stacked count = 0          Jobname = KARIMOP
Worldwide ID =          WORM = N
Creation:  Date = 07/27/2016  Time = 14:32:24  System ID = ZS22
Assign:    Date = 09/23/2021  Time = 15:01:15  System ID = ZS24
                                           User ID = KARIM
Expiration date = 12/31/1999(KeptByCatlg) Original =
Expiration time =          Datasets Kept By Catlg: 1      on Volume
      set by = OCE_MC
Retention date =          Set retained = NO
Retention method= EXPDT
      set by = MC_ATTR
      retain by = VOLUME
Data set name = KARIM.MCATTRK1.TEST.WCUEX1
```

Figure D-10 RMM LV volser ALL output

The result of the combination of these data set attributes to the volume level is that when the data sets on the volume are uncataloged, the expiration date is not adjusted and the volume is still held permanently by expiration date 12/31/1999. It does not expire without manual release. See Figure D-11.

```

Volume = B03571   VOL1 =           Rack =           Owner = KARIM
Type = LOGICAL   Stacked count = 0           Jobname = KARIMOP
Worldwide ID =           WORM = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign:   Date = 09/23/2021 Time = 15:01:15 System ID = ZS24
                                           User ID = KARIM
Expiration date = 12/31/1999           Original =
Expiration time =           Datasets Kept By Catlg: 0   on Volume
      set by = OCE_MC
Retention date =           Set retained = NO
Retention method= EXPDT
      set by = MC_ATTR
      retain by = VOLUME
Data set name = KARIM.MCATTRK1.TEST.WCUEX1
Volume status:  EDM = N   Hold = N   File 1 Data set seq = 1
Status = MASTER   Availability =           Label = SL

```

Figure D-11 LV output after delete of catalog entries

## Second test

In the second test in this case study, another mixed case shows the files that are written in the first two tests along with a third file, which is similar to the first test.

However, the files are written to the volume in a different order so that the first file selects RM(VRSEL) for this volume. All three data sets are written to the same volume under the MCATTR(VRSELXDI) setting in EDGRMMxx parmlib. If no value is specified for RETENTIONMETHOD, then it defaults to VRSEL.

The first data set includes MC settings NOLIMIT NOLIMIT and a blank setting for RM. The second data set has MC settings RM(EXPDT) WC(ON), and the third data set has MC settings RM(EXPDT) WC(UX).

Because the first file that is written to this volume is created by using an MC with no RM specified, it takes the EDGRMMxx parmlib default of VRSEL. After the RM is set for the volume, all of the remaining data sets that were created on the volume are also managed by using RM(VRSEL), even though their MC attributes specify RM(EXPDT).

As a result, these data sets are all retained by using a VRS rule that covers the data set naming pattern. The attributes in the MC for data sets two and three are ignored because the volume is VRSEL-managed rather than EXPDT-managed. The second and third data sets in this case do not contribute any expiration dates or attributes to the volume.

The job that is shown in Example D-2 was run to create the three data sets on the same volume.

*Example D-2 IEBGENER JCL*

---

```
//STEP001 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=SHR,DSN=KARIM.A.CNTL(JOBCARD)
//SYSUT2 DD DSN=KARIM.TEST.VTS1.VRSEL4,
// DISP=(,CATLG),LABEL=(1,SL),
// UNIT=ZOSVTS1,DCB=*.SYSUT1,VOL=(,RETAIN,,10)
//*****
//STEP002 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=SHR,DSN=KARIM.A.CNTL(JOBCARD)
//SYSUT2 DD DSN=KARIM.MCATTR1E.TEST.WCON4,
// DISP=(,CATLG),
// DCB=*.SYSUT1,VOL=(,RETAIN,,10,REF=*.STEP001.SYSUT2),
// LABEL=(2,SL)
//*****
//STEP003 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=SHR,DSN=KARIM.A.CNTL(JOBCARD)
//SYSUT2 DD DSN=KARIM.MCATTRK1.TEST.WCUEX4,
// DISP=(,CATLG),
// DCB=*.SYSUT1,VOL=(,RETAIN,,10,REF=*.STEP001.SYSUT2),
// LABEL=(3,SL)
//*****
```

---

The first data set on the volume sets the retention method to VRSEL with an expiration date that is taken from the default value in EDGRMMxx parmlib option RETPD(5). Now, the RM for the volume is determined to be VRSEL because the MC settings did not specify a value for RM and the EDGRMMxx parmlib RM value is not set and defaults to VRSEL.

The Retention Date is not specified because the EDGHSKP function VRSEL is not yet run. See Figure D-12 on page 132.

```

Data set name = KARIM.TEST.VTS1.VRSEL4
Volume       = B00690 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 09/28/2021 Create time = 14:32:35
Expiration date      = 10/03/2021
Expir. time = 14:32:35
      set by      = OCE_DEF      Original expir.date =
LASTREF Extra Days = 0          WHILECATALOG      = OFF
Management class  = MCVTS1     VRS management value =
Storage group     = SGVTS1     VRS retention date   =
Storage class     = SC3490     VRS retained         = NO
Data class        = DC3490     Closed by Abend      = NO
                                   Deleted           = NO
                                   Catalog status    = YES
VRSEL exclude    = NO
Primary VRS details:
  Name           =
  Job name       =
  Subchain NAME  =
  Type           =
  Subchain start date =
Secondary VRS details:
  Value or class =
  Job name       =
  Subchain NAME  =
  Subchain start date =

```

Figure D-12 VRSEL LD display

Although they have MC setting RM(EXPDT, the VRSEXCLUDE = YES setting is no applied to the second and third data sets on the volume. They also take the default expiration date from EDGRMMxx parmlib RETPD(5).

The data sets do not have any WHILECATALOG attribute set, even though they are specified in the MC settings. This issue occurs because they are not managed with RM(EXPDT). Therefore, those settings do not apply. See Figure D-13.

```

Data set name = KARIM.MCATTR1E.TEST.WCON4
Volume       = B00690 Physical file sequence number = 2
Owner        = KARIM Data set sequence = 2
Create date  = 09/28/2021 Create time = 14:32:36
Expiration date      = 10/03/2021
Expir. time = 14:32:36
      set by      = OCE_DEF      Original expir.date =
LASTREF Extra Days = 0          WHILECATALOG      = OFF
Management class  = MCATTR1E   VRS management value =
Storage group     = SGVTS1      VRS retention date   =
Storage class     = SC3490      VRS retained         = NO
Data class        = DC3490      Closed by Abend      = NO
                                   Deleted           = NO
                                   Catalog status    = YES
VRSEL exclude     = NO
Primary VRS details:
  Name            =
  Job name        =
  Subchain NAME   =
  Type            =
  Subchain start date =
Secondary VRS details:
  Value or class =
  Job name        =
  Subchain NAME   =
  Subchain start date =
  
```

Figure D-13 WC(ON) LD display

Figure D-14 shows the third data set on the volume.

```

Data set name = KARIM.MCATTRK1.TEST.WCUEX4
Volume       = B00690 Physical file sequence number = 3
Owner        = KARIM Data set sequence = 3
Create date  = 09/28/2021 Create time = 14:32:36
Expiration date = 10/03/2021
Expir. time = 14:32:36
      set by      = OCE_DEF      Original expir.date =
LASTREF Extra Days = 0          WHILECATALOG      = OFF
Management class = MCATTRK1    VRS management value =
Storage group    = SGVTS1      VRS retention date  =
Storage class    = SC3490      VRS retained        = NO
Data class       = DC3490      Closed by Abend     = NO
                                   Deleted              = NO
VRSEL exclude    = NO          Catalog status      = YES
Primary VRS details:
  Name           =
  Job name       =              Type                   =
  Subchain NAME =              Subchain start date    =
Secondary VRS details:
  Value or class =
  Job name       =
  Subchain NAME =              Subchain start date    =

```

Figure D-14 WC(UX) LD display

The state of the volume now is that it is RM(VRSEL), but not yet VRS-retained because the EDGHSKP function VRSEL is not yet run. It includes an expiration date that is taken from the data sets on the volume, all of which used RETPD(5) from EDGRMMxx parm lib.

Also, no dependency exists on the data sets that are cataloged now. If no VRS ever matched these data sets, the volume is retained until it reached its expiration date (see Figure D-15).

```

Volume = B00690 VOL1 = Rack = Owner = KARIM
Type = LOGICAL Stacked count = 0 Jobname = KARIMT4
Worldwide ID = WORM = N
Creation: Date = 06/29/2016 Time = 13:59:36 System ID = ZS22
Assign: Date = 09/28/2021 Time = 14:32:35 System ID = ZS24
User ID = KARIM
Expiration date = 10/03/2021 Original =
Expiration time = 14:32:36 Datasets Kept By Catlg: 0
set by = OCE_DEF
Retention date = Set retained = NO
Retention method= VRSEL
set by = OCE_DEF
retain by =
Data set name = KARIM.TEST.VTS1.VRSEL4
Volume status: EDM = N Hold = N File 1 Data set seq = 1
Status = MASTER Availability = Label = SL

```

Figure D-15 RMM LV output

After the first run of EDGHSKP VRSEL, a VRS is assigned that retains all three of these data sets. It also assigns a retention date to the volume of WHILECATALOG that comes from the VRS definition. See Figure D-16.

```

Volume = B00690  VOL1 =          Rack =          Owner   = KARIM
  Type = LOGICAL          Stacked count = 0          Jobname = KARIMT4
  Worldwide ID =          WORM      = N
Creation: Date = 06/29/2016  Time = 13:59:36  System ID = ZS22
Assign:   Date = 09/28/2021  Time = 14:32:35  System ID = ZS24
                                         User ID  = KARIM
Expiration date = 10/03/2021          Original =
Expiration time = 14:32:36  Datasets Kept By Catlg: 0
      set by = OCE_DEF
Retention date = WHILECATLG Set retained = NO
Retention method= VRSEL
      set by = OCE_DEF
      retain by =
Data set name = KARIM.TEST.VTS1.VRSEL4
Volume status:  EDM = N  Hold = N  File 1 Data set seq = 1
Status = MASTER  Availability = Vital Record  Label = SL

```

Figure D-16 RMM LV output after VRSEL

All three data sets on the volume now match the KARIM.\*\* VRS rule and all are retained by using this VRS, despite two of them being allocated by using MC with setting RM(EXPDT), as shown in Figure D-17.

```

Data set mask = KARIM.**          Type = DSN
Job name mask =          Retain until expired = NO
Count         = 99999 CYCLES          Retain while cataloged = YES
Delay         = 0      Days in the HOME location
Store number = 99999 CYCLES in the HOME location
Priority      = 0          Release Options:
                                         Expiry date ignore = NO
                                         Scratch immediate = NO

```

Figure D-17 KARIM VRS

The PRIMARY VRS details are in the LD display for the first data set. See Figure D-18.

```

Data set name = KARIM.TEST.VTS1.VRSEL4
Volume       = B00690 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 09/28/2021 Create time = 14:32:35
Expiration date = 10/03/2021
Expir. time = 14:32:35
      set by      = OCE_DEF      Original expir.date =
LASTREF Extra Days = 0          WHILECATALOG      = OFF
Management class = MCVTS1      VRS management value =
Storage group    = SGVTS1      VRS retention date  = WHILECATLG
Storage class    = SC3490      VRS retained        = YES
Data class       = DC3490      Closed by Abend     = NO
                                   Deleted              = NO
                                   Catalog status        = YES
VRSEL exclude   = NO
Primary VRS details:
  Name          = KARIM.**
  Job name      =              Type                  = DATASET
  Subchain NAME =              Subchain start date  =
Secondary VRS details:
  Value or class =
  Job name      =
  Subchain NAME =              Subchain start date  =

```

Figure D-18 VRS data set1

The PRIMARY VRS details are in the LD display for the second data set. See Figure D-19.

```

Data set name = KARIM.MCATTR1E.TEST.WCON4
Volume       = B00690 Physical file sequence number = 2
Owner        = KARIM Data set sequence = 2
Create date  = 09/28/2021 Create time = 14:32:36
Expiration date = 10/03/2021
Expir. time = 14:32:36
      set by      = OCE_DEF      Original expir.date =
LASTREF Extra Days = 0          WHILECATALOG      = OFF
Management class = MCATTR1E    VRS management value =
Storage group    = SGVTS1      VRS retention date  = WHILECATLG
Storage class    = SC3490      VRS retained        = YES
Data class       = DC3490      Closed by Abend     = NO
                                   Deleted              = NO
                                   Catalog status        = YES
VRSEL exclude   = NO
Primary VRS details:
  Name          = KARIM.**
  Job name      =              Type                  = DATASET
  Subchain NAME =              Subchain start date  =
Secondary VRS details:
  Value or class =
  Job name      =
  Subchain NAME =              Subchain start date  =

```

Figure D-19 VRS data set2



The PRIMARY VRS details are in the LD display for the first data set. See Figure D-20.

```
Data set name = KARIM.MCATTRK1.TEST.WCUEX4
Volume       = B00690 Physical file sequence number = 3
Owner        = KARIM Data set sequence = 3
Create date  = 09/28/2021 Create time = 14:32:36
Expiration date = 10/03/2021
Expir. time = 14:32:36
      set by = OCE_DEF      Original expir.date =
LASTREF Extra Days = 0      WHILECATALOG = OFF
Management class = MCATTRK1 VRS management value =
Storage group   = SGVTS1    VRS retention date = WHILECATLG
Storage class   = SC3490    VRS retained = YES
Data class      = DC3490    Closed by Abend = NO
                        Deleted = NO
VRSEL exclude   = NO       Catalog status = YES
Primary VRS details:
  Name          = KARIM.**
  Job name      =          Type = DATASET
  Subchain NAME =          Subchain start date =
Secondary VRS details:
  Value or class =
  Job name      =
  Subchain NAME =          Subchain start date =
```

Figure D-20 VRS data set3

The matching VRS specifies Retain while cataloged = YES and Expiry date ignore = NO. It is important not to confuse this attribute with the attribute WHILECATALOG(ON), which is the functional equivalent in RM(EXPDT).

Similarly for both of these attributes, when the catalog entry is deleted the expiration of the volume happens when the expiration date is reached. After the data sets on this volume are uncataloged, the data sets and volume leave VRS control and are eligible to expire on the volume expiration date. The volume expiration date is defined by the release options that are specified in the VRS. On 10/03/2021, this volume is released during EDGHSKP EXPROC processing.

## Summary

As you can see from reviewing the tests in this case study, combining VRSEL and EXPDT retention methods can be confusing, especially regarding file stacking. It is simplest to use one of the other of these two retention methods.

The addition of the SMS MC class attributes makes it relatively simple to assign new data set allocations to the EXPDT retention method and select expiration criteria in the Management Class.

The ACS routines can be customized to direct different workloads by dsname, jobname, or special JCL dates to different management classes with different expiration values.





# Using Management Class and Defaults Table together case study

In this case study, a DEFAULT TABLE is used to assign expiration attributes, and ACS routines are used to assign management class tape expiration values. The study shows which assignment receives precedence.

For this case study, two management classes are defined that represent a customer environment as they pertain to KEYDATE 99000, which usually is defined in a tape management system as being under WHILECATALOG control.

The second Management Class (MC) is altered to specify EXPDT retention and the first specifies VRSEL retention, but neither has any expiration date specified in the MC.

When MCATTR(VRSELXDI) or MCATTR(ALL) are put into effect and the values in preexisting MC classes are included, problematic results can occur.

The previous three case studies are considered best practices. This case study cautions against certain practices.

This chapter includes the following topics:

- ▶ “First Test” on page 140
- ▶ “Second test” on page 144

# First Test

Adding a Defaults Table in addition to the SMS MC class for expiration attributes can cause ambiguity about how the assignment of specific attributes is determined.

It is important when you migrate to MCATTR support or the Defaults Table to be aware of such possible ambiguities. It is also important that tape allocations through the ACS routines are correctly analyzed and changed to work with the MC tape expiration settings. In the case of VRSELXDI where RM is VRSEL, it does not matter because these values are not considered.

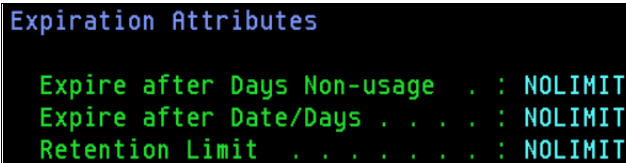
However, it does matter when an MC is changed to specify that RM is to be EXPDT or the default in EDGRMMxx parmlib is changed to RM(EXPDT).

An IEBGENER job that is used to allocate a data set is shown in Example E-1.

### Example E-1 IEBGENER JCL

```
//STEP001 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=SHR,DSN=KARIM.A.CNTL(JOBCARD)
//SYSUT2 DD DSN=KARIM.TEST.DEFTABL.MGMTVALU.TEST1,
// DISP=(,CATLG),LABEL=(1,SL),EXPDT=99000,
// UNIT=ZOSVTS1,DCB=*.SYSUT1,VOL=(,RETAIN,,10)
```

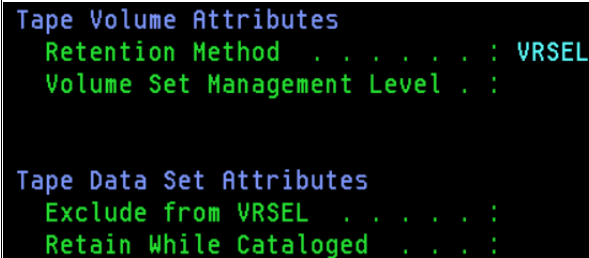
The SMS MC class definitions that are used are D99000 and M99000. The D99000 MC first page assignments are NOLIMIT, as shown in Figure E-1.



```
Expiration Attributes
Expire after Days Non-usage . . . : NOLIMIT
Expire after Date/Days . . . . . : NOLIMIT
Retention Limit . . . . . : NOLIMIT
```

Figure E-1 D99000 MC first page

The D99000 final page assignment of RM(EXPDT) is shown in Figure E-2.



```
Tape Volume Attributes
Retention Method . . . . . : VRSEL
Volume Set Management Level . . . :

Tape Data Set Attributes
Exclude from VRSEL . . . . . :
Retain While Cataloged . . . . . :
```

Figure E-2 D99000 final page

M99000 is also defined with NOLIMIT, as shown in Figure E-3.

```
Expiration Attributes
  Expire after Days Non-usage . . : NOLIMIT
  Expire after Date/Days . . . . : NOLIMIT
  Retention Limit . . . . . : NOLIMIT
```

Figure E-3 M99000 MC first page

Figure E-4 shows that instead of VRSEL, EXPDT and WC(ON) are assigned in the final page of the MC class definitions.

```
Tape Volume Attributes
  Retention Method . . . . . : EXPDT
  Volume Set Management Level . . :

Tape Data Set Attributes
  Exclude from VRSEL . . . . . :
  Retain While Cataloged . . . . : ON
```

Figure E-4 M99000 MC final page

The RMM LISTCONTROL ALL output is shown in Figure E-5.

```
Exit status:
  EDG_EXIT100 = NONE

Retention period: Default = 5          Maximum = NOLIMIT
                  Catalog = 12       hours
Use of Management Class Attributes = VRSELXDI
Retention method: Default = VRSEL

RM(VRSEL) defaults:
  Retain by      = VOLUME
  Move By       = VOLUME
  VRS selection  = NEW
  VRS change     = INFO
  VRSMIN action  = FAIL
  VRSMIN count   = 1
  VRS job name   = 2
  GDG duplicate  = BUMP
  GDG cycle by   = GENERATION

RM(EXPDT) defaults:
  Retain by      = VOLUME
  LASTREF extra days = 0
  Catalog Days   = 2
  GDG : WHILECATALOG = OFF
  RETPD         = 5
  Non-GDG: WHILECATALOG = OFF
  RETPD         = 5
```

Figure E-5 RMM LC ALL

Figure E-6 shows the Default Tables values being used.

```
F DFRMM,CMD=LC DEFTABLE
DEFTABLE is loaded from PARMLIB member EDGDEFT4, number of entries: 1
DEFTABLE content:
-----
DSNAME mask: *.TEST.DEFTABL.MGMTVALU.**                                CONTINUE:
JobName mask: *                PGMNAME: *                KeyDate: 99000
RETPD:      RO:      EDM:      POOL:      RM:
(VRSEL) VRSvalue: M99000      VX:
(EXPDT) LastRef:              WHILECAT:      RetainBy:
-----
```

Figure E-6 EDGDEFxx values

When this combination of MC class settings is used and a VRSvalue of M99000 is assigned in the Defaults Table, the assignment of the RM(EXPDT) attribute is not applied because the ACS routines have assigned D99000 Management Class.

A VRS Management value is different than an MC and does not take precedence over the ACS MC class assignment as shown in the data set display. See Figure E-7.

```
Data set name = KARIM.TEST.DEFTABL.MGMTVALU.TEST1
Volume        = B04623 Physical file sequence number = 1
Owner         = KARIM Data set sequence = 1
Create date   = 10/01/2021 Create time = 15:06:38
Expiration date = 00/00/1999 Expir. time =
    set by      = OCE_JFCB Original expir.date = 00/00/1999
LASTREF Extra Days = 0      WHILECATALOG      = OFF
DD name        = SYSUT2 Last DD name         = SYSUT2
Device number  = 2623 Last Device number    = 2623
Management class = D99000 VRS management value = M99000
Storage group  = SGVTS1 VRS retention date   =
Storage class  = SC3490 VRS retained        = NO
Data class     = DC3490 Closed by Abend     = NO
                Deleted                      = NO
VRSEL exclude  = NO      Catalog status     = YES
```

Figure E-7 LD output

The volume is also VRSEL-managed, as shown in Figure E-8.

```

Volume = B04623   VOL1 =           Rack =           Owner = KARIM
Type = LOGICAL           Stacked count = 0           Jobname = KARIMT4
Worldwide ID =           WORM = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign:   Date = 10/01/2021 Time = 15:06:38 System ID = ZS24
User ID = KARIM
Expiration date = 00/00/1999           Original = 00/00/1999
Expiration time =           Datasets Kept By Catlg: 0
set by = OCE_JFCB
Retention date =           Set retained = NO
Retention method= VRSEL
set by = MC_ATTR
retain by =
Data set name = KARIM.TEST.DEFTABL.MGMTVALU.TEST1

```

Figure E-8 LV output

Thus, a Defaults Table-specified VRSvalue does *not* contribute MC tape expiration settings. Also, the MC tape expiration settings for DATE/DAYS and Days Non-Usage is not applied in a VRSELXDI environment when VRSEL is the RM that is chosen for the volume or this volume is permanently retained(NOLIMIT NOLIMIT).

After EDGHSKP VRSEL is run, the volume is assigned the WHILECATALOG attribute as defined in the VRS for D99000, which came from the ACS routine assignment of the MC. Two VRSes are applied to this volume, but none are from the VRSvalue from the Defaults Table.

The VRS Type displays as DSN/MC to reflect the first VRS is from the DSN mask of KARIM.\*\* and the second VRS(D99000) is from the MC class assignment rather than the VRSVALUE of M99000 that was specified in the Defaults Table. See Figure E-9.

```

Data set name = KARIM.TEST.DEFTABL.MGMTVALU.TEST1
Volume       = B04623 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 10/01/2021 Create time = 15:06:38
Expiration date = 00/00/1999           Expir. time =
set by       = OCE_JFCB           Original expir.date = 00/00/1999
LASTREF Extra Days = 0           WHILECATALOG = OFF
Management class = D99000           VRS management value = M99000
Storage group   = SGVTS1           VRS retention date = WHILECATLG
Storage class   = SC3490           VRS retained = YES
Data class      = DC3490           Closed by Abend = NO
Deleted = NO
VRSEL exclude  = NO           Catalog status = YES
Primary VRS details:
Name = KARIM.**
Job name =           Type = DSN/MC
Subchain NAME =           Subchain start date =
Secondary VRS details:
Value or class = D99000
Job name =
Subchain NAME =           Subchain start date =

```

Figure E-9 LD output

Figure E-10 shows the result in the volume being retained as Availability = Vital Record.

```
Volume = B04623  VOL1 =          Rack =          Owner = KARIM
Type = LOGICAL      Stacked count = 0          Jobname = KARIMT4
Worldwide ID =          WORM = N
Creation: Date = 07/27/2016  Time = 14:32:24  System ID = ZS22
Assign:   Date = 10/01/2021  Time = 15:06:38  System ID = ZS24
                                           User ID = KARIM
Expiration date = 00/00/1999          Original = 00/00/1999
Expiration time =          Datasets Kept By Catlg: 0
      set by = OCE_JFCB
Retention date = WHILECATLG Set retained = NO
Retention method= VRSEL
      set by = MC_ATTR
      retain by =
Data set name = KARIM.TEST.DEFTABL.MGMTVALU.TEST1
Status = MASTER      Availability = Vital Record      Label = SL
```

Figure E-10 LV output

## Second test

In the second test in this case study, the Defaults Table is changed to specific RM(EXPDT) to test whether the assignment of this attribute determines the RM for the volume over the assignment of this attribute from the MC. See Figure E-11.

```
DEFTABLE is loaded from PARMLIB member EDGDEFT4, number of entries: 1
DEFTABLE content:
-----
DSNAME mask: *.TEST.DEFTABL.MGMTVALU.**          CONTINUE:
JobName mask: *          PGMNAME: *          KeyDate: 99000
RETPD:          RO:          EDM:          POOL:          RM: EXPDT
(VRSEL) VRSvalue:          VX:
(EXPDT) LastRef:          WHILECAT:          RetainBy:
-----
```

Figure E-11 EDGDEFxx contents



The same IEBGENER JCL was run to create the data set in the ListDataset display, as shown in Figure E-12.

```

Data set name = KARIM.TEST.DEFTABL.MGMTVALU.TEST2
Volume       = B02294 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 10/01/2021 Create time = 15:41:31
Expiration date = 00/00/1999 Expir. time =
    set by    = OCE_JFCB Original expir.date = 00/00/1999
LASTREF Extra Days = 0 WHILECATALOG = OFF
Device number = 262E Last Device number = 262E
Management class = D99000 VRS management value =
Storage group = SGVTS1 VRS retention date =
Storage class = SC3490 VRS retained = NO
Data class = DC3490 Closed by Abend = NO
Deleted = NO
VRSEL exclude = NO Catalog status = YES

```

Figure E-12 LD output

Figure E-13 shows the ListVolume output.

```

Volume = B02294 VOL1 = Rack = Owner = KARIM
Type = LOGICAL Stacked count = 0 Jobname = KARIMT4B
Worldwide ID = WORM = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign: Date = 10/01/2021 Time = 15:41:31 System ID = ZS24
User ID = KARIM
Expiration date = 00/00/1999 Original = 00/00/1999
Expiration time = Datasets Kept By Catlg: 0
    set by = OCE_JFCB
Retention date = Set retained = NO
Retention method= VRSEL
    set by = MC_ATTR
    retain by =
Data set name = KARIM.TEST.DEFTABL.MGMTVALU.TEST2

```

Figure E-13 LV output

The conclusion is that although the Defaults Table included RM(EXPDT), the MC provided attribute of RM(VRSEL) is a higher priority and determines the RM setting for the data set and volume.

The set by value for Retention method specifies MC\_ATTR because it came from the MC and the set by for the expiration date reflects OCE\_JFCB because EXPDT=199900 was coded on the JCL.





## **Using MCATTR(ALL) instead of MCATTR(VRSELXDI) case study**

This chapter includes the case study of the MCATTR value of ALL.

The difference between VRSELXDI and ALL is that the tape expiration values in the management class under VRSELXDI are applied only to EXPDT-managed volumes. In ALL, they are applied to all volumes, regardless of the retention method that is used.

## Introduction

In this case study, a Management Class (MC) is used that defaults to VRSEL retention method management. The case study shows two tests. The first test passes an expiration value in the JCL by way of the RETPD parameter. The second test omits EXPDT or RETPD from the JCL. Defaults Tables are not used in this case study.

Figure F-1 shows the RMM LISTCONTROL ALL output.

```
Exit status:
  EDG_EXIT100 = NONE

Retention period: Default = 5          Maximum = NOLIMIT
                  Catalog = 12       hours
Use of Management Class Attributes = ALL
Retention method: Default = VRSEL

RM(VRSEL) defaults:                RM(EXPDT) defaults:
  Retain by          = VOLUME        Retain by          = VOLUME
  Move By           = VOLUME
  VRS selection     = NEW            LASTREF extra days = 0
  VRS change        = INFO          Catalog Days       = 2 289
  VRSMIN action     = FAIL          GDG      : WHILECATALOG = OFF
  VRSMIN count      = 1             RETPD             = 5
  VRS job name      = 2             Non-GDG: WHILECATALOG = OFF
  GDG duplicate     = BUMP          RETPD             = 5
  GDG cycle by      = GENERATION
```

Figure F-1 RMM LC ALL output

In the first test, Example F-1 shows the JCL that was used to create the tape data set.

### Example F-1 IEBGENER JCL

---

```
//STEP001 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=SHR,DSN=KARIM.A.CNTL(JOBCARD)
//SYSUT2 DD DSN=KARIM.TEST.MCATTR.ALL,
// DISP=(,CATLG),LABEL=(1,SL),RETPD=30,
// UNIT=ZOSVTS1,DCB=*.SYSUT1,VOL=(,RETAIN,,10)
```

---

In Figure F-2, the data set shows that the RETPD value in the JCL was the source of the expiration date (set by = OCE\_JFCB). Although MCATTR is set to All and is NOLIMIT NOLIMIT in MC class settings, which normally is a permanent retention combination, JCL-provided EXPDTP/RETPD overrides this value.

```

Data set name = KARIM.TEST.MCATTR.ALL
Volume       = B01234 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 10/04/2021 Create time = 15:53:59
Expiration date = 11/03/2021
Expir. time = 15:53:59
              set by = OCE_JFCB Original expir.date = 11/03/2021
LASTREF Extra Days = 0 WHILECATALOG = OFF
Management class = MCVTS1 VRS management value =
Storage group    = SGVTS1 VRS retention date =
Storage class    = SC3490 VRS retained = NO
Data class       = DC3490 Closed by Abend = NO
                  Deleted = NO
VRSEL exclude   = NO Catalog status = YES
Primary VRS details:
  Name =
  Job name = Type =
  Subchain NAME = Subchain start date =
Secondary VRS details:
  Value or class =
  Job name =
  Subchain NAME = Subchain start date =

```

Figure F-2 LD output

In Figure F-3, the volume also reflects the same expiration date. The Retention method set by field reflects OCE\_DEF. Because the MC settings did not specify the RM, the default was taken from EDGRMMxx parmlib.

```

Volume = B01234 VOL1 = Rack = Owner = KARIM
Type = LOGICAL Stacked count = 0 Jobname = KARIMT4
Worldwide ID = WORM = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign: Date = 10/04/2021 Time = 15:53:59 System ID = ZS24
User ID = KARIM
Expiration date = 11/03/2021 Original = 11/03/2021
Expiration time = 15:53:59 Datasets Kept By Catlg: 0
set by = OCE_JFCB
Retention date = Set retained = NO
Retention method= VRSEL
set by = OCE_DEF
retain by =
Data set name = KARIM.TEST.MCATTR.ALL

```

Figure F-3 LV output

In see Figure F-4, after EDGHSKP VRSEL runs, Retention Date becomes WHILECATALOG and the data set VRS is retained as shown by Availability = Vital Record and is displayed in the volume entry.

```

Volume = B01234   VOL1 =           Rack =           Owner = KARIM
  Type = LOGICAL           Stacked count = 0           Jobname = KARIMT4
  Worldwide ID =           WORM = N
Creation:  Date = 07/27/2016   Time = 14:32:24   System ID = ZS22
Assign:    Date = 10/04/2021   Time = 15:53:59   System ID = ZS24
                                     User ID = KARIM
Expiration date = 11/03/2021           Original = 11/03/2021
Expiration time = 15:53:59   Datasets Kept By Catlg: 0
      set by = OCE_JFCB
Retention date = WHILECATLG Set retained = NO
Retention method= VRSEL
      set by = OCE_DEF
      retain by =
Data set name = KARIM.TEST.MCATTR.ALL
Status = MASTER   Availability = Vital Record   Label = SL

```

Figure F-4 LV output after VRSEL

In Figure F-5, this data set is kept until its catalog entry is deleted and then it reaches its expiration date per the VRS Release Options that are defined in the VRS rule. However, if Expiry date ignore was set to YES, it expires during EDGHSKP EXPROC after the catalog entry is deleted rather than waiting for the expiration date to pass.

```

Data set mask = KARIM.**   Type = DSN
Job name mask =           Retain until expired = NO
Count = 99999 CYCLES           Retain while cataloged = YES
Delay = 0   Days in the HOME location
Store number = 99999 CYCLES in the HOME location
Priority = 0           Release Options:
                                     Expiry date ignore = NO
                                     Scratch immediate = NO
Next VRS in chain =           using VRS

```

Figure F-5 DATASET VRS

For the second test in this case study, JCL is submitted where the EXPDT or RETPD value is not passed by way of JCL. JCL submitted expiration dates are a higher priority, but because no JCL submitted expiration date exists, the expiration date is set from the MC class, which is set to NOLIMIT NOLIMIT.

The JCL is shown in Example F-2.

Example F-2 IEBGENER JCL

---

```

//STEP001 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD DISP=SHR,DSN=KARIM.A.CNTL(JOBCARD)
//SYSUT2 DD DSN=KARIM.TEST.MCATTR.ALL.T2,
// DISP=(,CATLG),LABEL=(1,SL),
// UNIT=ZOSVTS1,DCB=*.SYSUT1,VOL=(,RETAIN,,10)

```

---

In Figure F-6, the expiration date now shows 12/31/1999 and set by = OCE\_MC instead of set by = OCE\_DEF.

```

Data set name = KARIM.TEST.MCATTR.ALL.T2
Volume       = B02123 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 10/04/2021 Create time = 15:59:47
Expiration date = 12/31/1999          Expir. time =
      set by   = OCE_MC          Original expir.date =
Management class = MCVTS1          VRS management value =
Storage group   = SGVTS1          VRS retention date =
Storage class   = SC3490          VRS retained = NO
Data class      = DC3490          Closed by Abend = NO
                                   Deleted = NO
VRSEL exclude   = NO             Catalog status = YES
  
```

Figure F-6 LD output

OCE\_MC is also reflected at the volume level expiration date setting, as shown in Figure F-7.

```

Volume = B02123 VOL1 = Rack = Owner = KARIM
Type = LOGICAL Stacked count = 0 Jobname = KARIMT4
Worldwide ID = WORM = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign: Date = 10/04/2021 Time = 15:59:47 System ID = ZS24
                                   User ID = KARIM
Expiration date = 12/31/1999 Original =
Expiration time = Datasets Kept By Catlg: 0
      set by = OCE_MC
Retention date = Set retained = NO
Retention method= VRSEL
      set by = OCE_DEF
      retain by =
Data set name = KARIM.TEST.MCATTR.ALL.T2
Status = MASTER Availability = Label = SL
Action on release:
Scratch immediate = N Expiry date ignore = N
  
```

Figure F-7 LV output

After EDGHSKP VRSEL is run, a VRS is matched at the data set level as shown in Figure F-8.

```

Data set name = KARIM.TEST.MCATTR.ALL.T2
Volume       = B02123 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 10/04/2021 Create time = 15:59:47
Expiration date = 12/31/1999 Expir. time =
    set by = OCE_MC Original expir.date =
LASTREF Extra Days = 0 WHILECATALOG = OFF
Management class = MCVTS1 VRS management value =
Storage group = SGVTS1 VRS retention date = WHILECATLG
Storage class = SC3490 VRS retained = YES
Data class = DC3490 Closed by Abend = NO
    Deleted = NO
VRSEL exclude = NO Catalog status = YES
Primary VRS details:
    Name = KARIM.**
    Job name = Type = DATASET
    Subchain NAME = Subchain start date =
Secondary VRS details:
    Value or class =
    Job name =
    Subchain NAME = Subchain start date =
  
```

Figure F-8 LD output after VRSEL

Figure F-9 shows that the Availability field is updated to Vital Record at the volume level.

```

Volume = B02123 VOL1 = Rack = Owner = KARIM
Type = LOGICAL Stacked count = 0 Jobname = KARIMT4
Worldwide ID = WORM = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign: Date = 10/04/2021 Time = 15:59:47 System ID = ZS24
    User ID = KARIM
Expiration date = 12/31/1999 Original =
Expiration time = Datasets Kept By Catlg: 0
    set by = OCE_MC
Retention date = WHILECATLG Set retained = NO
Retention method= VRSEL
    set by = OCE_DEF
    retain by =
Data set name = KARIM.TEST.MCATTR.ALL.T2
Status = MASTER Availability = Vital Record Label = SL
  
```

Figure F-9 LV after VRSEL



In Figure F-10, after the catalog entry is deleted, the data set is now WHILECATALOG = OFF; however, the data set is left with an expiration date that never expires.

```

Data set name = KARIM.TEST.MCATTR.ALL.T2
Volume       = B02123 Physical file sequence number = 1
Owner        = KARIM Data set sequence = 1
Create date  = 10/04/2021 Create time = 15:59:47 System ID      =
Expiration date = 12/31/1999 Expir. time =
      set by   = OCE_MC Original expir.date =
LASTREF Extra Days = 0 WHILECATALOG = OFF
Management class = MCVTS1 VRS management value =
Storage group    = SGVTS1 VRS retention date = 10/04/2021
Storage class    = SC3490 VRS retained = NO
Data class       = DC3490 Closed by Abend = NO
                  Deleted = NO
VRSEL exclude   = NO Catalog status = NO
Primary VRS details:
  Name           = KARIM.**
  Job name       = Type = DATASET
  Subchain NAME = Subchain start date =
Secondary VRS details:
  Value or class =
  Job name       =
  Subchain NAME = Subchain start date =

```

Figure F-10 LD Permanent

In Figure F-11 after running EDGHSKP VRSEL, the volume is no longer kept by VRS because Availability is blank. However, it is in MASTER status and stays that way without manual intervention because the expiration date is 12/31/1999.

```

Volume = B02123 VOL1 = Rack = Owner = KARIM
Type = LOGICAL Stacked count = 0 Jobname = KARIMT4
Worldwide ID = WORM = N
Creation: Date = 07/27/2016 Time = 14:32:24 System ID = ZS22
Assign: Date = 10/04/2021 Time = 15:59:47 System ID = ZS24
User ID = KARIM
Expiration date = 12/31/1999 Original =
Expiration time = Datasets Kept By Catlg: 0
      set by = OCE_MC
Retention date = 10/04/2021 Set retained = NO
Retention method= VRSEL
      set by = OCE_DEF
      retain by =
Data set name = KARIM.TEST.MCATTR.ALL.T2
Status = MASTER Availability = Label = SL
Action on release:
Scratch immediate = N Expiry date ignore = N

```

Figure F-11 LV Permanent

The results of this case study show that when MCATTR(ALL) is used, the NOLIMIT NOLIMIT values in the MC can cause a permanent expiration value to be set. Therefore, care must be taken when MCATTR(ALL) is used to ensure that all tape data sets are allocated to newly defined MC classes that specify the intended expiration values.

It is better to use MCATTR(VRSELXDI) in an environment where VRSEL-managed volumes are present to prevent unintended values from the MC from determining the expiration date.

However, one use of MCATTR(ALL) and VRSEL in combination that might provide value is to use the MC DATE/DAYS Days Non-Usage values to populate the expiration date when none is provided by way of the JCL.



# Key takeaways from the case studies

After a review of the case studies that are presented in the appendixes of this publication, consider the following key takeaways:

- ▶ The Defaults Table can be used as a replacement for UXTABLE/EDGUX100 exit, but both should not be used together. If they are used together, EDGUX100 generally takes precedence.

It is also possible to change ACS routines to look for key dates that are passed in JCL instead of relying on EDGUX100 or Defaults Table for such processing.

- ▶ The Defaults Table and SMS ACS routines under MCATTR(VRSELXDI or ALL) can be used for specifying Retention Method and Expiration Attributes. However, a best practice is to use one or the other of these methods and not mix them. If both are used, the SMS ACS routines generally take precedence.
- ▶ Retention Method EXPDT and Retention Method VRSEL can be used together in the same environment, but should not be used on the same volume.
- ▶ EDGRMMxx parmib option MCATTR(ALL) is acceptable in a Retention Method EXPDT environment, but should not be used if any new data set allocations are still occurring to VRSEL-managed volumes. MCATTR(VRSELXDI) should be used instead. Otherwise, unexpected results with expiration date setting might occur from the unintended use of management class tape expiration values.
- ▶ When moving to MCATTR support of tape expiration management class attributes, new management classes must be created and ACS routines must be updated, so it is clear which tape data sets use the new expiration attributes. Reusing Managements Classes (MCs) can lead to unexpected results, unless the ACS routines are carefully analyzed and rewritten.

- ▶ In general, the following order of precedence in RMM processing is used for selecting expiration attributes:
  - JCL for EXPDT and RETPD only and EDGUX100.
  - Management Class tape expiration values if MCATTR(VRSELXDI or ALL).
  - Defaults Table tape expiration values. OVERRIDE can overrule JCL for EXPDT and RETPD.
  - EDGRMMxx parmlib default parameters that are governing tape expiration attributes.
- ▶ DFSMSrmm aggregates the data set attributes of all the data sets on the volume to set the volume level expiration attributes, choosing the attributes that keeps the volume retained the longest. As these conditions are satisfied and the expiration criteria of the data sets no longer apply, they are also removed at the volume level.
- ▶ Although RETAINBY(VOLUME) was the default for RM(VRSEL) and RM(EXPDT), it is recommended to use RETAINBY(SET) as a default for EDGRMMxx parmlib, Defaults Table, and MC class Volume Set Management Level.
 

Unless a specific need exists for a volume in the volume set to be allowed to expire before the remainder of the volume set, RETAINBY(SET) keeps volumes better protected from premature expiration.
- ▶ As of this writing, APAR [OA63568](#) is open against RETAINBY(VOLUME) that describes a scenario that can lead to improper setting of WHILECATALOG(UNTILEXPIRED) on a multi-volume chain.

# Related publications

The publications that are listed in this section are considered particularly suitable for a more detailed discussion of the topics that are covered in this book.

## IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Note that some publications that are referenced in this list might be available in softcopy only:

- ▶ *DFSMSrmm Primer*, [SG24-5983](#)
- ▶ *IBM TS7700 Release 5.3 Guide*, [SG24-8464](#)

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft, and additional materials, at the following website:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Other publications

The following publications are also relevant as further information sources:

- ▶ *z/OS 2.5 DFSMSrmm Implementation and Customization Guide*, [SC23-6874](#)
- ▶ *z/OS 2.5 DFSMSrmm Managing and Using Removable Media*, [SC23-6873](#)
- ▶ *z/OS 2.5 DFSMSrmm Reporting*, [SC23-6875](#)

## Help from IBM

IBM Support and downloads:

[ibm.com/support](http://ibm.com/support)

IBM Global Services:

[ibm.com/services](http://ibm.com/services)

## Stay connected to IBM Redbooks

- ▶ Find us on LinkedIn:

<https://www.linkedin.com/groups/2130806/>



**Redbooks**

**What is New in DFSMSrmm**

(0.2"spine)  
0.17"->0.473"  
90->249 pages









SG24-8529-01

ISBN 0738461369

Printed in U.S.A.

Get connected

