



ABCs of IBM z/OS System Programming Volume 3

Jose Gilberto Biondo Jr.



IBM Z



Redbooks



International Technical Support Organization

ABCs of IBM z/OS System Programming Volume 3

January 2018

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

Fifth Edition (January 2018)

This edition applies to version 2 release 3 of IBM z/OS (product number 5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2004, 2018. 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

Notices	ix
Trademarks	x
Preface	xi
The team who wrote this book	xi
Now you can become a published author, too!	xii
Comments welcome	xii
Stay connected to IBM Redbooks	xiii
Chapter 1. DFSMS introduction	1
1.1 Introduction to DFSMS	2
1.1.1 DFSMS components	2
1.2 DFSMSdfp base element	3
1.2.1 Managing storage	3
1.3 DFSMSdss optional feature	5
1.4 DFSMSrmm optional feature	5
1.4.1 Library management	5
1.4.2 Shelf management	6
1.4.3 Volume management	6
1.4.4 Data set management	6
1.5 DFSMShsm optional feature	6
1.5.1 Storage and space management	6
1.5.2 Tape mount management	7
1.5.3 Availability management	7
1.6 DFSTMStvs optional feature	7
1.6.1 VSAM record-level sharing	7
1.6.2 Recoverable resource management services	8
Chapter 2. Data set basics	9
2.1 Data sets on storage devices	10
2.1.1 Storage devices	10
2.1.2 DASD volumes	10
2.1.3 Tape volumes	10
2.2 Data set types	11
2.2.1 Non-VSAM data sets	11
2.2.2 VSAM data sets	12
2.2.3 z/OS UNIX files	13
2.2.4 Object data sets	14
2.2.5 Other data set attributes	15
2.3 Data set striping	17
2.3.1 Physical sequential and VSAM data sets	18
2.4 Accessing your data sets	19
2.4.1 Accessing cataloged data sets	20
2.4.2 Accessing uncataloged data sets	21
2.5 VTOC and DSCBs	21
2.5.1 Data set control block	21
2.5.2 VTOC index	21
2.5.3 Creating VTOC and VTOC Index	22

Chapter 3. Extended address volumes	23
3.1 Traditional DASD capacity	24
3.1.1 Serialization granularity	24
3.1.2 DASD virtual visibility	25
3.1.3 Multiple allegiance	25
3.1.4 Parallel access volume	25
3.1.5 HyperPAV feature	26
3.2 Extended access volumes	27
3.2.1 3390 Model A	27
3.2.2 EAV volumes architecture	28
3.2.3 EAS eligible data sets	29
3.2.4 EAS non-eligible data sets	30
3.2.5 Dynamic volume expansion	30
3.3 EAV and IGDSMSxx parmlib member	31
3.3.1 IGDSMSxx USEEAV parm	31
3.3.2 IGDSMSxx BreakPointValue	31
3.4 EATTR attribute	32
3.5 Migration assistance tracker	33
Chapter 4. Storage management software	35
4.1 Overview of DFSMSdfp utilities	36
4.1.1 System utility programs	36
4.1.2 Data set utility programs	36
4.2 DFSMSdfp access methods	37
4.2.1 Basic sequential access method	38
4.2.2 Basic direct access method	38
4.2.3 Basic partitioned access method	38
4.2.4 Queued sequential access method (QSAM)	38
4.2.5 Object access method	39
4.2.6 Virtual Storage Access Method	39
4.3 Access method services (IDCAMS)	40
4.3.1 Access method services commands	40
4.3.2 Starting the IDCAMS utility program	40
4.3.3 IDCAMS functional commands	41
4.3.4 AMS modal commands	42
4.3.5 DFSMS Data Collection Facility (DCOLLECT)	43
4.3.6 Generation data group	44
4.4 Virtual Storage Access Method	48
4.4.1 Logical record	49
4.4.2 Physical record	49
4.4.3 Control interval	49
4.4.4 Control area	50
4.4.5 Component	50
4.4.6 Cluster	50
4.4.7 Sphere	50
4.4.8 Other VSAM terminologies	51
4.4.9 Typical KSDS processing	53
4.4.10 Typical ESDS processing	55
4.4.11 Typical RRDS processing	55
4.4.12 Typical LDS processing	56
4.4.13 VSAM: Data-in-virtual	57
4.4.14 VSAM resource pool	58
4.4.15 VSAM: System-managed buffering	59

4.5 Data set separation	60
4.6 Data Facility sort	61
4.6.1 DFSORT functions	61
4.7 Data Set Services (DFSMSdss)	62
4.7.1 Physical and logical processing	63
4.7.2 DFMSMsdss stand-alone services	65
4.8 Hierarchical Storage Manager	65
4.8.1 DFMSHsm data sets	65
4.8.2 Storage device hierarchy	66
4.8.3 Availability management	68
4.8.4 Space management	70
4.9 Removable media manager (DFSMSrmm)	72
4.9.1 Resources managed by DFSMSrmm	73
4.9.2 Libraries and locations	75
4.9.3 Managing data	76
Chapter 5. System-managed storage	79
5.1 DFSMS and storage management	80
5.1.1 Managing storage with DFSMS	80
5.1.2 Goals and benefits of system-managed storage	81
5.1.3 Service level objectives	83
5.1.4 SMS configuration	84
5.1.5 SMS control data sets	85
5.1.6 Data sets eligible for SMS management	86
5.1.7 Data sets non-eligible for SMS management	87
5.1.8 Implementing DFSMS	87
5.2 DFSMS constructs	88
5.2.1 Data class	89
5.2.2 Storage class	90
5.2.3 Management class	92
5.2.4 Storage group	94
5.2.5 Using aggregate backup and recovery support	96
5.2.6 Automatic class selection routines	97
5.3 Maintaining and monitoring SMS policies	99
5.3.1 Starting SMS	99
5.3.2 Activating a new SMS configuration	99
5.3.3 Controlling SMS processing using operator commands	100
5.4 Interactive Storage Management Facility	101
5.4.1 ISMF: What you can do with ISMF	101
5.4.2 Accessing ISMF	102
Chapter 6. Catalogs	103
6.1 The integrated catalog facility structure	104
6.1.1 Basic catalog structure	105
6.1.2 VSAM volume data set	106
6.2 Aliases	107
6.2.1 Multilevel aliases	108
6.3 Catalog search order	108
6.3.1 Search order for catalogs for a data set define request	109
6.3.2 Search order for locating a data set	109
6.4 Using multiple catalogs	109
6.4.1 Splitting catalogs	110
6.4.2 Merging catalogs	110

6.5 Sharing catalogs across systems	110
6.6 Catalog caching	111
6.6.1 In-storage cache	111
6.6.2 Catalog data space caching	112
6.6.3 Enhanced Catalog Sharing	112
6.6.4 VSAM record-level sharing	112
6.7 Catalog address space	112
6.8 Maintaining your catalogs	113
6.8.1 Listing a catalog	113
6.8.2 Backup procedures	114
6.8.3 Recovery procedures	115
6.8.4 Checking the integrity on an ICF structure	116
6.8.5 Catalog performance	117
6.8.6 Working with the catalog address space.	118
Chapter 7. DFSMS Transactional VSAM Services	119
7.1 VSAM record-level sharing introduction	120
7.1.1 Supported data set types	121
7.1.2 Coupling facility overview	121
7.1.3 VSAM RLS sharing control data sets	121
7.1.4 Data set sharing under VSAM RLS.	122
7.2 VSAM RLS locking	122
7.3 Buffering under VSAM RLS	124
7.4 VSAM RLS/CICS data set recovery	125
7.5 Backup and recovery of VSAM data sets	126
7.5.1 Recover a VSAM data set.	126
7.5.2 Transactional recovery	126
7.6 The SMSVSAM address space	127
7.7 DFSMStvs introduction	128
7.7.1 Components used by DFSMStvs	129
7.7.2 DFSMStvs use of z/OS RRMS	129
7.7.3 Atomic updates	130
7.7.4 Unit of work and unit of recovery.	131
7.7.5 DFSMStvs logging	131
7.8 Accessing a data set with DFSMStvs	133
7.9 Application considerations	133
7.9.1 Break processing into a series of transactions	133
7.9.2 Modify the program or JCL to request DFSMStvs access	134
7.10 The DFSMStvs instance	134
7.10.1 Interacting with DFSMStvs	135
Chapter 8. Storage management hardware	137
8.1 Overview of DASD types.	138
8.1.1 DASD based on RAID technology	138
8.2 Redundant Array of Independent Disks	138
8.2.1 RAID implementations	139
8.3 IBM DS8000 series	140
8.3.1 DS8000 hardware overview	141
8.3.2 DS8000 major components	143
8.4 IBM TotalStorage Resiliency Family	145
8.4.1 Copy Services.	145
8.4.2 FlashCopy.	146
8.4.3 Metro Mirror function.	146

8.4.4 Global Copy function	146
8.4.5 Global Mirror function	147
8.4.6 Metro/Global Mirror function	147
8.4.7 z/OS Global Mirror function	147
8.5 DS8000 performance features	148
8.5.1 I/O priority queuing	148
8.5.2 Custom volumes	148
8.5.3 Improved caching algorithms	148
8.5.4 I/O Priority manager	148
8.5.5 Easy Tier	149
8.5.6 Thin provisioning	149
8.6 Introduction to tape processing	149
8.6.1 SL and NL format	150
8.6.2 Tape cartridges	151
8.7 IBM TS1155 tape drive	152
8.8 IBM TS4500 tape library	152
8.9 Introduction to IBM TS7700	153
8.9.1 IBM TS7700 models	153
8.10 Introduction to TS7700 Grid configuration	154
8.10.1 Copy consistency	155
8.10.2 VTS advanced functions	155
8.11 Storage area network	156
8.11.1 FICON and SAN	158
Related publications	159
IBM Redbooks publications	159
Other publications	159
Online resources	160
How to get IBM Redbooks publications	160
Help from IBM	160

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 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.

CICS®	Geographically Dispersed Parallel	PowerPC®
DB2®	Sysplex™	PR/SM™
DS6000™	Hiperspace™	RACF®
DS8000®	HyperSwap®	Redbooks®
Easy Tier®	IBM®	Redbooks (logo)  ®
Enterprise Storage Server®	IMS™	RMF™
eServer™	Language Environment®	S/390®
FICON®	Magstar®	VTAM®
FlashCopy®	MVS™	z/Architecture®
GDPS®	Parallel Sysplex®	z/OS®
	POWER8®	

The following terms are trademarks of other companies:

Linear Tape-Open, LTO, Ultrium, the LTO Logo and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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

Preface

The ABCs of IBM® z/OS® System Programming is a 13-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information that you need to start your research into z/OS and related subjects. The ABCs collection serves as a powerful technical tool to help you become more familiar with z/OS in your current environment, or to help you evaluate platforms to consolidate your e-business applications.

This edition is updated to z/OS Version 2 Release 3.

The other volumes contain the following content:

- ▶ Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation
- ▶ Volume 2: z/OS implementation and daily maintenance, defining subsystems, IBM Job Entry Subsystem 2 (JES2) and JES3, link pack area (LPA), LNKLST, authorized libraries, System Modification Program Extended (SMP/E), IBM Language Environment®
- ▶ Volume 4: Communication Server, TCP/IP, and IBM VTAM®
- ▶ Volume 5: Base and IBM Parallel Sysplex®, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart manager (ARM), IBM Geographically Dispersed Parallel Sysplex™ (IBM GDPS®)
- ▶ Volume 6: Introduction to security, IBM RACF®, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries firewall technologies, LDAP, and Enterprise Identity Mapping (EIM)
- ▶ Volume 7: Printing in a z/OS environment, Infoprint Server, and Infoprint Central
- ▶ Volume 8: An introduction to z/OS problem diagnosis
- ▶ Volume 9: z/OS UNIX System Services
- ▶ Volume 10: Introduction to IBM z/Architecture®, the IBM Z platform, IBM Z connectivity, LPAR concepts, HCD, and DS Storage Solution.
- ▶ Volume 11: Capacity planning, performance management, WLM, IBM RMF™, and SMF
- ▶ Volume 12: WLM
- ▶ Volume 13: JES3, JES3 SDSF

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Jose Gilberto Biondo Jr. is an IT Specialist in Integrated Technology Delivery, ServerSystems Operations/Storage Management in IBM Brazil. He has eight years of experience in z/OS, working with storage management since 2007. Jose works mainly with IBM storage products (DFSMSdfp, DFSMSdss, DFSMSHsm, and DFSMSrmm), but he also works with OEM software products. Jose's areas of expertise include installing and maintaining storage products, and process automation.

Thanks also to the following contributors to this edition:

Lydia Parziale
Robert Haimowitz
William G. White
International Technical Support Organization, Poughkeepsie Center

Wayne Rhoten
IBM, US

Marcelo Lopes de Moraes
IBM, Brazil

Thanks to the authors of the previous editions of this book:
Paul Rogers, Redelf Janssen, Rita Pleus, Valeria Sokal, Andre Otto, Alvaro Salla

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 ITSO 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

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® in one of the following ways:

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

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts>
- ▶ Follow us on twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for 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>



DFSMS introduction

This chapter gives a brief overview of the Data Facility Storage Management Subsystem (DFSMS) and its primary functions in the z/OS operating system. DFSMS is a suite of data and storage management functions for the z/OS operating system. It is composed of a base element and four optional features. DFSMS manages data from creation to expiration.

DFSMS is an operating environment that helps automate and centralize the management of storage based on the policies that your installation defines for availability, performance, space, and security.

The heart of DFSMS is the storage management subsystem (SMS). Using SMS, the storage administrator defines policies that automate the management of storage and hardware devices. These policies describe data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements for the system.

DFSMSdfp is a base element of z/OS. DFSMSdfp is automatically included with z/OS. DFSMSdfp performs the essential data, storage, and device management functions of the system. DFSMSdfp and DFSMShsm provide disaster recovery functions such as Advanced Copy Services and aggregate backup and recovery support (ABARS).

The other elements of DFSMS (DFSMSdss, DFSMShsm, DFSMSrmm, and DFSMStvs) are optional features that complement DFSMSdfp to provide a fully integrated approach to data and storage management. In a system-managed storage environment, DFSMS automates and centralizes storage management based on the policies that your installation defines for availability, performance, space, and security. With the optional features enabled, you can take full advantage of all the functions that DFSMS offers.

This chapter includes the following sections:

- ▶ Introduction to DFSMS
- ▶ DFSMSdfp base element
- ▶ DFSMSdss optional feature
- ▶ DFSMSrmm optional feature
- ▶ DFSMShsm optional feature
- ▶ DFSMStvs optional feature

1.1 Introduction to DFSMS

Data management is the part of the operating system that organizes, identifies, stores, catalogs, and retrieves all the data information (including programs) that your installation uses. DFSMS is an exclusive element of the z/OS operating system. DFSMS is a software suite that automatically manages data from creation to expiration.

DFSMSdfp helps you store and catalog information about direct access storage device (DASD), optical, and tape devices so that it can be quickly identified and retrieved from the system. DFSMSdfp provides access to both record- and stream-oriented data in the z/OS environment. The z/OS operating system enables you to efficiently manage e-business workloads and enterprise transactions 24 hours a day. DFSMSdfp is automatically included with z/OS. It performs the essential data, storage, and device management functions of the system.

As a systems programmer, you can use DFSMS data management to perform these tasks:

- ▶ Allocate space on DASD and optical volumes
- ▶ Automatically locate cataloged data sets
- ▶ Control access to data
- ▶ Transfer data between the application program and the medium
- ▶ Mount magnetic tape volumes in the drive

1.1.1 DFSMS components

Five different DFSMS components provide specific data management options for systems programmers to allocate data in the correct media type and volume, and provide data access, data availability, and data retention management. The following elements comprise DFSMS:

- DFSMSdfp** A base element of z/OS that provides storage, data, program, and device management. It consists of components such as access methods, OPEN/CLOSE/EOV routines, catalog management, DADSM (DASD space control), utilities, IDCAMS, SMS, NFS, ISMF, and other functions.
- DFSMSdss** An optional element of z/OS that provides data movement, copy, backup, and space management functions.
- DFSMShsm** An optional element of z/OS that provides backup, recovery, migration, and space management functions. It calls DFSMSdss for certain of its functions.
- DFSMSrmm** An optional element of z/OS that provides management functions for removable media such as tape cartridges and optical media.
- DFSMStvs** An optional element of z/OS that enables batch jobs and IBM CICS® online transactions to update shared VSAM data sets concurrently.

Network File System

The Network File System (NFS) is a distributed file system that enables users to access UNIX files and directories that are on remote computers as though they were local, or export local files to remote computers. NFS is independent of machine types, operating systems, and network architectures.

Figure 1-1 shows the relationship between DFSMS and other z/OS components.

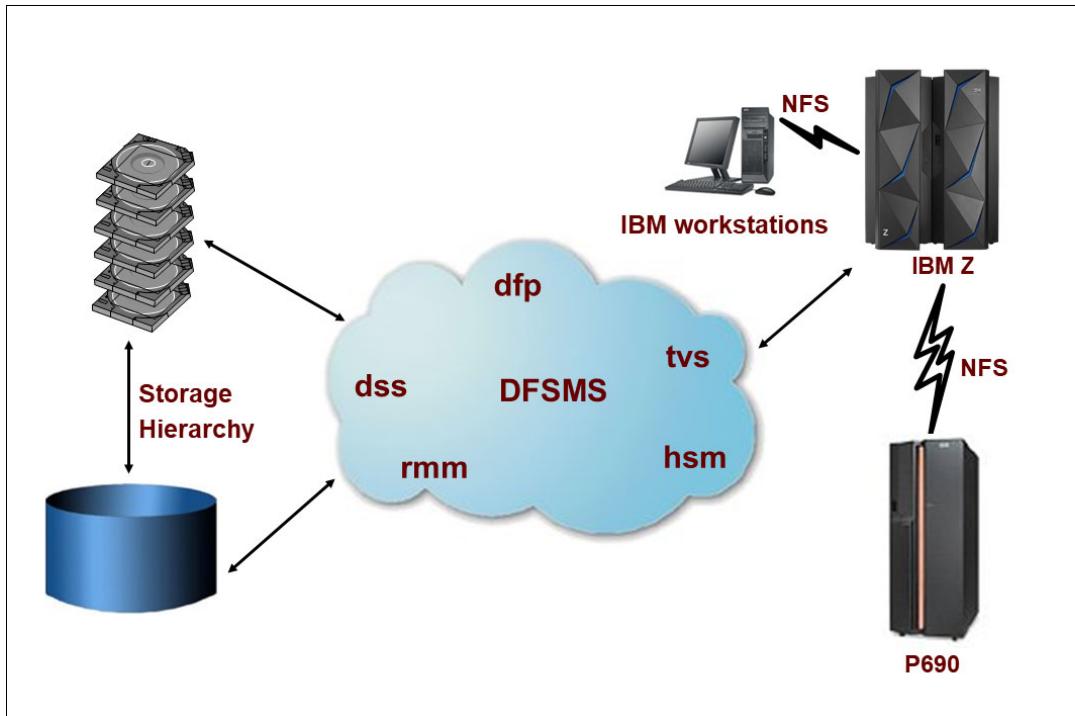


Figure 1-1 Data Facility Storage Management Subsystem

The following topics introduce you to each DFSMS component, and describe in more detail their function throughout the data set lifecycle.

1.2 DFSMSdfp base element

DFSMS Data Facility Product (DFSMSdfp) provides storage, data, program, and device management. It consists of components such as access methods, OPEN/CLOSE/EOV routines, catalog management, DADSM (DASD space control), utilities, IDCAMS, SMS, NFS, ISMF, and other functions. It is a base element on z/OS operating systems, and each of its functions are described in this section.

1.2.1 Managing storage

SMS is a DFSMSdfp facility designed for automating and centralizing storage management. SMS can be used to group a number of similar devices and direct different types of data to each group based on data requirements and system policies. Users and administrators can use ISMF windows to view some of these definitions and management policies.

For more information, see 5.4, “Interactive Storage Management Facility” on page 101.

Managing data

DFSMSdfp organizes, identifies, stores, catalogs, shares, and retrieves all the data that your installation uses. You can store data on DASD, magnetic tape volumes, or optical volumes. Using data management, you can complete the following tasks:

- ▶ Allocate space on DASD and optical volumes
- ▶ Automatically locate cataloged data sets

- ▶ Control access to data
- ▶ Transfer data between the application program and the medium
- ▶ Mount magnetic tape volumes in the drive

Using access methods, commands, and utilities

DFSMSdfp manages the organization and storage of data in the z/OS environment. You can use access methods with macro instructions to organize and process a data set or object. Access method services commands manage data sets, volumes, and catalogs. Utilities perform tasks such as copying and moving data. You can use system commands to display and set SMS configuration parameters, use DFSMSdfp callable services to write advanced application programs, and use installation exits to customize DFSMS.

Managing devices with DFSMSdfp

You need to use the hardware configuration definition (HCD) to define I/O devices to the operating system, and to control these devices. DFSMS manages DASD, storage control units, magnetic tape devices, optical devices, and printers. You can use DFSMS functions to manage many device types, but most functions apply specifically to one type or one family of devices.

Tape mount management

Tape mount management is a methodology for improving tape usage and reducing tape costs. This methodology involves intercepting selected tape data set allocations through the SMS automatic class selection (ACS) routines and redirecting them to a direct access storage device (DASD) buffer. After the data sets are on DASD, you can migrate them to a single tape or small set of tapes, reducing the overhead associated with multiple tape mounts.

Distributed data access with DFSMSdfp

In the distributed computing environment, applications must often access data on other computers in a network. Often, the most effective data access services occur when applications can access remote data as though it were local data.

The IBM Distributed FileManager/MVS™ is a DFSMSdfp client/server product that enables remote clients in a network to access data on z/OS systems. Distributed FileManager/MVS provides workstations with access to z/OS data. Users and applications on heterogeneous client computers in your network can take advantage of system-managed storage on z/OS, data sharing, and data security with RACF.

z/OS UNIX System Services (z/OS UNIX) provide the command interface that interactive UNIX users can use. z/OS UNIX allows z/OS programs to directly access UNIX data.

Advanced copy services

Advanced copy services include remote and point-in-time copy functions that provide backup and recovery of data. When used before a disaster occurs, Advanced Copy Services provide rapid backup of critical data with minimal impact to business applications. If a disaster occurs to your data center, Advanced Copy Services provide rapid recovery of critical data.

Object Access Method

Object access method (OAM) provides storage, retrieval, and storage hierarchy management for objects. OAM also manages storage and retrieval for tape volumes that are contained in system-managed libraries.

1.3 DFSMSdss optional feature

DFSMS Data Set Services (DFSMSdss) is the primary data mover for DFSMS. DFSMSdss copies and moves data to help manage storage, data, and space more efficiently. It can efficiently move multiple data sets from old to new DASD. The data movement capability that is provided by DFSMSdss is useful for many other operations, as well. You can use DFSMSdss to perform the following tasks:

- ▶ Data movement and replication

DFSMSdss allows you to move or copy data between volumes of like and unlike device types. By using DFSMSdss, you can create multiple backup copies of your data within a single DUMP command.

- ▶ Space management

DFSMSdss enables you to perform space management against DASD volumes, to both reduce the number of extents your data have and the fragmentation of your DASD volumes.

- ▶ Data backup and recovery

DFSMSdss provides you with host system backup and recovery functions at both the data set and volume levels. It also includes a stand-alone restore program that you can run without a host operating system.

- ▶ Data set and volume conversion

DFSMSdss can convert your data sets and volumes to system-managed storage. It can also return your data to a non-system-managed state as part of a recovery procedure.

- ▶ Distributed data management

DFSMSdss saves distributed data management (DDM) attributes that are associated with a specific data set and preserves those attributes during copy and move operations. DFSMSdss also offers the IBM FlashCopy® and SnapShot features with IBM DS8000® family. FlashCopy and SnapShot function automatically, work much faster than traditional data movement methods, and are well-suited for handling large amounts of data.

- ▶ Concurrent copy

When it is used with supporting hardware, DFSMSdss also provides concurrent copy capability. Concurrent copy allows you to copy or back up data while that data is being used. The user or application program determines when to start the processing, and the data is copied as though no updates have occurred.

1.4 DFSMSrmm optional feature

DFSMS Removable Media Manager (DFSMSrmm) manages your removable media resources, including tape cartridges. It provides the following functions:

- ▶ Library management
- ▶ Shelf management
- ▶ Volume management
- ▶ Data set management

1.4.1 Library management

You can create tape libraries, or collections of tape media associated with tape drives, to balance the work of your tape drives and help the operators that use them.

DFSMSrmm can manage the following devices:

- ▶ A removable media library, which incorporates all other libraries, such as:
 - System-managed manual tape libraries.
 - System-managed automated tape libraries. Examples of automated tape libraries include:
 - IBM TS4500
 - IBM TS7760
- ▶ Non-system-managed or traditional tape libraries, including automated libraries such as a library under Basic Tape Library Support (BTLS) control.

1.4.2 Shelf management

DFSMSrmm groups information about removable media by shelves into a central online inventory, and keeps track of the volumes on those shelves. DFSMSrmm can manage the shelf space that you define in your removable media library and in your storage locations.

1.4.3 Volume management

DFSMSrmm manages the movement and retention of tape volumes throughout their lifecycle. You can assign tapes to different vault locations to plan for disaster recovery, or send data to other sites.

1.4.4 Data set management

DFSMSrmm records information about the data sets on tape volumes. DFSMSrmm uses the data set information to validate volumes and to control the retention and movement of those data sets.

1.5 DFMSHsm optional feature

DFSMS Hierarchical Storage Manager (DFMSHsm) complements DFMSdss to provide the following functions:

- ▶ Storage and space management
- ▶ Tape mount management
- ▶ Availability management

1.5.1 Storage and space management

DFMSHsm improves DASD space usage by keeping only active data on fast-access storage devices. It automatically frees space on user volumes by deleting eligible data sets, releasing overallocated space, and moving low-activity data to lower cost-per-byte devices, based on storage administrator defined policies. DFMSHsm also includes commands to perform space management for tapes it controls, including a recycle command that allows merging tapes with low utilization levels into fewer tapes, increasing tape utilization efficiency.

1.5.2 Tape mount management

DFSMShsm can write multiple output data sets to a single tape, making it a useful tool for implementing tape mount management under SMS. When you redirect tape data set allocations to DASD, DFSMShsm can move those data sets to tape, as a group, during space management cycle. This methodology greatly reduces the number of tape mounts on the system. DFSMShsm uses a single-file format, which improves your tape usage and search capabilities.

1.5.3 Availability management

DFSMShsm backs up your data, automatically or by command, to ensure availability if accidental loss of the data sets or physical loss of volumes occurs. DFSMShsm also allows the storage administrator to copy backup and migration tapes, and to specify that copies be made in parallel with the original. You can store the copies onsite as protection from media damage, or offsite as protection from site damage.

DFSMShsm also provides disaster backup and recovery for user-defined groups of data sets (aggregates) so that you can restore critical applications at the same location or at an offsite location.

Important: You must also have DFMSDss to use the DFSMShsm functions.

1.6 DFSMStvs optional feature

DFSMS Transactional VSAM Services (DFSMStvs) allows you to share VSAM data sets across CICS, batch, and object-oriented applications on z/OS or distributed systems. DFSMStvs enables concurrent shared updates of recoverable VSAM data sets by CICS transactions and multiple batch applications. DFSMStvs enables 24-hour availability of CICS and batch applications.

1.6.1 VSAM record-level sharing

With VSAM record-level sharing (RLS), multiple CICS systems can directly access a shared VSAM data set, eliminating the need to move functions between the application-owning regions and file-owning regions. CICS provides the logging, commit, and backout functions for VSAM recoverable data sets. VSAM RLS provides record-level serialization and cross-system caching. CICSVR provides a forward recovery utility.

DFSMStvs is built on top of VSAM RLS, which permits sharing of recoverable VSAM data sets at the record level. Different applications often need to share VSAM data sets. Sometimes the applications only need to read the data set. Sometimes an application needs to update a data set while other applications are reading it. The most complex case of sharing a VSAM data set is when multiple applications need to update the data set and all require complete data integrity.

Transaction processing provides functions that coordinate work flow and the processing of individual tasks for the same data sets. VSAM record-level sharing and DFSMStvs provide key functions that enable multiple batch update jobs to run concurrently with CICS access to the same data sets, while maintaining integrity and recoverability.

1.6.2 Recoverable resource management services

RRMS is part of the operating system and comprises registration services, context services, and Resource Recovery Services (RRS). RRMS provides the context and unit of recovery management under which DFSMStvs participates as a recoverable resource manager.



Data set basics

A *data set* is a collection of logically related data. It can be a source program, a library of macros, or a file of data records used by a processing program. Data records (also called logical records) are the basic unit of information used by a processing program. This chapter introduces you to the data set types available for use on z/OS, the devices they can be allocated onto, and their characteristics.

By placing your data into volumes of organized data sets, you can save and process the data efficiently. You can also print the contents of a data set, or display the contents on a workstation.

This chapter includes the following sections:

- ▶ Data sets on storage devices
- ▶ Data set types
- ▶ Data set striping
- ▶ Accessing your data sets
- ▶ VTOC and DSCBs

2.1 Data sets on storage devices

A *z/OS data set* is a collection of logically related data records stored on one or a set of volumes. A data set can be, for example, a source program, a library of macros, or a file of data records used by a processing program. You can print a data set or display it on a workstation. The *logical record* is the basic unit of information used by a processing program.

Note: As an exception, the z/OS UNIX services component supports zSeries file system (ZFS) data sets, where the collection is of bytes and the concept of logically related data records is not used.

2.1.1 Storage devices

Data can be stored on a magnetic direct access storage device (DASD), magnetic tape volume, or optical media. As mentioned previously, the term *DASD* applies to disks or simulated equivalents of disks. All types of data sets can be stored on DASD, but only sequential data sets can be stored on magnetic tape. The types of data sets are described in 2.2, “Data set types” on page 11.

2.1.2 DASD volumes

Each block of data on a DASD volume has a distinct location and a unique address, making it possible to find any record without extensive searching. You can store and retrieve records either directly or sequentially. Use DASD volumes for storing data and executable programs, including the operating system itself, and for temporary working storage. You can use one DASD volume for many separate data sets, and reallocate or reuse space on the volume.

2.1.3 Tape volumes

A tape volume provides serial access to the media for read and write operations. Because only sequential access is possible, only sequential data sets can be stored on tapes. A tape can be physical like model 3592JD used in automated tape libraries (ATLs) or logical, emulated by hardware like IBM Virtualization Engine (VE).

The following sections discuss the logical attributes of a data set, which affects how a data set can be created, which devices can store it, and what access method can be used to access it.

2.2 Data set types

The data organization that you choose depends on your applications and the operating environment. z/OS allows you to use temporary or permanent data sets, and to use several ways to organize files for data to be stored on magnetic media, as described in this section.

2.2.1 Non-VSAM data sets

Non-VSAM data sets are the most common type of data set found on z/OS. They are a collection of fixed-length or variable-length records, which can be physically stored in groups (blocks), or individually. There are several different types of non-VSAM data sets:

- ▶ Physical sequential data set
- ▶ Partitioned data set
- ▶ Partitioned data set extended

Physical sequential data set

Physical sequential (PS) data sets contain logical records that are stored in physical order. New records are appended to the end of the data set. You can specify a sequential data set in extended format or not.

Partitioned data set

Partitioned data sets (PDSs) are similar in organization to a library and are often referred to this way. Normally, a library contains a great number of “books,” and sorted directory entries that are used to locate them. In a partitioned organized data set, the “books” are called members, and to locate them, they are pointed to by entries in a directory.

The members are individual sequential data sets and can be read or written sequentially, after they have been located by directory. Then, the records of a specific member are written or retrieved sequentially. Partitioned data sets can only exist on DASD and is not eligible for multivolume. Each member has a unique name that is one to eight characters in length and is stored in a directory that is part of the data set.

Space for the directory is expressed in 256 byte blocks. Each block contains from 3 to 21 entries, depending on the length of the user data field. If you expect 200 directory entries, request at least 30 blocks. Any unused space on the last track of the directory is wasted unless there is enough space left to contain a block of the first member.

Partitioned data set extended

Partitioned data set extended (PDSE) is a type of data set organization that improves the PDS organization. It has an improved indexed directory structure and a different member format. You can use PDSE for source (programs and text) libraries, macros, and program object (the name of executable code when loaded in PDSE) libraries.

Logically, a PDSE directory is similar to a PDS directory. It consists of a series of directory records in a block. Physically, it is a set of “pages” at the front of the data set, plus additional pages interleaved with member pages. Five directory pages are initially created at the same time as the data set.

New directory pages are added, interleaved with the member pages, as new directory entries are required. A PDSE always occupies at least five pages of storage. It cannot be overwritten by being opened for sequential output.

There are several advantages of using PDSE data sets over regular PDS. For a list of the differences, see *IBM z/OS DFSMS Using data sets*, SC23-6855.

You can define several attributes when allocating a new non-VSAM file, including data set size, record length, record format, and so on. For a complete list of attributes and their functions, see *IBM z/OS DFSMS Using data sets*, SC23-6855.

2.2.2 VSAM data sets

Virtual Storage Access Method (VSAM) data sets are formatted differently than non-VSAM data sets. Except for linear data sets, VSAM data sets are collections of records, grouped into control intervals. The *control interval* is a fixed area of storage space in which VSAM stores records. The control intervals are grouped into contiguous areas of storage called *control areas*. To access VSAM data sets, use the VSAM access method.

VSAM arranges records by an index key, by a relative byte address, or by a relative record number. VSAM data sets are cataloged for easy retrieval.

Any type of VSAM data set can be in extended format. Extended-format data sets have a different internal storage format than data sets that are not extended. This storage format gives extended-format data sets additional usability characteristics and possibly better performance due to striping. You can choose that an extended-format key-sequenced data set be in the compressed format. Extended-format data sets must be SMS-managed.

For more information about VSAM data sets, see 4.4, “Virtual Storage Access Method” on page 48.

Key-sequenced data set

In a key-sequenced data set (KSDS), logical records are placed in the data set in ascending collating sequence by key. The key contains a unique value, which determines the record's collating position in the cluster. The key must be in the same position (off set) in each record.

The key field must be contiguous and each key's contents must be unique. After it is specified, the value of the key cannot be altered, but the entire record can be deleted. When a new record is added to the data set, it is inserted in its logical collating sequence by key.

A KSDS has a data component and an index component. The index component tracks the used keys and is used by VSAM to retrieve a record from the data component quickly when a request is made for a record with a certain key.

A KSDS can have fixed or variable length records. A KSDS can be accessed in sequential mode, direct mode, or skip sequential mode (meaning that you process sequentially, but directly skip portions of the data set).

Entry-sequenced data set

An entry sequenced data set (ESDS) is comparable to a sequential data set. It contains fixed-length or variable-length records. Records are sequenced by the order of their entry in the data set, rather than by a key field in the logical record. All new records are placed at the end of the data set. An ESDS cluster has only a data component.

Records can be accessed sequentially or directly by relative byte address (RBA). When a record is loaded or added, VSAM indicates its RBA. The RBA is the offset of the first byte of the logical record from the beginning of the data set. The first record in a data set has an RBA of 0, the second record has an RBA equal to the length of the first record, and so on. The RBA of a logical record depends only on the record's position in the sequence of records. The RBA is always expressed as a full-word binary integer.

Although an entry-sequenced data set does not contain an index component, alternate indexes are allowed. You can build an alternate index with keys to track these RBAs.

Relative-record data set

A relative record data set (RRDS) consists of a number of preformed, fixed-length slots. Each slot has a unique relative record number, and the slots are sequenced by ascending relative record number. Each (fixed length) record occupies a slot, and it is stored and retrieved by the relative record number of that slot. The position of a data record is fixed, so its relative record number cannot change.

An RRDS cluster has a data component only. Random load of an RRDS requires a user program implementing such logic.

Linear data set

A linear data set (LDS) VSAM data set contains data that can be accessed as byte-addressable strings in virtual storage. A linear data set does not have embedded control information that other VSAM data sets hold.

The primary difference among these types of data sets is how their records are stored and accessed. VSAM arranges records by an index key, by relative byte address, or by relative record number. Data organized by VSAM must be cataloged and is stored in one of four types of data sets, depending on an application designer option.

2.2.3 z/OS UNIX files

z/OS UNIX System Services (z/OS UNIX) enables applications and even z/OS to access UNIX files. UNIX applications also can access z/OS data sets. You can use the hierarchical file system (HFS), z/OS Network File System (z/OS NFS), zSeries file system (zFS), and temporary file system (TFS) with z/OS UNIX. You can use the BSAM, QSAM, BPAM, and VSAM access methods to access data in UNIX files and directories. z/OS UNIX files are byte-oriented, similar to objects.

Hierarchical file system

On DASD, you can define an HFS data set on the z/OS system. Each HFS data set contains a hierarchical file system. Each hierarchical file system is structured like a tree with subtrees, and consists of directories and all their related files. Although HFS is still available, zFS provides the same capabilities with a better performance. Plan using zFS instead.

z/OS Network File System

The z/OS NFS is a distributed file system that enables users to access UNIX files and directories that are on remote computers as though they were local. NFS is independent of machine types, operating systems, and network architectures. Use the NFS for file serving (as a data repository) and file sharing between platforms supported by z/OS.

Figure 2-1 illustrates the client/server relationship:

- ▶ The upper center portion shows the DFSMS NFS address space server, and the lower portion shows the DFSMS NFS address space client.
- ▶ The left side of the figure shows various NFS clients and servers that can interact with the DFSMS NFS server and client.
- ▶ In the center of the figure is the Transmission Control Protocol/Internet Protocol (TCP/IP) network used to communicate between clients and servers.

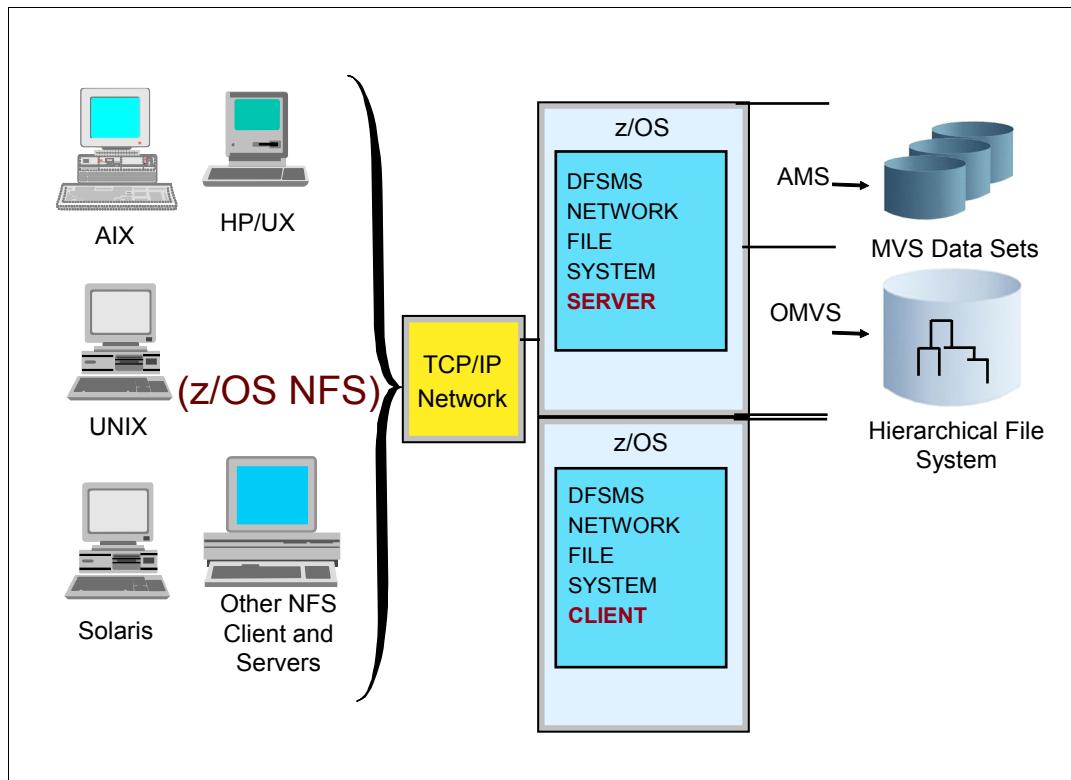


Figure 2-1 DFSMS Network File System

zSeries file system

A zFS is a UNIX file system that contains one or more file systems in a VSAM linear data set. zFS is application compatible with HFS and more performance efficient than HFS.

Temporary file system

A TFS is stored in memory and delivers high-speed I/O. A systems programmer can use a TFS for storing temporary files.

2.2.4 Object data sets

Objects are named streams of bytes that have no specific format or record orientation. Use the object access method (OAM) to store, access, and manage object data. You can use any type of data in an object because OAM does not recognize the content, format, or structure of the data. For example, an object can be a scanned image of a document, an engineering drawing, or a digital video. OAM objects are stored either on DASD in an IBM DB2® database, on an optical drive, or on tape storage volumes.

The storage administrator assigns objects to object storage groups and object backup storage groups. The object storage groups direct the objects to specific DASD, optical, or tape devices, depending on their performance requirements. You can have one primary copy of an object and up to two backup copies of an object. A Parallel Sysplex allows you to access objects from all instances of OAM and from optical hardware within the sysplex.

2.2.5 Other data set attributes

This section describes these data set types and their attributes:

- ▶ Basic format data sets
- ▶ Large format data sets
- ▶ Extended-format data sets
- ▶ Virtual input/output data sets
- ▶ Data set organization
- ▶ Data set naming

Basic format data sets

Basic format data sets are sequential data sets that are specified as neither extended-format nor large-format. Basic format data sets have a size limit of 65,535 tracks (4369 cylinders) per volume. They can be system-managed or not, and can be accessed by using QSAM, BSAM, or EXCP.

You can allocate a basic format data set by using the DSNTYPE=BASIC parameter on the DD statement, dynamic allocation (SVC 99), TSO/E ALLOCATE, or the access method services ALLOCATE command, or the data class. If no DSNTYPE value is specified from any of these sources, then its default is BASIC.

Large format data sets

Large format data sets are sequential data sets that can grow beyond the size limit of 65,535 tracks (4369 cylinders) per volume that applies to other sequential data sets. Large format data sets can be system-managed or not. They can be accessed by using QSAM, BSAM, or EXCP.

Large format data sets reduce the need to use multiple volumes for single data sets, especially very large ones such as spool data sets, memory dumps, logs, and traces. Unlike extended-format data sets, which also support more than 65,535 tracks per volume, large format data sets are compatible with EXCP and do not need to be SMS-managed.

Data sets defined as large format must be accessed by using QSAM, BSAM, or EXCP.

Large format data sets have a maximum of 16 extents on each volume. Each large format data set can have a maximum of 59 volumes. Therefore, a large format data set can have a maximum of 944 extents (16 times 59).

A large format data set can occupy any number of tracks, without the limit of 65,535 tracks per volume. The minimum size limit for a large format data set is the same as for other sequential data sets that contain data: One track, which is about 56,000 bytes. Primary and secondary space can both exceed 65,535 tracks per volume.

Large format data sets can be on SMS-managed DASD or non-SMS-managed DASD.

Restriction: The following types of data sets cannot be allocated as large format data sets:

- ▶ PDS, PDSE, and direct data sets
- ▶ Virtual I/O data sets, password data sets, and system memory dump data sets

You can allocate a large format data set by using the DSNTYPE=LARGE parameter on the DD statement, dynamic allocation (SVC 99), TSO/E ALLOCATE, or the access method services ALLOCATE command.

For more information about large data sets, see *IBM z/OS DFSMS Using data sets*, SC23-6855.

Extended-format data sets

While sequential data sets have a maximum of 16 extents on each volume, extended-format data sets have a maximum of 123 extents on each volume. Each extended-format data set can have a maximum of 59 volumes, so an extended-format sequential data set can have a maximum of 7257 extents (123 times 59).

When defined as extended-format, sequential data sets can go beyond the 65,535 tracks limitation per data set. For VSAM files, extended addressability needs to be enabled for the data set in addition to extended-format.

An extended-format, striped sequential data set can contain up to approximately four billion blocks. The maximum size of each block is 32 760 bytes.

An extended-format data set supports the following additional functions:

- ▶ Compression, which reduces the space for storing data and improves I/O, caching, and buffering performance.
- ▶ Data striping, which in a sequential processing environment distributes data for one data set across multiple SMS-managed DASD volumes, improving I/O performance and reducing the batch window. For more information about data set striping, see 2.3, "Data set striping" on page 17.
- ▶ Extended-addressability, which enables you to create a VSAM data set that is larger than 4 GB.
- ▶ They are able to recover from padding error situations.
- ▶ They can use the system-managed buffering (SMB) technique.

Virtual input/output data sets

You can manage temporary data sets with a function called virtual input/output (VIO). VIO uses DASD space and system I/O more efficiently than other temporary data sets.

You can use the BPAM, BSAM, QSAM, BDAM, and EXCP access methods with VIO data sets. SMS can direct SMS-managed temporary data sets to VIO storage groups.

Data set organization

Data set organization (DSORG) specifies the organization of the data set as physical sequential (PS), partitioned (PO), or direct (DA). If the data set is processed using absolute rather than relative addresses, you must mark it as unmovable by adding a U to the DSORG parameter (for example, by coding DSORG=PSU). You must specify the data set organization in the DCB macro. In addition, remember these guidelines:

- ▶ When creating a direct data set, the DSORG in the DCB macro must specify PS or PSU and the DD statement must specify DA or DAU.
- ▶ PS is for sequential and extended format DSNTYPE.
- ▶ PO is the data set organization for both PDSEs and PDSs. DSNTYPE is used to distinguish between PDSEs and PDSs.

For more information about data set organization, data set types, and other data set attributes, see *IBM z/OS DFSMS Using data sets*, SC23-6855.

Data set naming

Whenever you allocate a new data set, you (or z/OS) must give the data set a unique name. Usually, the data set name is given as the DSNAME keyword in job control language (JCL).

A data set name can be one name segment, or a series of joined name segments. Each name segment represents a level of qualification. For example, the data set name HARRY.FILE.EXAMPLE.DATA is composed of four name segments. The first name on the left is called the high-level qualifier (HLQ), and the last name on the right is the lowest-level qualifier (LLQ).

Each name segment (qualifier) is 1 to 8 characters, the first of which must be alphabetic (A - Z) or national (# @ \$). The remaining seven characters are either alphabetic, numeric (0 - 9), national, or a hyphen (-). Name segments are separated by a period (.).

Note: Including all name segments and periods, the length of the data set name must not exceed 44 characters. Thus, a maximum of 22 name segments can make up a data set name.

2.3 Data set striping

Striping is a software implementation that distributes sequential data sets across multiple 3390 volumes. Sequential data striping can be used for physical sequential data sets that cause I/O bottlenecks for critical applications. Sequential data striping uses extended-format sequential data sets that SMS can allocate over multiple volumes, preferably on separate channel paths and control units, to improve performance. These data sets must be on 3390 volumes that are on the IBM DS8000 family.

Sequential data striping can reduce the processing time that is required for long-running batch jobs that process large, physical sequential data sets. Smaller sequential data sets can also benefit because of DFSMS's improved buffer management for QSAM and BSAM access methods for striped extended-format sequential data sets.

A *stripe* in DFSMS is the portion of a striped data set that is on a single volume. The records in that portion are not always logically consecutive. The system distributes records among the stripes so that the volumes can be read from or written to simultaneously to gain better performance. Whether the data set is striped is not apparent to the application program.

Data striping distributes data for one data set across multiple SMS-managed DASD volumes, improving I/O performance and reducing batch window. For example, a data set with 28 stripes is distributed across 28 volumes.

You can write striped extended-format sequential data sets with the maximum physical block size for the data set plus the control information required by the access method. The access method writes data on the first volume selected until a track is filled. The next physical blocks are written on the second volume selected until a track is filled, continuing until all volumes selected have been used or no more data exists. Data is written again to selected volumes in this way until the data set has been created. A maximum of 59 stripes can be allocated for a data set. For striped data sets, the maximum number of extents on a volume is 123.

2.3.1 Physical sequential and VSAM data sets

The sustained data rate (SDR) only affects extended-format data sets. Striping allows you to spread data across DASD volumes and controllers. The number of stripes is the number of volumes on which the data set is initially allocated. Striped data sets must be system-managed and must be in an extended format. When no volumes that use striping are available, the data set is allocated as nonstriped with EXT=P specified in the data class. The allocation fails if EXT=R is specified in the data class.

Physical sequential data sets cannot be extended if none of the stripes can be extended. For VSAM data sets, each stripe can be extended to an available candidate volume if extensions fail on the current volume.

Data classes

Only extended-format, SMS-managed data sets can be striped. To achieve that, you can use data class to allocate sequential and VSAM data sets in extended format for the benefits of compression (sequential and VSAM KSDS), striping, and large data set sizes.

Storage groups

SMS calculates the average preference weight of each storage group using the preference weights of the volumes that will be selected if the storage group is selected for allocation. SMS then selects the storage group that contains at least as many primary volumes as the stripe count and has the highest average weight.

If there are no storage groups that meet these criteria, the storage group with the largest number of primary volumes is selected. If multiple storage groups are tied for the largest number of primary volumes, the one with the highest average weight is selected. If there are still multiple storage groups that meet the selection criteria, SMS selects one at random.

For striped data sets, ensure that enough separate paths are available to DASD volumes in the storage group to allow each stripe to be accessible through a separate path. The maximum number of stripes for physical sequential (PS) data sets is 59. For VSAM data sets, the maximum number of stripes is 16. Only sequential or VSAM data sets can be striped.

Striping volume selection

Striping volume selection is very similar to conventional volume selection. Volumes that are eligible for selection are classified as primary and secondary, and assigned a volume preference weight based on preference attributes.

Data set separation

Data set separation allows you to designate groups of data sets in which all SMS-managed data sets within a group are kept separate, on the physical control unit (PCU) level or the volume level, from all the other data sets in the same group.

To use data set separation, you must create a data set separation profile and specify the name of the profile to the base configuration. During allocation, SMS attempts to separate the data sets listed in the profile. A data set separation profile contains at least one data set separation group.

Each data set separation group specifies whether separation is at the PCU or volume level, whether it is required or preferred, and includes a list of data set names to be separated from each other during allocation.

2.4 Accessing your data sets

After your data sets are created, you must be able to locate your data for future use by other programs and applications. To accomplish that, z/OS provides catalogs and volume tables of contents (VTOCs) to store data set related information, and allow a quick search interface.

Several structures can be searched during a locate request on z/OS systems, including these:

VTOC	The volume table of contents is a sequential data set located in each DASD volume that describes the data set contents of this volume. The VTOC is used to find empty space for new allocations and to locate non-VSAM data sets.
Master catalog	This structure is where all catalog searches begin. The master catalog contains information about system data sets, or points to the catalog that has the information about the requested data set.
User catalog	A user catalog is the next step in the search hierarchy. After the master catalog points to a user catalog related to an Alias, the user catalog retrieves the data set location information.
Alias	A special entry in the master catalog that points to a user catalog that coincides with the HLQ of a data set name. The alias is used to find in which user catalog the data set location information exists. That means that the data set with this HLQ is cataloged in that user catalog.

Figure 2-2 shows the difference between referencing a cataloged and an uncataloged data set on JCL.

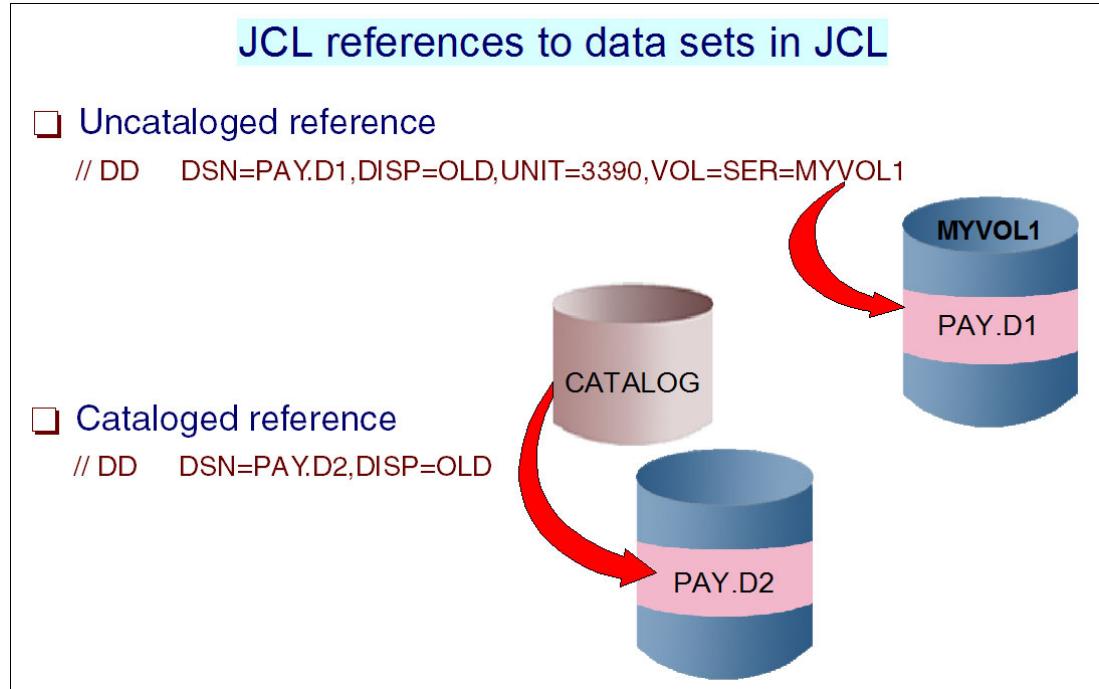


Figure 2-2 Referencing cataloged and uncataloged data sets in JCL

2.4.1 Accessing cataloged data sets

When an existing data set is cataloged, z/OS obtains unit and volume information from the catalog by using the LOCATE macro service. When z/OS tries to locate an existing data set, the following sequence takes place:

1. The master catalog is examined:
 - If it has the searched data set name, the volume information is picked up and the volume VTOC is used to locate the data set in the specified volume. If the data set is not found in the catalog, there is no alias for the data set HLQ, or the data set does not physically exist, the locate returns a not found error.
 - If the HLQ is a defined alias in the master catalog, the user catalog is searched. If the data set name is found, processing proceeds as in a master catalog find. If the data set is not found in the catalog, or the data set does not physically exist, the locate returns a not found error.
2. The requesting program accesses the data set. As you can imagine, it is impossible to keep track of the location of millions of data sets without the catalog concept.

For detailed information about catalogs, see Chapter 6, “Catalogs” on page 103.

2.4.2 Accessing uncataloged data sets

When your existing data set is not cataloged, you *must* know in advance its volume location and specify it in your JCL. This specification can be done through the UNIT and VOL=SER.

See *z/OS MVS JCL Reference*, SA22-7597 for information about UNIT and VOL parameters.

Note: Avoid having uncataloged data sets in your installation because uncataloged data sets can cause problems with duplicate data and possible incorrect data set processing.

2.5 VTOC and DSCBs

The VTOC is a data set that describes the contents of the DASD volume on which it resides. It is a contiguous data set. That is, it is in a single extent on the volume, starts after cylinder 0, track 0, and ends before track 65,535.

Each VTOC record contains information about the volume or data sets on the volume. These records are called data set control block (DSCBs) and are described next.

2.5.1 Data set control block

The VTOC is composed of 140-byte DSCBs that point to data sets currently residing on the volume, or to contiguous, unassigned (free) tracks on the volume (depending on the DSCB type).

DSCBs also describe the VTOC itself. The common VTOC access facility (CVAF) routines automatically construct a DSCB when space is requested for a data set on the volume. Each data set on a DASD volume has one or more DSCBs (depending on its number of extents) describing space allocation and other control information such as operating system data, device-dependent information, and data set characteristics. There are nine kinds of DSCBs, each with a different purpose and a different format number.

The first record in every VTOC is the VTOC DSCB (format-4). The record describes the device, the volume that the data set is on, the volume attributes, and the size and contents of the VTOC data set itself. The next DSCB in the VTOC data set is a free-space DSCB (format-5) that describes the unassigned (free) space in the full volume. The function of various DSCBs depends on whether an optional Index VTOC is allocated in the volume. Index VTOC is a sort of B-tree that helps make the search in VTOC faster.

For more information about VTOC and DSCB allocation and usage, see *z/OS DFSMSdfp Advanced Services*, SC23-6861.

2.5.2 VTOC index

The VTOC index enhances the performance of VTOC access. The VTOC index is a physical-sequential data set on the same volume as the related VTOC, created by the ICKDSF utility program. It consists of an index of data set names in format-1 DSCBs contained in the VTOC and volume free space information.

Important: An SMS-managed volume *requires* an indexed VTOC. Otherwise, the VTOC index is highly preferred. For more information about SMS-managed volumes, see *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407.

If the system detects a logical or physical error in a VTOC index, the system disables further access to the index from all systems that might be sharing the volume. In that case, the VTOC remains usable but with possibly degraded performance.

If a VTOC index becomes disabled, you can rebuild the index without taking the volume offline to any system. All systems can continue to use that volume without interruption to other applications, except for a brief pause during the index rebuild. After the system rebuilds the VTOC index, it automatically reenables the index on each system that has access to it.

2.5.3 Creating VTOC and VTOC Index

Before you can use a DASD, you must initialize your volume, creating a volume label, a VTOC, and preferably a VTOC index. ICKDSF is a program that you can use to perform the functions needed for the initialization, installation, use, and maintenance of DASD volumes. You can also use it to perform service functions, error detection, and media maintenance. However, due to the virtualization of the volumes there is no need for running media maintenance.

You use the INIT command to initialize volumes. The INIT command writes a volume label (on cylinder 0, track 0) and a VTOC on the device for use by z/OS. It reserves and formats tracks for the VTOC at the location specified by the user and for the number of tracks specified. If no location is specified, tracks are reserved at the default location.

If the volume is SMS-managed, the STORAGEGROUP option must be declared to keep such information (SMS-managed) in a format-4 DSCB.

For more information about how to use ICKDSF to maintain your DASD devices, see *ICKDSF R17 User's Guide*, GC35-0033.



Extended address volumes

z/OS V1R10 introduced extended address volume (EAV), which allowed direct access storage devices (DASD) storage volumes to be larger than 65,520 cylinders. The increased volume capacity relieves a major system constraint, which is the maximum number of units you can define to a system at any single time.

This chapter introduces you to EAV, and brief you about its architecture, usage, and restrictions. It includes the following sections:

- ▶ Traditional DASD capacity
- ▶ Extended access volumes
- ▶ EAV and IGDSMSxx parmlib member
- ▶ EATTR attribute
- ▶ Migration assistance tracker

3.1 Traditional DASD capacity

In the past decade, as processing power has dramatically increased, great care and appropriate solutions have been deployed so that the amount of data that is directly accessible can be kept proportionally equivalent. Over the years, DASD volumes have increased in size by increasing the number of cylinders and thus GB capacity.

However, the existing track addressing architecture has limited growth to relatively small GB capacity volumes. This limitation has placed increasing strain on the 4-digit device number limit and the number of unit control blocks (UCBs) that can be defined. The largest available volume is one with 65,520 cylinders or approximately 54 GB, as shown in Figure 3-1.

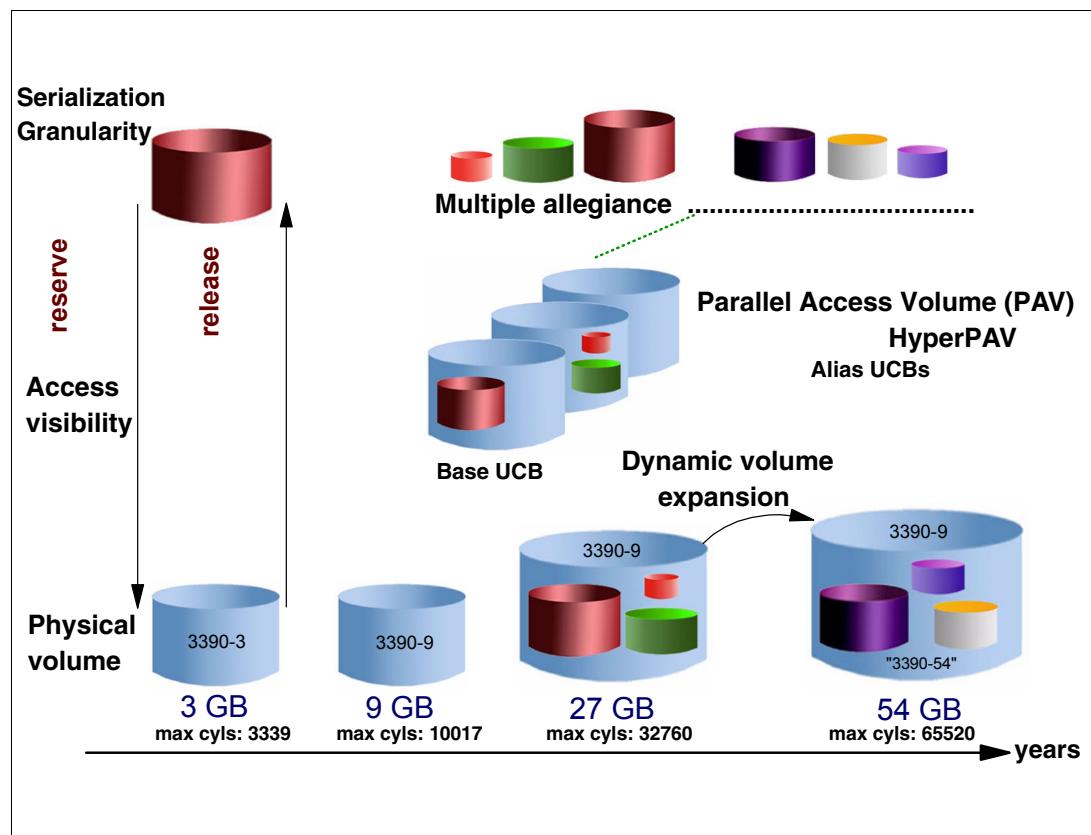


Figure 3-1 Traditional DASD volumes sizes and serialization

Rapid data growth on the z/OS platform is leading to a critical problem for various clients. As a result, this limitation is becoming a real constraint on growing data on z/OS. Business resilience solutions (GDPS, IBM HyperSwap®, and more) that provide continuous availability are also driving this constraint.

3.1.1 Serialization granularity

Since the 1960s, shared DASD can be serialized through a sequence of RESERVE/RELEASE CCWs that are today under the control of GRS, as shown in 3.1, "Traditional DASD capacity" on page 24. This was a useful mechanism while the volume of data so serialized (the granularity) was not too large. But whenever such a device grew to contain too much data, bottlenecks became an issue.

3.1.2 DASD virtual visibility

Traditional IBM S/390® architecture does not allow more than one I/O operation to the same S/390 device because such devices can only handle, physically, one I/O operation at a time. However, in modern DASD subsystems such as DS8000 family, the device is only a logical view. The contents of this logical device are spread in Head-Disk Assembly (HDA) Redundant Array of Independent Disks (RAID) arrays and in caches. Therefore, it is technically possible to have more than one I/O operation towards the same logical device.

Changes have been made in z/OS (in IOS code), in the channel subsystem (System Assist Processor or SAP), and in DASD controllers to allow more than one I/O operation on the same logical device. This is called parallel I/O, and it is available in two types:

- ▶ Multiple allegiance
- ▶ Parallel access volume (PAV)

This relief builds on prior technologies that were implemented in part to help reduce the pressure on running out of device numbers. These include PAV and HyperPAV. PAV alias UCBs can be placed in an alternate subchannel set. HyperPAV reduces the number of alias UCBs over traditional PAVs and provides the I/O throughput required.

3.1.3 Multiple allegiance

Multiple allegiance (MA) was introduced to alleviate the following constraint. It allows serialization on a limited amount of data within a specific DASD volume, which leads to the possibility of having several (non-overlapping) serializations held at the same time on the same DASD volume. This is a useful mechanism on which any extension of the DASD volume addressing scheme can rely.

In other terms, multiple allegiance provides finer (than RESERVE/RELEASE) granularity for serializing data on a volume. It gives the capability to support I/O requests from multiple systems, one per system, to be concurrently active against the same logical volume, if they do not conflict with each other. Conflicts occur when two or more I/O requests require access to overlapping extents (an *extent* is a contiguous range of tracks) on the volume, and at least one of the I/O requests involves writing of data.

Requests involving writing of data can run concurrently with other requests while they operate on non-overlapping extents on the volume. Conflicting requests are internally queued in the DASD controller. Read requests can always run concurrently regardless of their extents.

Without the MA capability, DS8000 generates a busy for the volume whenever one of the systems issues a request against the volume, thereby causing the I/O requests to be queued within the channel subsystem (CSS). However, this concurrency can be achieved as long as no data accessed by one channel program can be altered through the actions of another channel program.

3.1.4 Parallel access volume

z/OS system IOS maps a device in a UCB. Traditionally this I/O device does not support concurrency, being treated as a single resource, serially used. High I/O activity towards the same device can adversely affect performance. This contention is worst for large volumes with many small data sets. The symptom displayed is extended IOSQ time, where the I/O request is queued in the UCB. z/OS cannot attempt to start more than one I/O operation at a time to the device.

The DS800 family supports concurrent data transfer operations to or from 3390/3380 devices residing on the same system. A device (volume) accessed in this way is called a PAV.

PAV exploitation requires both software enablement and an optional feature on your controller. PAV support must be installed on each controller. It enables the issuing of multiple channel programs to a volume from a single system, and allows simultaneous access to the logical volume by multiple users or jobs. Reads, and writes to other extents, can be satisfied simultaneously.

PAV benefits

Workloads that are most likely to benefit from PAV functions being available include:

- ▶ Volumes with many concurrently open data sets, such as volumes in a work pool
- ▶ Volumes that have a high read to write ratio per extent
- ▶ Volumes reporting high IOSQ times

3.1.5 HyperPAV feature

HyperPAV is an enhancement to regular PAV. It removes the base and alias binding that was necessary on PAV, reducing the required number of alias UCBs to provide parallel access to volumes. HyperPAV provides these advantages:

- ▶ A pool of UCB aliases for use by z/OS, reducing the required number of aliases.
- ▶ As each application I/O is requested, if the base volume is busy with another I/O:
 - z/OS selects a free alias from the pool, quickly binds the alias device to the base device, and starts the I/O.
 - When the I/O completes, the alias device is used for another I/O on the LSS or is returned to the free alias pool.

If too many I/Os are started simultaneously:

- ▶ z/OS queues the I/Os at the LSS level.
- ▶ When an exposure frees up that can be used for queued I/Os, they are started.
- ▶ Queued I/O is done within assigned I/O priority.

For each z/OS image within the sysplex, aliases are used independently. WLM is not involved in alias movement so it does not need to collect information to manage HyperPAV aliases.

Benefits of HyperPAV

HyperPAV has been designed to provide an even more efficient PAV function. When implementing larger volumes, it provides a way to scale I/O rates without the need for additional PAV alias definitions. HyperPAV uses IBM FICON® architecture to reduce overhead, improve addressing efficiencies, and provide storage capacity and performance improvements, as follows:

- ▶ More dynamic assignment of PAV aliases improves efficiency.
- ▶ The number of PAV aliases that are needed might be reduced, taking fewer from the 64 K. device limitation and leaving more storage for capacity use.

Figure 3-2 shows how pools of alias UCBs are used by z/OS images to satisfy I/O request by applications pointing to data sets on the same base volume.

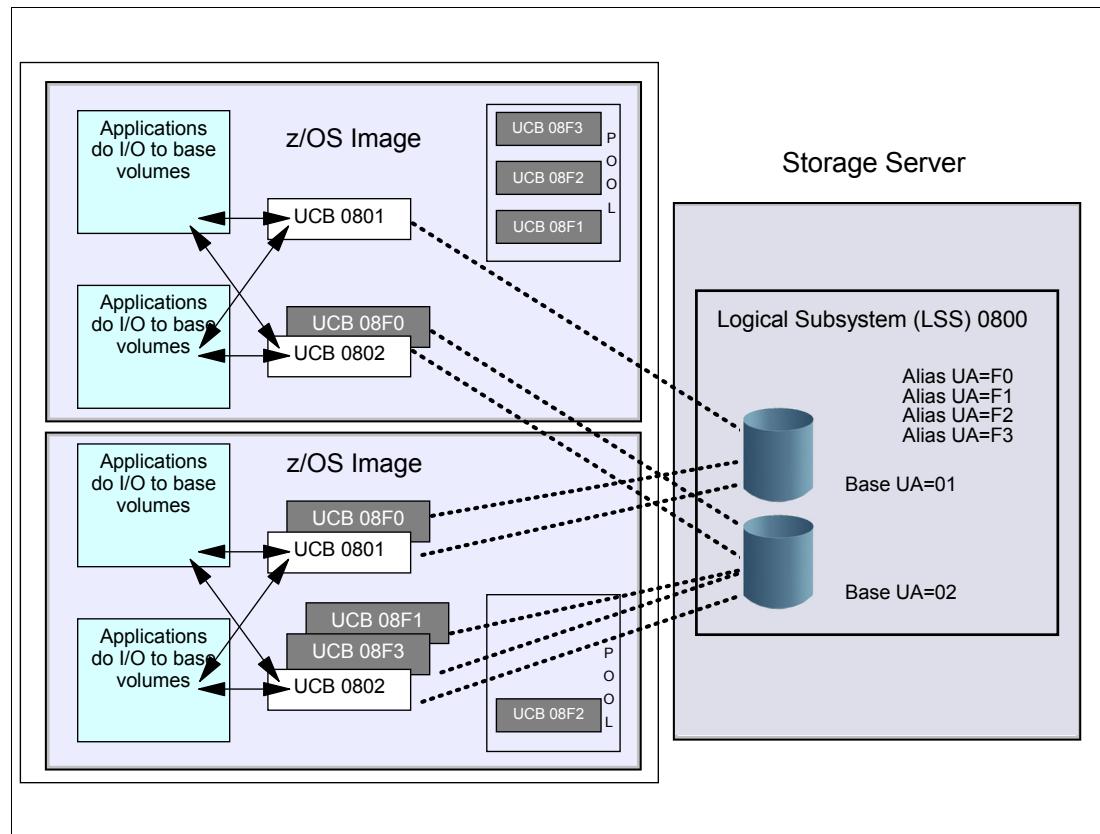


Figure 3-2 HyperPAV implementation using a pool of aliases

3.2 Extended access volumes

As customer data continues to grow rapidly, the amount of storage available to hold this data also needs to increase. This change brings back the time when the capacity of the volumes and the number of UCBs available were close to the limit, and further improvement of data storage was required.

EAV was a breakthrough in DASD storage capacity, allowing volumes to greatly expand the 65,520 cylinders limitation for each volume.

The benefit of this support is that the amount of z/OS addressable disk storage is further significantly increased. This support provides relief for customers that are approaching the 4-digit device number limit by defining fewer, larger volumes to allocate data. This approach is especially helpful for customers that use DASD copy and replication services, where the limit of UCBs available can become a constraint.

3.2.1 3390 Model A

To define larger volumes, you need to define your 3390 as model A in your DASD controller. This new DASD model allows you to define a volume with any number of cylinders in the range 1 - 1,182,006 cylinders.

Note the following points regarding extended address volumes:

- ▶ Only 3390 Model A devices can be EAV.
- ▶ EAV is supported starting in z/OS V1R10 and further releases.
- ▶ The size is limited to 1 tB (1,182,006 cylinders) in z/OS V2R3

Note: With the 3390 Model A, the model A refers to the model configured in the DS8000. It has no association with the 3390A notation in HCD that indicates a PAV-alias UCB in the z/OS operating system. The Model "A" was chosen so that it did not imply a particular device size as previous models 3390-3 and 3390-9 did.

3.2.2 EAV volumes architecture

To address the space above cylinder 65,520, you must change how tracks are addressed in the Volume Table of Contents (VTOC) records, and the minimum allocation values. The specifics for each change are described next. For more detailed information about EAV, see *DFSMS: Extended Address Volume*, REDP-5364.

Multicylinder unit

A multicylinder unit (MCU) is a fixed unit of disk space that is larger than a cylinder. Currently, on an EAV volume, a multicylinder unit is 21 cylinders and the number of the first cylinder in each multicylinder unit is a multiple of 21.

The cylinder-managed space is space on the volume that is managed only in multicylinder units. Cylinder-managed space begins at cylinder address 65,520. Each data set occupies an integral multiple of multicylinder units. Space requests targeted for the cylinder-managed space are rounded up to the next multicylinder unit. The cylinder-managed space only exists on EAV volumes.

The 21-cylinder value for the MCU is the smallest unit that can map out the largest possible EAV volume and stay within the index architecture with a block size of 8,192 bytes.

Additionally:

- ▶ It is also a value that divides evenly into the 1 GB storage segments of an IBM DS8000.
- ▶ These 1 GB segments are the allocation unit in the IBM DS8000 and are equivalent to 1,113 cylinders.
- ▶ These segments are allocated in multiples of 1,113 cylinders starting at cylinder 65,520.

One of the more important EAV design points is that IBM maintains its commitment to customers that the 3390 track format and image size, and tracks per cylinders, will remain the same as the previous 3390 model devices. An application using data sets on an EAV is comparable to how it runs today on 3390 "numerics" kind of models. The extended address volume has two managed spaces:

- ▶ The track-managed space
- ▶ The cylinder-managed space

Cylinder-managed space

The cylinder-managed space is the space on the volume that is managed only in MCUs. Cylinder-managed space begins at cylinder address 65,520. Each data set occupies an integral multiple of multicylinder units. Space requests targeted for the cylinder-managed space are rounded up to the next multicylinder unit. The cylinder-managed space exists only on EAV volumes. A data set allocated in cylinder-managed space might have its requested space quantity rounded up to the next MCU.

Data sets allocated in cylinder-managed space are described with a new type of data set control blocks (DSCBs) in the VTOC. Tracks allocated in this space is also addressed using the new track address. Existing programs that are not changed will not recognize these new DSCBs and therefore will be protected from seeing how the tracks in cylinder-managed space are addressed. For more detailed information about EAV and cylinder-managed space, see *DFSMS: Extended Address Volume*, REDP-5364.

Track-managed space

The track-managed space is the space on a volume that is managed in track and cylinder increments. All volumes today have track-managed space. Track-managed space ends at cylinder number 65,519. Each data set occupies an integral multiple of tracks. The track-managed space allows existing programs and physical migration products to continue to work. Physical copies can be done from a non-EAV to an EAV and have those data sets accessible.

New VTOC for volume expansion

When a volume is dynamically expanded, the VTOC and VTOC index must be reformatted to map the additional space. This process is done automatically by z/OS, unless DISABLE(REFVTOC) is specified in your DEVSUPxx member.

VTOC index

The VTOC index enhances the performance of VTOC access. The VTOC index is a physical-sequential data set on the same volume as the related VTOC. It consists of an index of data set names in format-1 DSCBs contained in the VTOC and volume free space information.

An SMS-managed volume requires an indexed VTOC. Otherwise, the VTOC index is highly preferred. For more detailed information about VTOC and VTOC Index information related to EAV volumes, see *DFSMS: Extended Address Volume*, REDP-5364.

3.2.3 EAS eligible data sets

EAS-eligible data sets are those that can be allocated in the extended addressing space, which is the area on an EAV located above the first 65,520 cylinders. This is sometimes referred to as cylinder-managed space. All of the following data sets can be allocated in the base addressing space of an EAV:

- ▶ SMS-managed VSAM (all types)
 - With control area (CA) sizes of 1, 3, 5, 7, 9, or 15 tracks
- ▶ Non-SMS VSAM (all types)
 - With control area (CA) sizes of 1, 3, 5, 7, 9, or 15 tracks
- ▶ zFS data sets (which are VSAM LS)
- ▶ Database (IBM DB2, IBM IMS™) use of VSAM
- ▶ Basic, Large, or Extended-format sequential data sets
- ▶ Partitioned data set (PDS) and partitioned data set extended (PDSE)
- ▶ Catalog (VSAM volume data set (VVDS) and basic catalog structure (BCS))

For a detailed list of data sets supported on EAS space, see *DFSMS: Extended Address Volume*, REDP-5364.

3.2.4 EAS non-eligible data sets

An extended addressing space (EAS)-ineligible data set is a data set that can exist on an EAV but is not eligible to have extents (through Create or Extend) in the cylinder-managed space. The following are the exceptions to EAS eligibility:

- ▶ VTOC (continues to be restricted to within the first 64 K-1 tracks)
- ▶ VTOC index
- ▶ Page data sets
- ▶ VSAM data sets with embed or keyrange attributes
- ▶ VSAM data sets with incompatible CA sizes
- ▶ Hierarchical file system (HFS)
- ▶ Extended Remote Copy (XRC) control, master, or cluster non-VSAM data sets.

For a detailed list of data sets unsupported on EAS space, see *DFSMS: Extended Address Volume*, REDP-5364.

Note: In a future release, various of these data sets might become EAS-eligible. All data set types, even those listed here, can be allocated in the track-managed space on a device with cylinder-managed space on an EAV volume. Eligible EAS data sets can be created and extended anywhere on an EAV. Data sets that are not eligible for EAS processing can only be created or extended in the track-managed portions of the volume.

3.2.5 Dynamic volume expansion

It is possible to grow an existing volume with the DS8000 family with dynamic volume expansion (DVE). Previously, it was necessary to use migration utilities that require an additional volume for each volume that is being expanded and require the data to be moved.

A logical volume can be increased in size while the volume remains online to host systems for the following types of volumes:

- ▶ 3390 model 3 to 3390 model 9
- ▶ 3390 model 9 to EAV volume sizes

Dynamic volume expansion can be used to expand volumes beyond 65,520 cylinders without moving data or causing an application outage.

Dynamic volume expansion is performed by the DS8000 Storage Manager and can be requested using its web GUI or through command line. 3390 volumes can be increased in size, for example from a 3390 model 3 to a model 9 or a model 9 to a model A (EAV). z/OS V1R11 introduces an interface that can be used to make requests for dynamic volume expansion of a 3390 volume on a DS8000 from the system.

Note: For the dynamic volume expansion function, volumes cannot be in Copy Services relationships (point-in-time copy, FlashCopy SE, Metro Mirror, Global Mirror, Metro/Global Mirror, and z/OS Global Mirror) during expansion.

For more information about how to use DVE, see *DFSMS: Extended Address Volume*, REDP-5364.

3.3 EAV and IGDSMSxx parmlib member

Two SMS IGDSMSxx parmlib member parameters need to be set before you can use extended address volumes with SMS for volume selection:

- ▶ USEEAV
- ▶ BreakPointValue

3.3.1 IGDSMSxx USEEAV parm

USEEAV(YESINO) specifies, at the system level, whether SMS can select an extended address volume during volume selection processing. This check applies to new allocations and when extending data sets to a new volume.

USEEAV(YESINO)

- YES** This setting means that extended address volumes can be used to allocate new data sets or to extend existing data sets to new volumes.
- NO** (Default) This setting means that SMS does not select any extended address volumes during volume selection. Note that data sets might still exist on extended address volumes in either the track-managed or cylinder-managed space of the volume.

Note: You can use the SETSMS command to change the setting of USEEAV without having to restart the system. This modified setting is in effect until the next IPL, when it reverts to the value specified in the IGDSMSxx member of PARMLIB.

To make the setting change permanent, you must alter the value in SYS1.PARMLIB. The syntax of the operator command is:

```
SETSMS USEEAV(YES|NO)
```

SMS requests will not use EAV volumes if the USEEAV setting in the IGDSMSxx parmlib member is set to NO.

Specific allocation requests are failed. For non-specific allocation requests (UNIT=SYSDA), EAV volumes are not selected. Messages indicating no space available are returned when non-EAV volumes are not available.

For non-EAS eligible data sets, all volumes (EAV and non-EAV) are equally preferred (or they have no preference). This is the same as today, with the exception that extended address volumes are rejected when the USEEAV parmlib value is set to NO.

3.3.2 IGDSMSxx BreakPointValue

How the system determines which managed space is to be used is based on the derived BreakPointValue for the target volume of the allocation. EAV volumes are preferred over non-EAV volumes where space is greater and or equal to BreakPointValue (there is no EAV volume preference where space is less than BreakPointValue).

The BreakPointValue (0- 65520) is used by SMS in making volume selection decisions and later by DADSM. If the allocation request is less than the BreakPointValue, the system prefers to satisfy the request from free space available from the track-managed space.

For an extended address volume, the system and storage group use of the BreakPointValue helps direct disk space requests to cylinder-managed or track-managed space. The breakpoint value is expressed in cylinders and is used as follows:

- ▶ When the size of a disk space request is the breakpoint value or more, the system prefers to use the cylinder-managed space for that extent. This rule applies to each request for primary or secondary space for data sets that are eligible for the cylinder-managed space.
- ▶ If cylinder-managed space is insufficient, the system uses the track-managed space or uses both types of spaces.
- ▶ When the size of a disk space request is less than the breakpoint value, the system prefers to use the track-managed space.

If space is insufficient, the system uses the cylinder-managed space or uses both types of space.

Note: The BreakPointValue value is only used to direct placement on an extended address volume.

If you do not specify the BreakPointValue parm in IGDSMSxx member, it defaults to 10 cylinders.

Note: You can use the SETSMS command to change the setting of BreakPointValue without having to restart the system. This modified setting is in effect until the next IPL when it reverts to the value specified in the IGDSMSxx member of PARMLIB. To make the setting change permanent, you must alter the value in SYS1.PARMLIB. The syntax of the operator command is:

```
SETSMS BreakPointValue(0-65520)
```

3.4 EATTR attribute

The support for extended-format sequential data sets includes the EATTR attribute. This attribute has been added for all data set types to allow a user to control whether a data set can have extended attribute DSCBs, which controls whether it can be allocated in the EAS.

EAS-eligible data sets are defined to be those that can be allocated in the extended addressing space and have extended attributes. This is sometimes referred to as cylinder-managed space.

The EATTR data set level attribute specifies whether a data set can have extended attributes (Format 8 and 9 DSCBs) and optionally be in EAS on an EAV. Valid values for the EATTR are NO and OPT:

EATTR = OPT Extended attributes are optional. The data set can have extended attributes and are in EAS. This is the default value for VSAM data sets if EATTR(OPT) is not specified.

EATTR = NO No extended attributes. The data set cannot have extended attributes (format-8 and format-9 DSCBs) or be in EAS. This is the default value for non-VSAM data sets. This setting is equivalent to specifying EATTR=NO on the job control language (JCL) and is applicable to all data set types.

Although data sets with EATTR=OPT are eligible to EAS, they can still be allocated in track-managed devices, so first plan to implement EATTR=OPT to your data sets, and implement EAV volumes later.

For more information about the EATTR attribute, see *z/OS DFSMS Access Method Services Commands*, SC23-6846.

3.5 Migration assistance tracker

The EAV migration assistance tracker can help you find programs that you might need to change if you want to support EAVs. The EAV migration assistance tracker is an extension of the console ID tracking facility. Programs identified in this phase of migration assistance tracking will continue to fail if the system service is issued for an EAV and you do not specify the EADSCB=OK keyword for them.

DFSMS provides an EAV migration assistance tracker program. The EAV migration assistance tracker helps you to do the following tasks:

- ▶ Identify select systems services by job and program name, where the starting programs might require analysis for changes to use new services. The program calls are identified as informational instances for possible migration actions. They are not considered errors because the services return valid information.
- ▶ Identify possible instances of improper use of returned information in programs, such as parsing 28-bit cylinder numbers in output as 16-bit cylinder numbers. These instances are identified as warnings.
- ▶ Identify instances of programs that will either fail or run with an informational message if they run on an EAV. These are identified as programs in error. The migration assistance tracker flags programs with the following functions: When the target volume of the operation is non-EAV, and the function started did not specify the EADSCB=OK keyword.

Note: Being listed in the tracker report does not imply that there is a problem. It is simply a way to help you determine what to examine to see whether it needs to be changed.

For more information about how to use migration assistance tracker, see *z/OS DFSMSdfp Advanced Services*, SC23-6861.



Storage management software

DFSMS is an exclusive element of the z/OS operating system that automatically manages data from creation to expiration. This chapter covers the following topics:

- ▶ The DFSMS utility programs to assist you in organizing and maintaining data
- ▶ The major DFSMS access methods
- ▶ The data set organizations
- ▶ An overview of the elements that comprise DFSMS:
 - DFSMSdfp, a base element of z/OS
 - DFSMSdss, an optional feature of z/OS
 - DFSMShsm, an optional feature of z/OS
 - DFSMSrmm, an optional feature of z/OS
 - z/OS DFSORT

This chapter includes the following sections:

- ▶ Overview of DFSMSdfp utilities
- ▶ DFSMSdfp access methods
- ▶ Access method services (IDCAMS)
- ▶ Virtual Storage Access Method
- ▶ Data set separation
- ▶ Data Facility sort
- ▶ Data Set Services (DFSMSdss)
- ▶ Hierarchical Storage Manager
- ▶ Removable media manager (DFSMSrmm)

4.1 Overview of DFSMSdfp utilities

Utilities are programs that perform commonly needed functions. DFSMS provides utility programs to assist you in organizing and maintaining data. There are system and data set utility programs that are controlled by job control language (JCL), and utility control statements.

This section is intended to give you only an overview about what system utilities and data set utilities are, along with their name and a brief description of their purpose. For more details and to help you find the program that performs the function you need, see “Guide to Utility Program Functions” in *z/OS DFSMSdfp Utilities*, SC26-7414.

4.1.1 System utility programs

System utility programs are used to list or change information related to data sets and volumes, such as data set names, catalog entries, and volume labels. Most functions that system utility programs can perform are accomplished more efficiently with other programs, such as IDCAMS, ISPF/PDF, ISMF, or DFSMSrmm.

Table 4-1 lists and describes system utilities that are provided by DFSMSdfp.

Table 4-1 System utility programs

System utility	Alternate program	Purpose
IEHINITT	DFSMSrmm EDGINERS	Write standard labels on tape volumes.
IEHLIST	ISMF, PDF 3.4	List system control data.
IEHMOVE	DFSMSdss, IEBCOPY	Move or copy collections of data.
IEHPROGM	Access Method Services, PDF 3.2	Build and maintain system control data.
IFHSTATR	DFSMSrmm, EREP	Select, format, and write information about tape errors from the IFASMFDP tape.

4.1.2 Data set utility programs

You can use data set utility programs to reorganize, change, or compare data at the data set or record level. These programs are controlled by JCL statements and utility control statements.

These utilities allow you to manipulate partitioned, sequential, or indexed sequential data sets, or PDSEs, which are provided as input to the programs. You can manipulate data ranging from fields within a logical record to entire data sets. The data set utilities included in this section cannot be used with VSAM data sets. You use the IDCAMS utility to manipulate VSAM data set. For more information, see 4.3.2, “Starting the IDCAMS utility program” on page 40.

Table 4-2 lists data set utility programs and their use.

Table 4-2 Data set utility programs

Data set utility	Use
IEBCOMPR	Compare records in sequential or partitioned data sets, or PDSEs.
IEBCOPY	Copy, compress, or merge partitioned data sets or PDSEs; add RLD count information to load modules; select or exclude specified members in a copy operation; rename or replace selected members of partitioned data sets or PDSEs.
IEBDG	Create a test data set consisting of patterned data.
IEBEDIT	Selectively copy job steps and their associated JOB statements.
IEBGENER or ICEGENER	Copy records from a sequential data set, or convert a data set from sequential organization to partitioned organization.
IEBIMAGE	Modify, print, or link modules for use with the IBM 3800 Printing Subsystem, the IBM 3262 Model 5, or the 4284 printer.
IEBPTPCH	Print or punch records of a sequential or partitioned data set.
IEBUPDTE	Incorporate changes to sequential or partitioned data sets, or PDSEs.

4.2 DFSMSdfp access methods

An *access method* is a friendly program interface between programs and their data. It is in charge of interfacing with Input/Output Supervisor (IOS), the z/OS code that starts the I/O operation. An access method makes the physical organization of data transparent to you by using these techniques:

- ▶ Managing data buffers
- ▶ Blocking and deblocking logical records into physical blocks
- ▶ Synchronizing your task and the I/O operation (wait/post mechanism)
- ▶ Writing the channel program
- ▶ Optimizing the performance characteristics of the control unit (such as caching and data striping)
- ▶ Compressing and decompressing I/O data
- ▶ Running software error recovery

In contrast to other platforms, z/OS supports several types of access methods and data organizations.

An access method defines the organization by which the data is stored and retrieved. DFSMS access methods have their own data set structures for organizing data, macros, and utilities to define and process data sets. It is an application choice, depending on the type of access (sequential or random), to allow or disallow insertions and deletions, to pick up the most adequate access method for its data.

Access methods are identified primarily by the data set organization to which they apply. For example, you can use the basic sequential access method (BSAM) with sequential data sets.

However, there are times when an access method identified with one data organization type can be used to process a data set organized in a different manner. For example, a sequential data set (not an extended format data set) created by using BSAM can be processed by the basic direct access method (BDAM), and vice versa.

4.2.1 Basic sequential access method

BSAM arranges logical records sequentially in the order in which they are entered. A data set that has this organization is a sequential data set. Blocking, deblocking and the I/O synchronization is done by the application program. This is basic access. You can use BSAM with the following data types:

- ▶ Sequential data sets
- ▶ Extended-format data sets
- ▶ z/OS UNIX files

4.2.2 Basic direct access method

BDAM arranges records in any sequence that your program indicates, and retrieves records by actual or relative address. If you do not know the exact location of a record, you can specify a point in the data set to begin a search for the record. Data sets organized this way are called *direct data sets*.

Optionally, BDAM uses hardware keys. Hardware keys are less efficient than the optional software keys in VSAM KSDS.

Note: Because BDAM tends to require the use of device-dependent code, it is not a recommended access method. In addition, using keys is much less efficient than in VSAM. BDAM is supported by DFSMS only to enable compatibility with other IBM operating systems.

4.2.3 Basic partitioned access method

Basic partitioned access method (BPAM) arranges records as members of a PDS or a PDSE on DASD. You can use BPAM to view a UNIX directory and its files as though it were a PDS. You can view each PDS, PDSE, or UNIX member sequentially with BSAM or QSAM. A PDS or PDSE includes a directory that relates member names to locations within the data set. Use the PDS, PDSE, or UNIX directory to retrieve individual members.

A member is a sequential file that is contained in the PDS or PDSE data set. When members contain load modules (executable code in PDS) or program objects (executable code in PDSE), the directory contains program attributes that are required to load and rebind the member. Although UNIX files can contain program objects, program management does not access UNIX files through BPAM.

4.2.4 Queued sequential access method (QSAM)

QSAM arranges logical records sequentially in the order that they are entered to form sequential data sets, which are the same as those data sets that BSAM creates. The system organizes records with other records. QSAM anticipates the need for records based on their order. To improve performance, QSAM reads these records into main storage before they are requested. This is called *queued access*. QSAM blocks and deblocks logical records into physical blocks. QSAM also ensures the synchronization between the task and the I/O operation.

You can use QSAM with the following data types:

- ▶ Sequential data sets
- ▶ Basic format sequential data sets
- ▶ Large format sequential data sets
- ▶ Extended-format data sets
- ▶ z/OS UNIX files

4.2.5 Object access method

Object access method (OAM) processes very large named byte streams (objects) that have no record boundary or other internal orientation as image data. These objects can be recorded in a DB2 database or on an optical storage volume. For information about OAM, see *z/OS DFSMS Object Access Method Application Programmer's Reference*, SC35-0425, and *z/OS DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support*, SC35-0426.

4.2.6 Virtual Storage Access Method

Virtual Storage Access Method (VSAM) is an access method that has several ways of organizing data, depending on the application's needs.

VSAM arranges and retrieves logical records by an index key, relative record number, or relative byte addressing (RBA). A logical record has an RBA, which is the relative byte address of its first byte in relation to the beginning of the data set. VSAM is used for direct, sequential, or skip sequential processing of fixed-length and variable-length records on DASD. VSAM data sets (also named clusters) are always cataloged. There are five types of cluster organization:

- ▶ Entry-sequenced data set (ESDS)

This data set contains records in the order in which they were entered. Records are added to the end of the data set and can be accessed sequentially or randomly through the RBA.

- ▶ Key-sequenced data set (KSDS)

This data set contains records in ascending collating sequence of the contents of a logical record field called key. Records can be accessed by using the contents of the key, or by an RBA.

- ▶ Linear data set (LDS)

This data set contains data that has no record boundaries. Linear data sets contain none of the control information that other VSAM data sets do. Data in Virtual (DIV) is an optional intelligent buffering technique that includes a set of assembler macros that provide buffering access to VSAM linear data sets. For more information about DIV, see 4.4.13, "VSAM: Data-in-virtual" on page 57.

- ▶ Relative record data set (RRDS)

This data set contains logical records in relative record number order. The records can be accessed sequentially or randomly based on this number. There are two types of relative record data sets:

- Fixed-length RRDS: Logical records must be of fixed length.
- Variable-length RRDS: Logical records can vary in length.

A z/OS UNIX file (HFS or zFS) can be accessed as though it were a VSAM ESDS. Although UNIX files are not actually stored as entry-sequenced data sets, the system attempts to simulate the characteristics of such a data set. To identify or access a UNIX file, specify the path that leads to it.

4.3 Access method services (IDCAMS)

You can use the access method services utility (also known as IDCAMS) to establish and maintain catalogs and data sets (VSAM and non-VSAM). It is used mainly to create and manipulate VSAM data sets. IDCAMS has other functions (such as catalog updates), but it is most closely associated with the use of VSAM.

4.3.1 Access method services commands

There are two types of access method services commands:

Functional commands Used to request the actual work (for example, defining a data set or listing a catalog)

Modal commands Allow the conditional execution of functional commands (to make it look like a language)

All access method services commands have the following general structure:

COMMAND parameters ... [terminator]

The command defines the type of service requested, the parameters further describe the service requested, and the terminator indicates the end of the command statement.

Time Sharing Option (TSO) users can use functional commands only. For more information about modal commands, refer to *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

The automatic class selection (ACS) routines (established by your storage administrator) and the associated SMS classes eliminate the need to use many access method services command parameters. The SMS environment is discussed in more detail in Chapter 5, “System-managed storage” on page 79.

4.3.2 Starting the IDCAMS utility program

When you want to use an access method services function, enter a command and specify its parameters. Your request is decoded one command at a time. The appropriate functional routines perform all services that are required by that command.

You can call the access method services program in the following ways:

- ▶ As a job or jobstep
- ▶ From a TSO session
- ▶ From within your own program

TSO users can run access method services functional commands from a TSO session as though they were TSO commands.

For more information, refer to “Invoking Access Method Services from Your Program” in *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

As a job or jobstep

You can use JCL statements to call access method services. PGM=IDCAMS identifies the access method services program, as shown in Figure 4-1.

```
//YOURJOB JOB YOUR INSTALLATION'S JOB=ACCOUNTING DATA
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
*
access method services commands and their parameters
/*
```

Figure 4-1 JCL statements to call IDCAMS

From a TSO session

You can use TSO with VSAM and access method services to:

- ▶ Run access method services commands
- ▶ Run a program to call access method services

Each time that you enter an access method services command as a TSO command, TSO builds the appropriate interface information and calls access method services. You can enter one command at a time. The access method services processes the command completely before TSO lets you continue processing.

To use IDCAMS and certain of its parameters from TSO/E, you must update the IKJTSOxx member of SYS1.PARMLIB. Add IDCAMS to the list of authorized programs (AUTHPGM). For more information, see *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

From within your own program

You can also call the IDCAMS program from within another program and pass the command and its parameters to the IDCAMS program.

4.3.3 IDCAMS functional commands

This section describes the IDCAMS functional commands and their respective functions. Table 4-3 lists and describes the functional commands. For more information about each command, a detailed description, usage, and syntax, see *z/OS DFSMS Access Method Services Commands*, SC23-6846.

Table 4-3 Functional commands

Command	Functions
ALLOCATE	Allocates VSAM and non-VSAM data sets.
ALTER	Alters attributes of data sets, catalogs, tape library entries, and tape volume entries that have already been defined.
BLDINDEX	Builds alternate indexes for existing VSAM base clusters.
CREATE	Creates tape library entries and tape volume entries.
DCOLLECT	Collects data set, volume usage, and migration utility information.
DEFINE ALIAS	Defines an alternate name for a user catalog or a non-VSAM data set.
DEFINE ALTERNATEINDEX	Defines an alternate index for a KSDS or ESDS VSAM data set.

Command	Functions
DEFINE CLUSTER	Creates KSDS, ESDS, RRDS, VRRDS, and linear VSAM data sets.
DEFINE GENERATIONDATAGROUP	Defines a catalog entry for a generation data group (GDG).
DEFINE NONVSAM	Defines a catalog entry for a non-VSAM data set.
DEFINE PAGESPACE	Defines an entry for a page space data set.
DEFINE PATH	Defines a path directly over a base cluster or over an alternate index and its related base cluster.
DEFINE USERCATALOG	Defines a user catalog.
DELETE	Deletes catalogs, VSAM clusters, and non-VSAM data sets.
DIAGNOSE	Scans an integrated catalog facility BCS or a VVDS to validate the data structures and detect structure errors.
EXAMINE	Analyzes and reports the structural consistency of either an index or data component of a KSDS VSAM data set cluster.
EXPORT	Disconnects user catalogs, and exports VSAM clusters and ICF catalog information about the cluster.
EXPORT DISCONNECT	Disconnects a user catalog.
IMPORT	Connects user catalogs, and imports VSAM cluster and its integrated catalog facility (ICF) catalogs information.
IMPORT CONNECT	Connects a user catalog or a volume catalog.
LISTCAT	Lists catalog entries.
PRINT	Used to print VSAM data sets, non-VSAM data sets, and catalogs.
REPRO	Performs the following functions: <ul style="list-style-type: none"> ▶ Copies VSAM and non-VSAM data sets, user catalogs, master catalogs, and volume catalogs. ▶ Splits ICF catalog entries between two catalogs. ▶ Merges ICF catalog entries into another ICF user or master catalog. ▶ Merges tape library catalog entries from one volume catalog into another volume catalog.
SHCDS	Lists SMSVSAM recovery associated with subsystems spheres and controls that allow recovery of a VSAM RLS environment.
VERIFY	Causes a catalog to correctly reflect the end of a data set after an error occurred while closing a VSAM data set. The error might have caused the catalog to be incorrect.

4.3.4 AMS modal commands

With access method services (AMS), you can set up jobs to run a sequence of modal commands with a single invocation of IDCAMS. Modal command execution depends on the success or failure of prior commands.

Using modal commands

With access method services, you can set up jobs to run a sequence of modal commands with a single invocation of IDCAMS. Modal command execution depends on the success or failure of prior commands. The access method services modal commands are used for the conditional execution of functional commands. The following commands are available:

- ▶ The **IF-THEN-ELSE** command sequence controls command execution based on condition codes.
- ▶ The **NULL** command causes the program to take no action.
- ▶ The **DO-END** command sequence specifies more than one functional access method services command and its parameters.
- ▶ The **SET** command resets condition codes.
- ▶ The **CANCEL** command ends processing of the current job step.
- ▶ The **PARM** command chooses diagnostic aids and options for printed output.

Note: These commands cannot be used when access method services are run in TSO. See *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394, for a complete description of the AMS modal commands.

Commonly used single job step command sequences

A sequence of commands commonly used in a single job step includes **EXPORT-DELETE-DEFINE-IMPORT** or **DELETE-DEFINE-BLDINDEX**, as follows:

- ▶ You can specify either a data definition (DD) name or a data set name with these commands.
- ▶ When you refer to a DD name, allocation occurs at job step initiation. The allocation can result in a job failure if a command such as **IMPORT** follows a **EXPORT-DELETE-DEFINE** sequence that changes the location (volser) of the data set. Such failures can occur with either SMS-managed data sets or non-SMS-managed data sets.

Avoiding potential command sequence failures

To avoid potential failures with a modal command sequence in your IDCAMS job, perform either one of the following tasks:

- ▶ Specify the data set name instead of the DD name.
- ▶ Use a separate job step to perform any sequence of commands (for example, REPRO, IMPORT, BLDINDEX, PRINT, or EXAMINE) that follow a DEFINE command.

4.3.5 DFSMS Data Collection Facility (DCOLLECT)

The DFSMS Data Collection Facility (DCOLLECT) is a function of access method services. DCOLLECT collects data in a sequential file that you can use as input to other programs or applications.

The IDCAMS DCOLLECT command collects DASD performance and space occupancy data in a sequential file that you can use as input to other programs or applications.

An installation can use this command to collect information about these items:

- ▶ Active data sets: DCOLLECT provides data about space use and data set attributes and indicators on the selected volumes and storage groups.
- ▶ VSAM data set information: DCOLLECT provides specific information related to VSAM data sets that are on the selected volumes and storage groups.

- ▶ Volumes: DCOLLECT provides statistics and information about volumes that are selected for collection.
- ▶ Inactive data: DCOLLECT produces output for DFSMShsm-managed data (inactive data management), which includes both migrated and backed up data sets:
 - Migrated data sets: DCOLLECT provides information about space utilization and data set attributes for data sets migrated by DFSMShsm.
 - Backed up data sets: DCOLLECT provides information about space utilization and data set attributes for every version of a data set backed up by DFSMShsm.
- ▶ Capacity planning: Capacity planning for DFSMShsm-managed data (inactive data management) includes the collection of both DASD and tape capacity planning:
 - DASD Capacity Planning: DCOLLECT provides information and statistics for volumes managed by DFSMShsm (ML0 and ML1).
 - Tape Capacity Planning: DCOLLECT provides statistics for tapes managed by DFSMShsm.
- ▶ SMS configuration information: DCOLLECT provides information about the SMS configurations. The information can be from either an active control data set (ACDS) or a source control data set (SCDS), or the active configuration.

Data is gathered from the VTOC, VVDS, and DFSMShsm control data sets for both managed and non-managed storage. ISMF provides the option to build the JCL necessary to run **DCOLLECT**.

With the sample JCL shown in Figure 4-2 you can gather information about all volumes belonging to the storage group STGGP001.

```
//COLLECT2 JOB    ...
//STEP1   EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=A
//OUTDS   DD  DSN=USER.DCOLLECT.OUTPUT,
//           STORCLAS=LARGE,
//           DSORG=PS,
//           DCB=(RECFM=VB,LRECL=644,BLKSIZE=0),
//           SPACE=(1,(100,100)),AVGREC=K,
//           DISP=(NEW,CATLG,KEEP)
//SYSIN   DD  *
          DCOLLECT -
             OFILE(OUTDS) -
             STORAGEGROUP(STGGP001) -
             NODATAINFO
/*

```

Figure 4-2 DCOLLECT job to collect information about all volumes in on storage group

Many clients run DCOLLECT on a time-driven basis triggered by automation. In such a scenario, the peak hours should be avoided due to the heavy access rate caused by DCOLLECT in VTOC and catalogs.

4.3.6 Generation data group

GDG is a catalog function that makes it easier to process data sets with the same type of data but in different update levels. For example, the same data set is produced every day, but with different data. You can then catalog successive updates or generations of these related data sets. Each data set within a GDG is called a generation data set (GDS) or generation.

Within a GDG, the generations can have like or unlike DCB attributes and data set organizations. If the attributes and organizations of all generations in a group are identical, the generations can be retrieved together as a single data set.

Generation data sets can be sequential, PDSs, PDSEs, or direct (BDAM). Generation data sets cannot be UNIX files or VSAM data sets. The same GDG can contain SMS and non-SMS data sets.

There are usability benefits to grouping related data sets using a function such as GDS. For example, the catalog management routines can refer to the information in a special index called a generation index in the catalog, which provides these benefits:

- ▶ All data sets in the group can be referred to by a common name.
- ▶ z/OS is able to keep the generations in chronological order.
- ▶ Outdated or obsolete generations can be automatically deleted from the catalog by z/OS.

Another benefit is the ability to reference a new generation using the same JCL.

A GDG base is allocated in a catalog before the GDSs are cataloged. Each GDG is represented by a *GDG base* entry. Use the access method services DEFINE command to allocate the GDG base.

The GDG base is a construct that exists only in a user catalog, it does not exist as a data set on any volume. The GDG base is used to maintain the GDS, which are the real data sets.

The number of GDSs in a GDG depends on the limit you specify when you create a new GDG in the catalog. The maximum number of generations available is 255 for regular, before z/OS 2.2, or 999 for extended GDGs on z/OS 2.2 and later releases.

A GDS has sequentially ordered *absolute* and *relative* names that represent its age. The catalog management routines use the absolute generation name in the catalog. The relative name is a signed integer used to refer to the most current (0), the next to most current (-1), and so forth, generation.

Absolute generation and version numbers

An absolute generation and version number is used to identify a specific generation of a GDG. The same GDS can have different versions, which are maintained by your installation. The version number allows you to perform normal data set operations without disrupting the management of the generation data group. For example, if you want to update the second generation in a three-generation group, then replace generation 2, version 0, with generation 2, version 1. Only one version is kept under GDG control for each generation.

The generation and version number are in the form *GxxxxVyy*, where xxxx is an unsigned four-digit decimal generation number (0001 - 9999) and yy is an unsigned two-digit decimal version number (00 - 99). For example:

- ▶ A.B.C.G0001V00 is generation data set 1, version 0, in generation data group A.B.C.
- ▶ A.B.C.G0009V01 is generation data set 9, version 1, in generation data group A.B.C.

The number of generations and versions is limited by the number of digits in the absolute generation name, there can be a maximum of 9,999 generations. Each generation can have up to 100 versions. The system automatically maintains the generation number.

You can catalog a generation by using either absolute or relative numbers. When a generation is cataloged, a generation and version number is placed as a low-level entry in the generation data group. To catalog a version number other than V00, you must use an absolute generation and version number.

Relative generation numbers

As an alternative to using absolute generation and version numbers when cataloging or referring to a generation, you can use a relative generation number. To specify a relative number, use the generation data group name followed by a negative integer, a positive integer, or a zero (0), enclosed in parentheses, such as A.B.C(-1), A.B.C(+1), or A.B.C(0).

The value of the specified integer tells the operating system what generation number to assign to a *new generation data set*, or it tells the system the location of an entry representing a previously cataloged *old generation data set*.

When you use a relative generation number to catalog a generation, the operating system assigns an absolute generation number and a version number of V00 to represent that generation. The absolute generation number assigned depends on the number last assigned and the value of the relative generation number that you are now specifying. For example, if in a previous job generation, A.B.C.G0006V00 was the last generation cataloged, and you specify A.B.C(+1), the generation now cataloged is assigned the number G0007V00.

Though any positive relative generation number can be used, a number greater than 1 can cause absolute generation numbers to be skipped for a new generation data set. For example, if you have a single step job and the generation being cataloged is a +2, one generation number is skipped. However, in a multiple step job, one step might have a +1 and a second step a +2, in which case no numbers are skipped. The mapping between relative and absolute numbers is kept until the end of the job.

GDG example

In the example in Figure 4-3, the limit is 5. That means the GDG can hold a maximum of five GDSs. The data set name is ABC.GDG. You can then access the GDSs by their relative names. For example, ABC.GDG(0) corresponds to the absolute name ABC.GDG.G0005V00. ABC.GDG(-1) corresponds to generation ABC.GDG.G0004V00, and so on. The relative number can also be used to catalog a new generation (+1), which will be generation number 6 with an absolute name of ABC.GDG.G0006V00. Because the limit is 5, the oldest generation (G0001V00) is rolled-off if you define a new one.

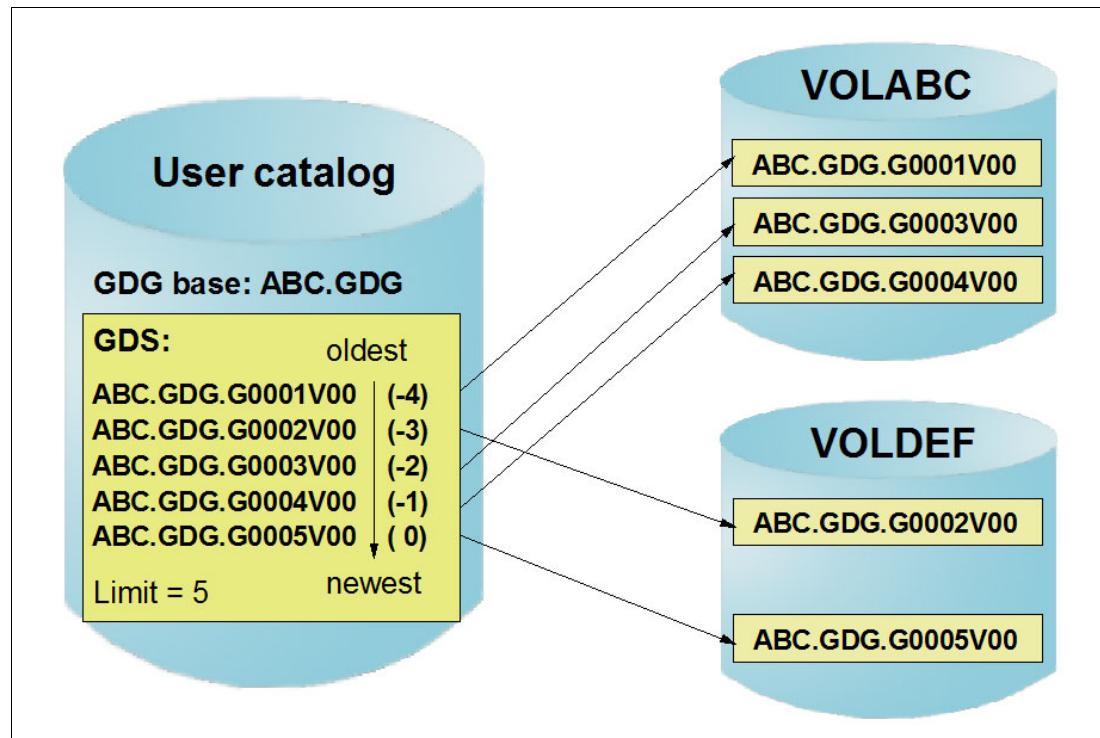


Figure 4-3 Sample GDG structure

Extended GDG

The extended attribute allows system programmers to extend the maximum number of active generations within the GDG base from 255 to 999. This increase provides better flexibility to keep all required versions under GDG control, and reduce the requirement of NOSCRATCH GDGs in your systems. For more information about extended GDGs, see *IBM z/OS V2R2: Storage Management and Utilities*, SG24-8289.

GDG processing order

There are times where you will need to process all GDS files at once. You can achieve that by providing only the GDG base name in your JCL. During the GDG definition, you can also decide how the system performs bulk processing. If you do not specify any reading order, or select FIFO, the data sets are stacked for read in the order they were created. If LIFO was used during GDG definition, the system reverses the order of stacking, going from the newest to the oldest data set.

Rolled in and rolled off

When a GDG contains its maximum number of active generation data sets in the catalog, which is defined in the LIMIT parameter, and a new GDS is rolled in at the end-of-job step, the oldest generation data set is rolled off from the catalog. If a GDG is defined that use DEFINE GENERATIONDATAGROUP EMPTY and is at its limit, then when a new GDS is rolled in, all the currently active GDSs are rolled off.

The parameters that you specify on the DEFINE GENERATIONDATAGROUP IDCAMS command determine what happens to rolled-off GDSs. For example, if you specify the SCRATCH parameter, the GDS is scratched from VTOC when it is rolled off. If you specify the NOSCRATCH parameter, the rolled-off generation data set is recataloged as rolled off and is disassociated with its generation data group.

Creating GDGs with the NOSCRATCH parameter can result in generation overlap if rolled off versions are not deleted/renamed before the GDG version reaches 9999 and the counter is reset to 0001. If version 0001 still exists, a “duplicate data set name” error is issued when an attempt is made to catalog the new GDS.

GDSs can be in a deferred roll-in state if the job never reached end-of-step or if they were allocated as DISP=(NEW,KEEP) and the data set is not system-managed. However, GDSs in a deferred roll-in state can be referred to by their absolute generation numbers. You can use the IDCAMS command ALTER ROLLIN to roll in these GDSs.

For more information about generation data groups, see *z/OS DFSMS: Using Data Sets*, SC26-7410.

4.4 Virtual Storage Access Method

VSAM is a DFSMSdfp component that is used to organize data and maintain information about the data in a catalog. VSAM arranges records by an index key, relative record number, or relative byte addressing. VSAM is used for direct or sequential processing of fixed-length and variable-length records on DASD. Data that is organized by VSAM is cataloged for easy retrieval and is stored in one of five types of data sets.

There are two major parts of VSAM:

- ▶ Catalog management: The catalog contains information about the data sets.
- ▶ Record management: VSAM can be used to organize records into five types of data sets, that is, the VSAM access method function:
 - Key-sequenced data sets (KSDSs) contain records in ascending collating sequence. Records can be accessed by a field, called a key, or by a relative byte address.
 - Entry-sequenced data sets (ESDSs) contain records in the order in which they were entered. Records are added to the end of the data set.
 - Linear data sets (LDSs) contain data that has no record boundaries. Linear data sets contain none of the control information that other VSAM data sets do. Linear data sets must be cataloged in a catalog.
 - Relative record data sets (RRDSs) contain records in relative record number order, and the records can be accessed only by this number. There are two types of relative record data sets:
 - Fixed-length RRDS: The records must be of fixed length.
 - Variable-length RRDS (VRRDS): The records can vary in length.

Several components compose VSAM data sets. Each component plays a unique role in VSAM processing. They are briefly described in the next sections. For a more in-depth description of VSAM processing, see *VSAM Demystified*, SG24-6105.

4.4.1 Logical record

A *logical record* is a unit of application information used to store data in a VSAM cluster. The logical record is designed by the application programmer from the business model. The application program, through a GET, requests that a specific logical record be moved from the I/O device to memory in order to be processed. Through a PUT, the specific logical record is moved from memory to an I/O device. A logical record can be of a fixed size or a variable size, depending on the business requirements.

The logical record is divided into fields by the application program, such as the name of the item, code, and so on. One or more contiguous fields can be defined as a key field to VSAM, and a specific logical record can be retrieved directly by its key value.

Logical records of VSAM data sets are stored differently from logical records in non-VSAM data sets.

4.4.2 Physical record

A *physical record* is device-dependent and is a set of logical records moved during an I/O operation by just one Transport Control Word (TCW) Read or Write. VSAM calculates the physical record size to optimize the track space (to avoid many gaps) at the time the data set is defined. All physical records in VSAM have the same length. A physical record is also referred to as a *physical block* or simply a *block*. VSAM can have control information along with logical records in a physical record.

4.4.3 Control interval

The control interval (CI) is a concept that is unique to VSAM. A CI is formed by one or several physical records (usually just one). It is the fundamental building block of every VSAM file. A CI is a contiguous area of direct-access storage that VSAM uses to store data records and control information that describes the records. A CI is the unit of information that VSAM transfers between the storage device and the main storage during one I/O operation.

Whenever a logical record is requested by an application program, the entire CI containing the logical record is read into a VSAM I/O buffer in virtual storage. The specified logical record is then transferred from the VSAM buffer to a user-defined buffer or work area (if in move mode).

Based on the CI size, VSAM calculates the best size of the physical block to better use the 3390/3380 logical track. The CI size can be from 512 bytes to 32 KB. A CI contents depends on the cluster organization.

The size of CIs can vary from one component to another, but all the CIs within the data or index component of a particular cluster data set must be of the same length. The CI components and properties can vary, depending on the data set organization. For example, an LDS does not contain CIDFs and RDFS in its CI. All of the bytes in the LDS CI are data bytes.

4.4.4 Control area

Control area (CA) is also a concept unique to VSAM. A CA is formed by two or more CIs put together into fixed-length contiguous areas of direct-access storage. A VSAM data set is composed of one or more CAs. In most cases, a CA is the size of a 3390/3380 cylinder. The minimum size of a CA is one track. The CA size is implicitly defined when you specify the size of a data set at data set definition.

CAs are needed to implement the concept of splits. The size of a VSAM file is always a multiple of the CA size, and VSAM files are extended in units of CAs.

4.4.5 Component

A *component* in systems with VSAM is a named, cataloged collection of stored records, such as the data component or index component of a key-sequenced file or alternate index. A component is a set of CAs. It is the VSAM terminology for a z/OS data set. A component has an entry in the VTOC. An example of a component can be the data set containing only data for a KSDS VSAM organization. A VSAM can have up to two components: Data and Index.

Data component

The *data component* is the part of a VSAM cluster, alternate index, or catalog that contains the data records. All VSAM cluster organizations have the data component.

Index component

The *index component* is a collection of records containing data keys and pointers (RBA). The data keys are taken from a fixed defined field in each data logical record. The keys in the index logical records are compressed (rear and front). The RBA pointers are compacted. Only KSDS and VRRDS VSAM data set organizations have the index component.

Using the index, VSAM is able to retrieve a logical record from the data component when a request is made randomly for a record with a certain key. A VSAM index can consist of more than one level (binary tree). Each level contains pointers to the next lower level. Because there are random and sequential types of access, VSAM divides the index component into two parts: The sequence set and the index set.

4.4.6 Cluster

A *cluster* is a named structure that consists of a group of related components. VSAM data sets can be defined with either the DEFINE CLUSTER command or the ALLOCATE command. The cluster is a set of components that have a logical binding between them. For example, a KSDS cluster is composed of the data component and the index component.

The concept of cluster was introduced to improve the flexibility of JCL while accessing VSAM datasets. If you want to access a KSDS normally, just use the cluster's name on a DD card. Otherwise, if you want special processing with just the data, use the data component name on the DD card.

4.4.7 Sphere

A *sphere* is a VSAM cluster and its associated data sets. The cluster is originally defined with the access method services ALLOCATE command, the DEFINE CLUSTER command, or through JCL. The most common use of the sphere is to open a single cluster. The base of the sphere is the cluster itself.

4.4.8 Other VSAM terminologies

Besides the components described previously, this section covers a few terminologies that are related to VSAM data sets that system programmers need to be familiar with.

Spanned records

Spanned records are logical records that are larger than the CI size. They are needed when the application requires very long logical records. To have spanned records, the file must be defined with the SPANNED attribute at the time it is created. Spanned records are allowed to extend across or “span” control interval boundaries, but not beyond control area limits. The RDFS describe whether the record is spanned or not.

A spanned record always begins on a control interval boundary, and fills one or more control intervals within a single control area. A spanned record does *not* share the CI with any other records, so the free space at the end of the last segment is not filled with the next record. This free space is only used to extend the spanned record.

Splits

CI splits and CA splits occur as a result of data record insertions (or increasing the length of an existing record) in KSDS and VRRDS organizations. If a logical record is to be inserted (in key sequence) and there is not enough free space in the CI, the CI is *split*. Approximately half the records in the CI are transferred to a free CI provided in the CA, and the record to be inserted is placed in the original CI.

If there are no free CIs in the CA and a record is to be inserted, a *CA split* occurs. Half the CIs are sent to the first available CA at the end of the data component. This movement creates free CIs in the original CA, and the record to be inserted causes a CI split.

Sequence set

The *sequence set* is the lowest level of index. It directly points (through an RBA) to the data CI in the CA of the data component. Each index logical record has these functions:

- ▶ Maps one CI in the data component
- ▶ Contains pointers and high key information for the data CI

Each index CI has these functions:

- ▶ Maps one data CA.
- ▶ Contains horizontal pointers from one sequence set CI to the next keyed sequence set CI. These horizontal pointers are needed because of the possibility of splits, which make the physical sequence different from the logical collating sequence by key.

Index set

The records in all levels of the index above the sequence set are called the *index set*. An entry in an index set logical record consists of the highest possible key in an index record in the next lower level, and a pointer to the beginning of that index record. The highest level of the index always contains a single index CI.

The structure of VSAM prime indexes is built to create a single index record at the lowest level of the index. If there is more than one sequence-set-level record, VSAM automatically builds another index level.

Figure 4-4 shows a view of the VSAM data set concepts mentioned.

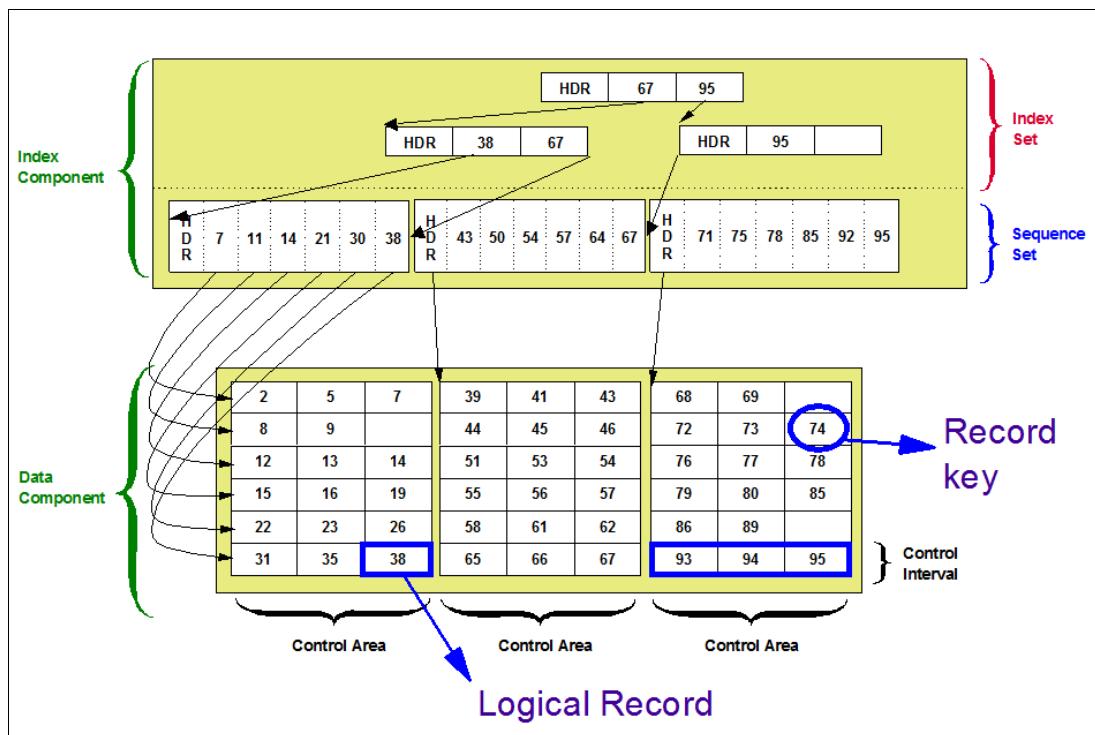


Figure 4-4 VSAM data set components

Alternate index

The alternate index is a VSAM function that allows logical records of a KSDS or ESDS to be accessed sequentially and directly by more than one key field. The cluster that has the data is called the *base cluster*. An alternate index cluster is then built from the base cluster. Alternate indexes eliminate the need to store the same data in different sequences in multiple data sets for the purposes of various applications. Each alternate index is a KSDS cluster that consists of an index component and a data component.

The records in the alternate index index component contain the alternate key and the RBA pointing to the alternate index data component. The records in the alternate index data component contain the alternate key value itself and all the primary keys corresponding to the alternate key value (pointers to data in the base cluster). The primary keys in the logical record are in ascending sequence within an alternate index value.

Any field in the base cluster record can be used as an alternate key. It can also overlap the primary key (in a KSDS), or any other alternate key. The same base cluster can have several alternate indexes that vary the alternate key. There can be more than one primary key value per the same alternate key value. For example, the primary key might be an employee number and the alternate key might be the department name. Obviously, the same department name can have several employee numbers.

Alternate index cluster is created with the IDCAMS DEFINE ALTERNATEINDEX command. It is then populated with the BLDINDEX command. Before a base cluster can be accessed through an alternate index, a path must be defined. A *path* provides a way to gain access to the base data through a specific alternate index. To define a path, use the DEFINE PATH command.

SHAREOPTIONS (crossregion,crosssystem)

The cross-region share options specify the amount of sharing allowed among regions within the same system or multiple systems. Cross-system share options specify how the data set is shared among systems. Use global resource serialization (GRS) or a similar product to perform the serialization.

SHAREOPTIONS (1,x)

The data set can be shared by any number of users for read access (open for input), or it can be accessed by only one user for read/write access (open for output). If the data set is open for output by one user, a read or read/write request by another user will fail. With this option, VSAM ensures complete data integrity for the data set. When the data set is already open for RLS processing, any request to open the data set for non-RLS access will fail.

SHAREOPTIONS (2,x)

The data set can be shared by one user for read/write access, and by any number of users for read access. If the data set is open for output by one user, another open for output request will fail, but a request for read access will succeed. With this option, VSAM ensures write integrity. If the data set is open for RLS processing, non-RLS access for read is allowed. VSAM provides full read and write integrity for its RLS users, but no read integrity for non-RLS access.

SHAREOPTIONS (3,x)

The data set can be opened by any number of users for read and write request. VSAM does not ensure any data integrity. It is the responsibility of the users to maintain data integrity by using enqueue and dequeue macros. This setting does not allow any type of non-RLS access while the data set is open for RLS processing.

SHAREOPTIONS (4,x)

The data set can be fully shared by any number of users. For each request, VSAM refreshes the buffers used for direct processing. This setting does not allow any non-RLS access when the data set is already open for VSAM RLS or DFSMStvs processing. If the data set is opened in non-RLS mode, a VSAM RLS or DFSMStvs open is allowed. As in SHAREOPTIONS 3, each user is responsible for maintaining both read and write integrity for the data that the program accesses.

For more information about VSAM share options, see *z/OS DFSMS: Using Data Sets*, SC26-7410.

4.4.9 Typical KSDS processing

A KSDS has an index that relates key values to the relative locations in the data set. This index is called the *prime index*. It has two uses:

- ▶ Locate the collating position when inserting records
- ▶ Locate records for retrieval

When initially loading a KSDS data set, records must be presented to VSAM in key sequence. This loading can be done through the IDCAMS VSAM utility named REPRO. The index for a key-sequenced data set is built automatically by VSAM as the data set is loaded with records.

When a data CI is completely loaded with logical records, free space, and control information, VSAM makes an entry in the index. The entry consists of the highest possible key in the data control interval and a pointer to the beginning of that control interval.

When accessing records sequentially, VSAM refers only to the sequence set. It uses a horizontal pointer to get from one sequence set record to the next record in collating sequence.

Request for data direct access

When accessing records directly, VSAM follows vertical pointers from the highest level of the index down to the sequence set to find vertical pointers to the requested logical record.

Figure 4-5 shows how VSAM searches the index when an application issues a GET for a logical record with key value 23.

The following is the sequence:

1. VSAM scans the index record in the highest level of the index set for a key that is greater or equal to 23.
2. The entry 67 points to an index record in the next lower level. In this index record, VSAM scans for an entry for a key that is higher than or equal to 23.
3. The entry 38 points to the sequence set that maps the CA holding the CI containing the logical record.
4. VSAM scans the sequence set record with the highest key, searching for a key that is greater than or equal to 23.
5. The entry 26 points to the data component CI that holds the record that you want.
6. VSAM reads the storage device, and loads the CI containing the information into VSAM buffers. Then, it searches the CI for the record with key 23. VSAM finds the logical record and gives it to the application program.

If VSAM does not find a record with the key that you want, the application receives a return code indicating that the record was not found.

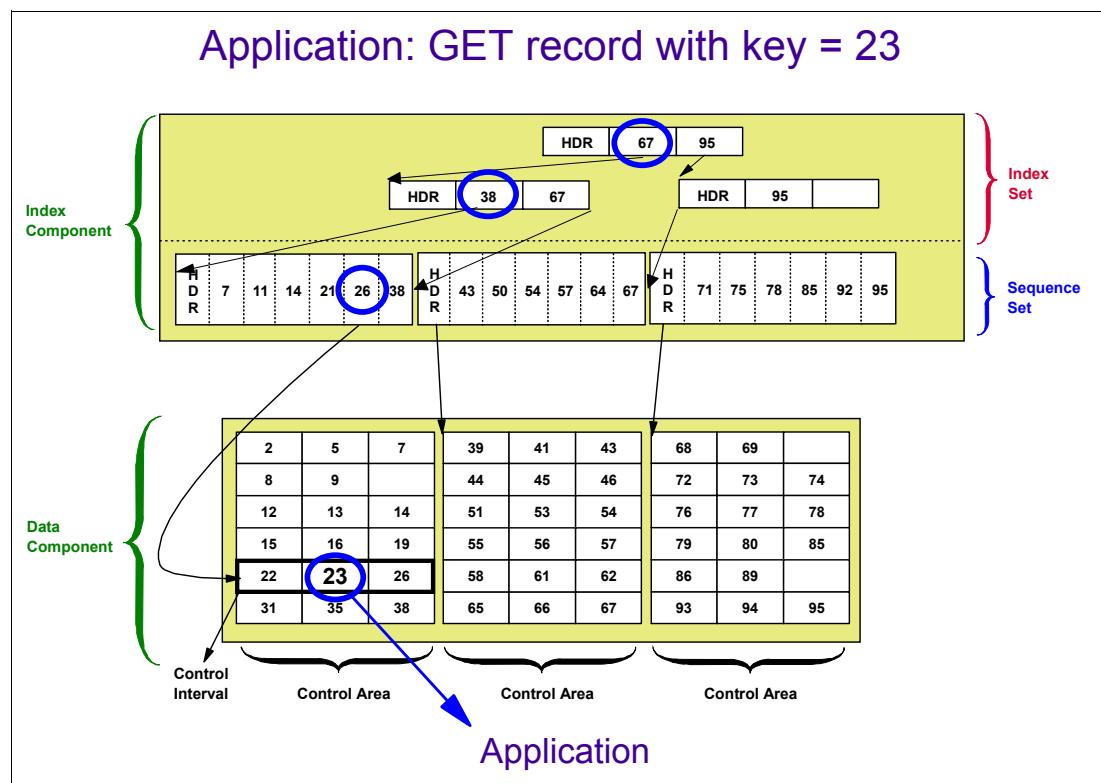


Figure 4-5 Processing an indexed VSAM cluster: Direct access

For more detailed information about VSAM KSDS processing, see *VSAM Demystified*, SG24-6105.

4.4.10 Typical ESDS processing

For an ESDS, two types of processing are supported:

- ▶ Sequential access (the most common).
- ▶ Direct (or random) access requires the program to give the RBA of the record.

Skip sequential is not allowed.

Existing records can never be deleted. If the application wants to delete a record, it must flag that record as inactive. As far as VSAM is concerned, the record is not deleted. Records can be updated, but without length change.

ESDS organization is suited for sequential processing with variable records, but in a few read accesses you need a direct (random) access by key (here using the alternate index cluster).

Figure 4-6 shows a sample structure of VSAM ESDS data sets.

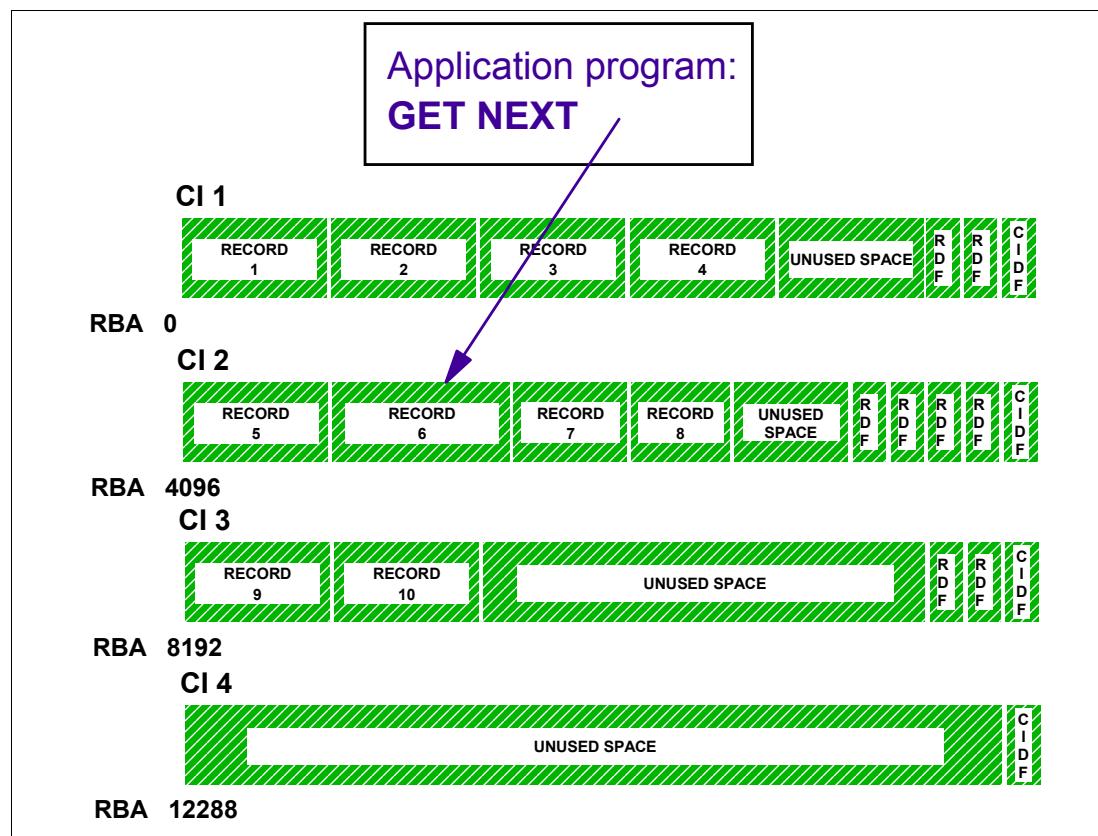


Figure 4-6 Typical ESDS processing (ESDS)

For more information about ESDS processing, see *VSAM Demystified*, SG24-6105.

4.4.11 Typical RRDS processing

The application program inputs the relative record number of the target record. VSAM is able to find its location quickly by using a formula that considers the geometry of the DASD device.

The relative number is always used as a search argument. For an RRDS, three types of processing are supported:

- ▶ Sequential processing
- ▶ Skip-sequential processing
- ▶ Direct processing (in this case, the randomization routine is supported by the application program)

Figure 4-7 shows a typical processing of a VSAM fixed-length RRDS data set.

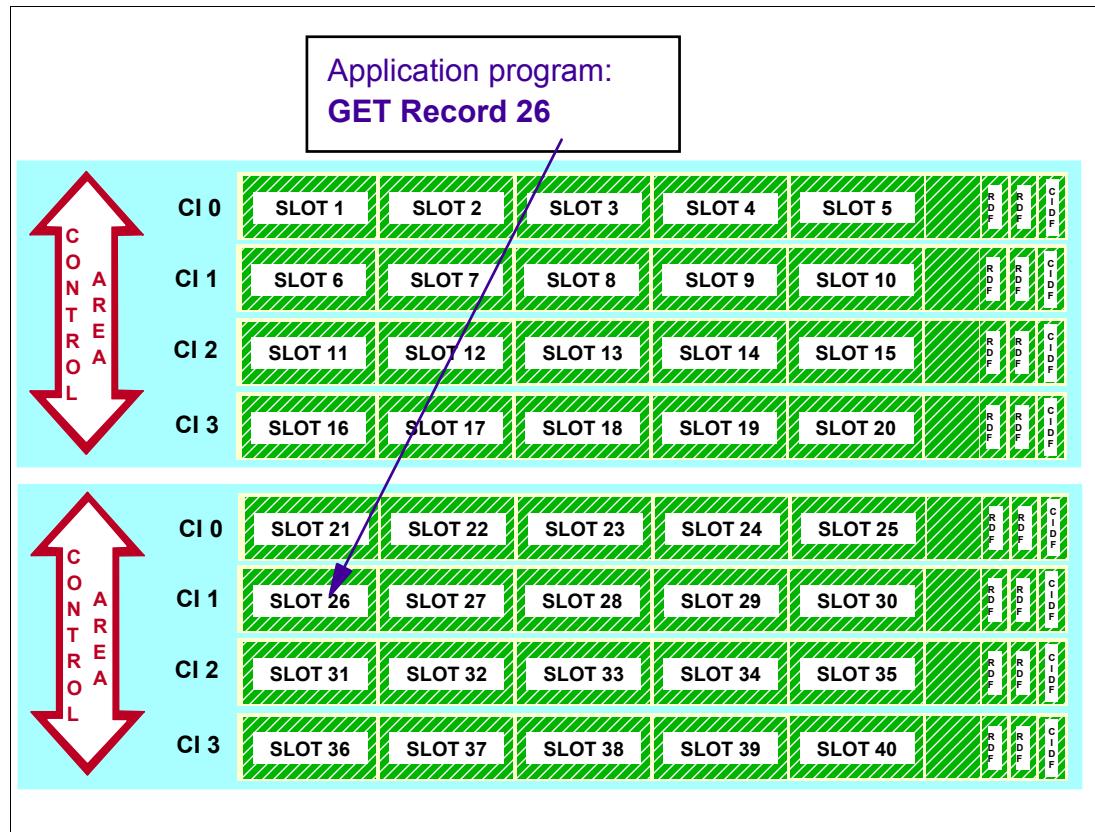


Figure 4-7 Typical RRDS processing

For extended information about RRDS processing, see *VSAM Demystified*, SG24-6105.

4.4.12 Typical LDS processing

A linear data set is a VSAM data set with a control interval size from 4096 bytes to 32768 bytes, in increments of 4096 bytes. An LDS has no embedded control information in its CI, that is, no record definition fields (RDFs) and no control interval definition fields (CIDFs).

Therefore, all LDS bytes are data bytes. Logical records must be blocked and deblocked by the application program (although logical records do not exist, from the point of view of VSAM).

IDCAMS is used to define a linear data set. An LDS has only a data component. An LDS data set is just a physical sequential VSAM data set that consists of 4 KB physical records, but with a revolutionary buffer technique called DIV. A linear data set is processed as an entry-sequenced data set, with certain restrictions.

Because a linear data set does not contain control information, it cannot be accessed as though it contained individual records. You can access a linear data set with the DIV macro. If using DIV to access the data set, the control interval size must be 4096. Otherwise, the data set will not be processed.

When a linear data set is accessed with the DIV macro, it is referred to as the data-in-virtual object or the data object.

Figure 4-8 shows a sample structure of a VSAM LDS data set.



Figure 4-8 Typical LDS data set

4.4.13 VSAM: Data-in-virtual

DIV is an optional and unique buffering technique that is used for LDS data sets. Application programs can use DIV to *map* a data set (or a portion of a data set) into an address space, a data space, or a hyperspace. An LDS cluster is sometimes referred to as a *DIV object*. After setting the environment, the LDS cluster looks to the application as a table in virtual storage with no need to issue I/O requests.

Data is read into main storage by the paging algorithms only when that block is actually referenced. During real storage manager (RSM) page-steal processing, only changed pages are written to the cluster in DASD. Unchanged pages are discarded because they can be retrieved again from the permanent data set.

DIV is designed to improve the performance of applications that process large files non-sequentially and process them with significant locality of reference. It reduces the number of I/O operations that are traditionally associated with data retrieval. Likely candidates are large arrays or table files.

For information about how to use data-in-virtual, see *z/OS MVS Programming: Assembler Services Guide*, SA22-7605.

4.4.14 VSAM resource pool

Buffering is one of the key aspects as far as I/O performance is concerned. A VSAM *buffer* is a virtual storage area where the CI is transferred during an I/O operation. VSAM KSDS uses two types of buffers: Buffers for data CIs and buffers for index CIs. A *buffer pool* is a set of buffers with the same size. A *resource pool* is a buffer pool with several control blocks describing the pool and describing the clusters with CIs in the resource pool.

The objective of a buffer pool is to avoid I/O operations in random accesses (due to revisiting data) and to make these I/O operations more efficient in sequential processing, improving performance.

For more efficient use of virtual storage, buffer pools can be shared among clusters using locally or globally shared buffer pools. There are four types of resource pool management, called *modes*, defined according to the technique used to manage them:

- ▶ Nonshared resources (NSR)
- ▶ Local shared resources (LSR)
- ▶ Global shared resources (GSR)
- ▶ Record-level shared resources (RLS)

These modes can be declared in the ACB macro of the VSAM data set (MACRF keyword) and are described in the following sections.

Non-shared resource

Non-shared resource (NSR) is the default VSAM buffering technique. The following are some of the characteristics of NSR buffers:

- ▶ The resource pool is implicitly constructed at data set open time.
- ▶ The buffers are not shared among VSAM data sets. Only one cluster has CIs in this resource pool.
- ▶ Buffers are located in the private area.
- ▶ For sequential reads, VSAM uses the read-ahead function. When the application finishes processing half the buffers, VSAM schedules an I/O operation for that half of the buffers.

This process continues until a CA boundary is encountered. The application must wait until the last I/O to the CA is done before proceeding to the next CA. The I/O operations are always scheduled within CA boundaries.

- ▶ For sequential writes, VSAM postpones the writes to DASD until half the buffers are filled by the application. Then VSAM schedules an I/O operation to write that half of the buffers to DASD. The I/O operations are always scheduled within CA boundaries.
- ▶ CIs are discarded as soon as they are used by using a sequential algorithm to keep CIs in the resource pool.
- ▶ There is dynamic addition of strings. Strings are like cursors. Each string represents a position in the data set for the requested record.
- ▶ For random access, there is no look-ahead, but the algorithm remains sequential.

NSR is used by high-level languages. Because buffers are managed by a sequential algorithm, NSR is not the best choice for random processing. For more details about NSR buffers, see *VSAM Demystified*, SG24-6105.

Local shared resource

An LSR resource pool is suitable for random processing, not for sequential processing. LSR has the following characteristics:

- ▶ Shared among VSAM clusters accessed by tasks in the same address space.
- ▶ Located in the private area and expanded storage only (ESO) hiperspace. With hiperspace, VSAM buffers are located in expanded storage to improve the processing of VSAM clusters. With z/Architecture, ESO hiperspaces are mapped in main storage.
- ▶ A resource pool is explicitly constructed by macro BLDVRP before the OPEN.
- ▶ Buffers are managed by the last recently used (LRU) algorithm, so the most referenced CIs are kept in the resource pool. It is adequate for random processing.
- ▶ LSR expects that CIs in buffers *are* revisited.

For extra information about LSR, see *VSAM Demystified*, SG24-6105.

Global shared resource

GSR is similar to the LSR buffering technique. GSR differs from LSR in the following ways:

- ▶ The buffer pool is shared among VSAM data sets accessed by tasks in *multiple* address spaces in the same z/OS image.
- ▶ Buffers are located in CSA.
- ▶ The code using GSR must be in the supervisor state.
- ▶ Buffers cannot use hiperspace.
- ▶ The separate index resource pools are not supported for GSR.

GSR is not commonly used by applications, so consider using VSAM RLS instead. For extra information about LSR, see *VSAM Demystified*, SG24-6105.

Record-level sharing

RLS is the implementation of VSAM data sharing. Record-level sharing is discussed in detail in Chapter 7, “DFSMS Transactional VSAM Services” on page 119.

For more information about NSR, LSR, and GSR, see *VSAM Demystified*, SG24-6105.

4.4.15 VSAM: System-managed buffering

System-managed buffering (SMB) enables VSAM to perform these tasks:

- ▶ Determine the optimum number of index and data buffers
- ▶ Change the buffer algorithm as declared in the application program in the ACB MACRF parameter, from NSR (sequential) to LSR (LRU)

Usually, SMB allocates many more buffers than are allocated without SMB. Performance improvements can be dramatic with random access (particularly when few buffers were available). The use of SMB is transparent from the point of view of the application, so no application changes are needed.

SMB is available to a data set when *all* of the following conditions are met:

- ▶ It is an SMS-managed data set.
- ▶ It is in extended format (DSNTYPE = EXT in the data class).
- ▶ The application opens the data set for NSR processing.

SMB is started or disabled by using one of the following methods:

1. The Record Access Bias data class field.
2. The ACCBIAS subparameter of AMP in the JCL DD statement. JCL information takes precedence over data class information.

If all of the required conditions are met, SMB is started when option SYSTEM or an SMB processing technique is used in the fields described previously. SMB is disabled when USER is entered instead (USER is the default). Because JCL information takes precedence over data class information, installations can enable or disable SMB for some executions.

SMB uses formulas to calculate the storage and buffer numbers needed for a specific access type. SMB takes the following actions:

- ▶ It changes the defaults for processing VSAM data sets. This action enables the system to take better advantage of current and future hardware technology.
- ▶ It initiates a buffering technique to improve application performance. The technique is one that the application program does not specify. You can choose or specify any of the four processing techniques that SMB implements:

Direct Optimized (DO)	The DO processing technique optimizes for totally random record access. This technique is appropriate for applications that access records in a data set in totally random order. This technique overrides the user specification for NSR buffering with an LSR implementation of buffering.
Sequential Optimized (SO)	The SO technique optimizes processing for record access that is in sequential order. This technique is appropriate for backup and for applications that read the entire data set or a large percentage of the records in sequential order.
Direct Weighted (DW)	This technique is optimal when most of the processing is direct, and some is sequential. DW processing provides the minimum read-ahead buffers for sequential retrieval and the maximum index buffers for direct requests.
Sequential Weighted (SW)	This technique is optimal when most of the processing is sequential processing, and some is direct. This technique uses read-ahead buffers for sequential requests and provides additional index buffers for direct requests. The read-ahead will not be as large as the amount of data transferred with SO.

For more information about the use of SMB, see *VSAM Demystified*, SG24-6105.

4.5 Data set separation

Data set separation allows you to designate groups of data sets in which all SMS-managed data sets within a group are kept separate, on the physical control unit (PCU) level or the volume level, from all the other data sets in the same group.

When allocating new data sets or extending existing data sets to new volumes, SMS volume selection frequently calls SRM to select the best volumes. Unfortunately, SRM might select the same set of volumes that currently have the lowest I/O delay. Poor performance or single points of failure might occur when a set of functional-related critical data sets are allocated onto the same volumes. SMS provides a function to separate their critical data sets, such as DB2 partitions, onto different volumes to prevent DASD hot spots and reduce I/O contention.

The user defines the volume separation groups in a data set separation profile. During data set allocation, SMS attempts to separate data sets that are specified in the same separation group onto different extent pools and volumes.

This technique provides a facility for an installation to separate functional-related critical data sets onto different extent pools and volumes for better performance and to avoid single points of failure.

To use data set separation, you must create a data set separation profile and specify the name of the profile to the base configuration. During allocation, SMS attempts to separate the data sets listed in the profile.

A data set separation profile contains at least one data set separation group. Each data set separation group specifies whether separation is at the PCU or volume level, and whether it is required or preferred. It also includes a list of data set names to be separated from each other during allocation.

For details about data set separation implementation and syntax, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

Important: Use data set separation only for a small set of mission-critical data.

4.6 Data Facility sort

The Data Facility sort (DFSORT) licensed program is a high-performance data arranger for z/OS users. Using DFSORT, you can sort, merge, and copy data sets using EBCDIC, UNICODE, z/Architecture decimal or binary keys. It also helps you to analyze data and produce detailed reports using the ICETOOL utility or the OUTFIL function. DFSORT is an optional feature of z/OS.

DFSORT, together with DFSMS and RACF, form the strategic product base for the evolving system-managed storage environment. DFSORT is designed to optimize the efficiency and speed with which operations are completed through synergy with processor, device, and system features (for example, memory objects, IBM Hiperspace™, data space, striping, compression, extended addressing, DASD and tape device architecture, processor memory, and processor cache).

You can use DFSORT to do simple application tasks such as alphabetizing a list of names, or you can use it to aid complex tasks such as taking inventory or running a billing system. You can also use the record-level editing capability of DFSORT to perform data management tasks.

For most of the processing done by DFSORT, the whole data set is affected. However, certain forms of DFSORT processing involve only certain individual records in that data set.

4.6.1 DFSORT functions

DFSORT adds the ability to do faster and easier sorting, merging, copying, reporting, and analysis of your business information, as well as versatile data handling at the record, fixed position/length or variable position/length field, and bit level.

While sorting, merging, or copying data sets, you can perform these tasks:

- ▶ Select a subset of records from an input data set. You can include or omit records that meet specified criteria. For example, when sorting an input data set containing records of course books from many different school departments, you can sort the books for only one department.
- ▶ Reformat records, add or delete fields, and insert blanks, constants, or binary zeros. For example, you can create an output data set that contains only certain fields from the input data set, arranged differently.
- ▶ Sum the values in selected records while sorting or merging (but not while copying). For the example of a data set containing records of course books, you can use DFSORT to add up the dollar amounts of books for one school department.
- ▶ Create multiple output data sets and reports from a single pass over an input data set. For example, you can create a different output data set for the records of each department.
- ▶ Sort, merge, include, or omit records according to the collating rules defined in a selected local.
- ▶ Alter the collating sequence when sorting or merging records (but not while copying). For example, you can have the lowercase letters collate after the uppercase letters.
- ▶ Sort, merge, or copy Japanese data if the IBM Double Byte Character Set Ordering Support (DBCS Ordering, the 5665-360 Licensed Program, Release 2.0, or an equivalent product) is used with DFSORT to process the records.

DFSOR has utilities such as ICETOOL, which is a multipurpose DFSORT utility that uses the capabilities of DFSORT to perform multiple operations on one or more data sets in a single step.

Tip: You can use the ICEGENER facility of DFSORT to achieve faster and more efficient processing for applications that are set up to use the IEBGENER system utility. For more information, see *z/OS DFSORT Application Programming Guide*, SC26-7523.

4.7 Data Set Services (DFSMSdss)

DFSMSdss is a DASD data and space management tool. DFSMSdss works on DASD volumes only in a z/OS environment. You can use DFSMSdss to perform the following tasks:

- ▶ Copy and move data sets between volumes of like and unlike device types.
- ▶ Dump and restore data sets, entire volumes, or specific tracks.
- ▶ Convert data sets and volumes to and from SMS management.
- ▶ Compress partitioned data sets.
- ▶ Release unused space in data sets.
- ▶ Reduce or eliminate DASD free-space fragmentation by consolidating free space on a volume.
- ▶ Implement concurrent copy or a FlashCopy in ESS or DS8000 DASD controllers.

You can use the DFSMSdss to back up, copy, dump, move, or restore data in your systems. DFSMSdss can be called by JCL or by external programs, such as DFMSHsm, to perform data movement. It also participates in disaster recovery solutions, by allowing the recovery of your systems from a stand-alone DFSMSdss restore function, or copy and replication services, such as GDPS solutions.

This section provides a high-level description of DFSMSdss basic functions, so no data replication services are covered here. For more information about DFSMSdss, capabilities, and copy services, see *z/OS DFSMSdss Storage Administration*, SC23-6868.

4.7.1 Physical and logical processing

Before you begin using DFSMSdss, you should understand the difference between logical processing and physical processing and how to use data set filtering to select data sets for processing. DFSMSdss can perform two kinds of processing when running COPY, DUMP, and RESTORE commands:

- ▶ *Logical processing* operates against data sets independently of physical device format.
- ▶ *Physical processing* moves data at the track-image level and operates against volumes, tracks, and data sets.

Each type of processing offers different capabilities and advantages.

During a restore operation, the data is processed the same way that it is dumped because physical and logical dump tapes have different formats. If a data set is dumped logically, it is restored logically, and if it is dumped physically, it is restored physically. A data set restore operation from a full volume dump is a physical data set restore operation.

Logical processing

A logical copy, dump, or restore operation treats each data set and its associated information as a logical entity, and processes an entire data set before beginning the next one.

Each data set is moved by tracks from the source device and is potentially written to the target device as a set of data records, allowing data movement between devices with different track and cylinder configurations. Checking of data record consistency is not performed during the dump operation.

DFSMSdss performs logical processing in these circumstances:

- ▶ You specify the DATASET keyword with the COPY command. A data set copy is always a logical operation, regardless of how or whether you specify input volumes.
- ▶ You specify the DATASET keyword with the DUMP command, and either no input volume is specified, or LOGINDDNAME, LOGINDYNAM, or STORGRP is used to specify input volumes.
- ▶ The RESTORE command is performed, and the input volume was created by a logical dump.

Catalogs and VTOCs are used to select data sets for logical processing. If you do not specify input volumes, the catalogs are used to select data sets for copy and dump operations. If you specify input volumes using the LOGINDDNAME, LOGINDYNAM, or STORGRP keywords on the COPY or DUMP command, DFSMSdss uses VTOCs to select data sets for processing.

When to use logical processing

Use logical processing for the following situations:

- ▶ Data is copied to an unlike device type.
Logical processing is the only way to move data between unlike device types.
- ▶ Data that might need to be restored to an unlike device is dumped.

This concern is particularly important to remember when making backups that you plan to retain for a long time (such as vital records backups).

If a backup is retained for a long time, the device type that it originally was on might no longer be in use at your site when you want to restore it. This situation means you must restore it to an unlike device, which can be done only if the backup has been made logically.

- ▶ Aliases of VSAM user catalogs are to be preserved during copy and restore functions.
Aliases are not preserved for physical processing.
- ▶ Unmovable data sets or data sets with absolute track allocation are moved to different locations.
- ▶ Multivolume data sets are processed.
- ▶ VSAM and multivolume data sets are to be cataloged as part of DFSMSdss processing.
- ▶ Data sets are to be deleted from the source volume after a successful dump or copy operation.
- ▶ Both non-VSAM and VSAM data sets are to be renamed after a successful copy or restore operation.
- ▶ You want to control the percentage of space allocated on each of the output volumes for copy and restore operations.
- ▶ You want to copy and convert a PDS to a PDSE or vice versa.
- ▶ You want to copy or restore a data set with an undefined DSORG to an unlike device.
- ▶ You want to keep together all parts of a VSAM sphere.

Physical processing

Physical processing moves data based on physical track images. Because data movement is carried out at the track level, only target devices with track sizes equal to those of the source device are supported. Physical processing operates on volumes, ranges of tracks, or data sets. For data sets, it relies only on volume information (in the VTOC and VVDS) for data set selection, and processes only that part of a data set residing on the specified input volumes.

DFSMSdss performs physical processing in these circumstances:

- ▶ You specify the FULL or TRACKS keyword with the COPY or DUMP command. This situation results in a physical volume or physical tracks operation.

Attention: Take care when using the TRACKS keyword with the COPY and RESTORE commands. The TRACKS keyword should be used only for a data recovery operation. For example, you can use it to “repair” a bad track in the VTOC or a data set, or to retrieve data from a damaged data set. You cannot use it in place of a full-volume or a logical data set operation. Doing so can destroy a volume or impair data integrity.

- ▶ You specify the data set keyword on the DUMP command and input volumes with the INDDNAME or INDYNAM parameter. This situation produces a physical data set dump.
- ▶ The RESTORE command is run and the input volume is created by a physical dump operation.

When to use physical processing

Use physical processing in these circumstances:

- ▶ Backing up system volumes that you might want to restore with a stand-alone DFSMSdss restore operation.

Stand-alone DFSMSdss restore supports only physical dump tapes.

- ▶ Performance is an issue.
Generally, the fastest way (measured by elapsed time) to copy or dump an entire volume is by using a physical full-volume command. This advantage is primarily because minimal catalog searching is necessary for physical processing.
- ▶ Substituting one physical volume for another or recovering an entire volume.
With a COPY or RESTORE (full volume or track) command, the volume serial number of the input DASD volume can be copied to the output DASD volume.
- ▶ Dealing with I/O errors.
Physical processing provides the capability to copy, dump, and restore a specific track or range of tracks.
- ▶ Dumping or copying between volumes of the same device type but different capacity.

4.7.2 DFSMSdss stand-alone services

The DFSMSdss stand-alone restore function is a single-purpose program. It is designed to allow the system programmer to restore vital system packs during disaster recovery without relying on an MVS environment. Stand-alone services can perform either a full-volume restore or a track restore from dump tapes produced by DFSMSdss or DFDSS.

For detailed information about the stand-alone service and other DFSMSdss information, see *z/OS DFSMSdss Storage Administration Reference*, SC35-0424, and *z/OS DFSMSdss Storage Administration Guide*, SC35-0423.

4.8 Hierarchical Storage Manager

Hierarchical Storage Manager (DFSMShsm) is a disk storage management and productivity product for managing low activity and inactive data. It provides backup, recovery, migration, and space management functions as well as full-function disaster recovery support.

DFSMShsm improves disk use by automatically managing both space and data availability in a storage hierarchy.

Availability management is used to make data available by automatically copying new and changed data set to backup volumes.

Space management is used to manage DASD space by enabling inactive data sets to be moved off fast-access storage devices, thus creating free space or new allocations.

DFSMShsm also provides for other supporting functions that are essential to your installation's environment. For further information about DFSMShsm, see *z/OS DFSMShsm Storage Administration Guide*, SC35-0421 and *z/OS DFSMShsm Storage Administration Reference*, SC35-0422.

4.8.1 DFSMShsm data sets

DFSMShsm assists you in the space and availability tasks related to user and application data sets. It stores and maintains information about backups and migrated data sets under its control, based on the management policies defined in SMS Management Class (MC) construct and DFSMShsm parmlib.

To retrieve information in a timely matter of time, DFSMShsm uses control data sets (CDSs) that record data about data sets, tapes, and DASD volumes. These records are stored in different CDSs depending on the record content. There are four CDSs used by DFSMShsm, which are explained next:

- ▶ **Backup control data set (BCDS)**

The BCDS provides DFSMShsm with information defining the backup and dump environment. DFSMShsm needs this information to manage its backup data sets and volumes. DFSMShsm updates the contents of the BCDS with current backup version information.

- ▶ **Migration control data set (MCDS)**

The MCDS provides information about migrated data sets and the volumes they migrate to and from. Additionally, internal processing statistics and processing-unit-environment data are in the MCDS. DFSMShsm needs this information to manage its data sets and volumes. With this information you can verify, monitor, and tune your storage environment

- ▶ **Offline control data set (OCDS)**

The OCDS provides DFSMShsm with information about each migration and backup tape and about each data set on these tapes.

- ▶ **Journal data set**

The journal data set provides DFSMShsm with a record of each critical change made to a control data set from any host since the last time that CDS was successfully backed up. DFSMShsm recovers control data sets by merging the journal with a backed-up version of the control data set. Control data sets cannot be recovered to the point of failure without the journal. Use of the journal is highly preferred.

The data sets mentioned in this section are critical to DFSMShsm, and the loss of any of these data sets can potentially result in data loss. For this reason, create backups of DFSMShsm CDSs.

You can achieve that by using SETSYS CDSVERSIONBACKUP command. For more information about CDS data sets and how to create the backup files, see *z/OS DFSMShsm Implementation and Customization Guide*, SC23-6869.

4.8.2 Storage device hierarchy

A storage device hierarchy consists of a group of storage devices that have different costs for storing data, different amounts of data stored, and different speeds of accessing the data, which is processed as follows:

Level 0 volumes A volume that contains data sets that are directly accessible by the user. The volume can be either DFSMShsm-managed or non-DFSMShsm-managed.

Level 1 volumes A volume owned by DFSMShsm containing data sets that migrated from a level 0 volume.

Level 2 volumes A volume under control of DFSMShsm containing data sets that migrated from a level 0 volume, from a level 1 volume, or from a volume that is not managed by DFSMShsm.

Besides the storage hierarchy defined here, there are several volume types that are supported/controlled by DFSMShsm, including the following:

- ▶ *Level 0 (L0) volumes* contain data sets that are directly accessible to you and the jobs that you run. DFSMShsm-managed volumes are those L0 volumes that are managed by the DFSMShsm automatic functions. These volumes must be mounted and online when you refer to them with DFSMShsm commands.
- ▶ *Migration level 1 (ML1) volumes* are DFSMShsm-supported DASD on which DFSMShsm maintains your data in DFSMShsm format. These volumes are normally permanently mounted and online. They can be these types of volumes:
 - Volumes containing data sets that DFSMShsm migrated from L0 volumes.
 - Volumes containing backup versions created from a DFSMShsm BACKDS or HBACKDS command. Backup processing requires ML1 volumes to store incremental backup and dump VTOC copy data sets, and as intermediate storage for data sets that are backed up by data set command backup.
- ▶ *Migration level 2 (ML2) volumes* are DFSMShsm-supported tape or DASD on which DFSMShsm maintains your data in DFSMShsm format. These volumes are normally not mounted or online. They contain data sets migrated from ML1 volumes or L0 volumes.
- ▶ *Daily backup volumes* are DFSMShsm-supported tape or DASD on which DFSMShsm maintains your data in DFSMShsm format. These volumes are normally not mounted or online. They contain the most current backup versions of data sets copied from L0 volumes. These volumes can also contain earlier backup versions of these data sets.
- ▶ *Spill backup volumes* are DFSMShsm-supported tape or DASD on which DFSMShsm maintains your data sets in DFSMShsm format. These volumes are normally not mounted or online. They contain earlier backup versions of data sets, which were moved from DASD backup volumes.
- ▶ *Dump volumes* are DFSMShsm-supported tape. They contain image copies of volumes that are produced by the full volume dump function of DFSMSdss (write a copy of the entire allocated space of that volume), which is started by DFSMShsm.
- ▶ *Aggregate backup volumes* are DFSMShsm-supported tape. These volumes are normally not mounted or online. They contain copies of the data sets of a user-defined group of data sets, along with control information for those data sets. These data sets and their control information are stored as a group so that they can be recovered (if necessary) as an entity by an aggregate backup and recovery process (ABARS).
- ▶ *Fast replication target volumes* are contained within SMS copy pool backup storage groups. They contain the fast replication backup copies of DFSMShsm-managed volumes. Beginning with z/OS V1R8, fast replication is done with a single command.
- ▶ *Overflow volumes* are ML1 volumes with OVERFLOW parameter set during ADDVOL statement. The overflow volumes are used in situations where a migration or backup task is started, and the data set exceeds the size specified in SETSYS ML1OVERFLOW command, and the NOOVERFLOW volumes do not have enough space to allocate the data set.
- ▶ *Cloud migration* are cloud storage that can be used by DFSMShsm to migrate and recall data sets. Using cloud services to store migrated data sets relieve many DFSMShsm constraints, including data set reblocking, need to recycle, and CPU processing.

For more information about DFSMShsm devices, see *z/OS DFSMShsm Implementation and Customization Guide*, SC23-6869.

4.8.3 Availability management

DFSMShsm backs up your data, automatically or by command, to ensure availability if accidental loss of the data sets or physical loss of volumes occurs. DFSMShsm also allows the storage administrator to copy backup and migration tapes, and to specify that copies be made in parallel with the original.

You can store the copies on site as protection from media damage, or offsite as protection from site damage. DFSMShsm also provides disaster backup and recovery for user-defined groups of data sets (aggregates) so that you can restore critical applications at the same location or at an offsite location.

Figure 4-9 shows DFSMShsm processing for availability management tasks.

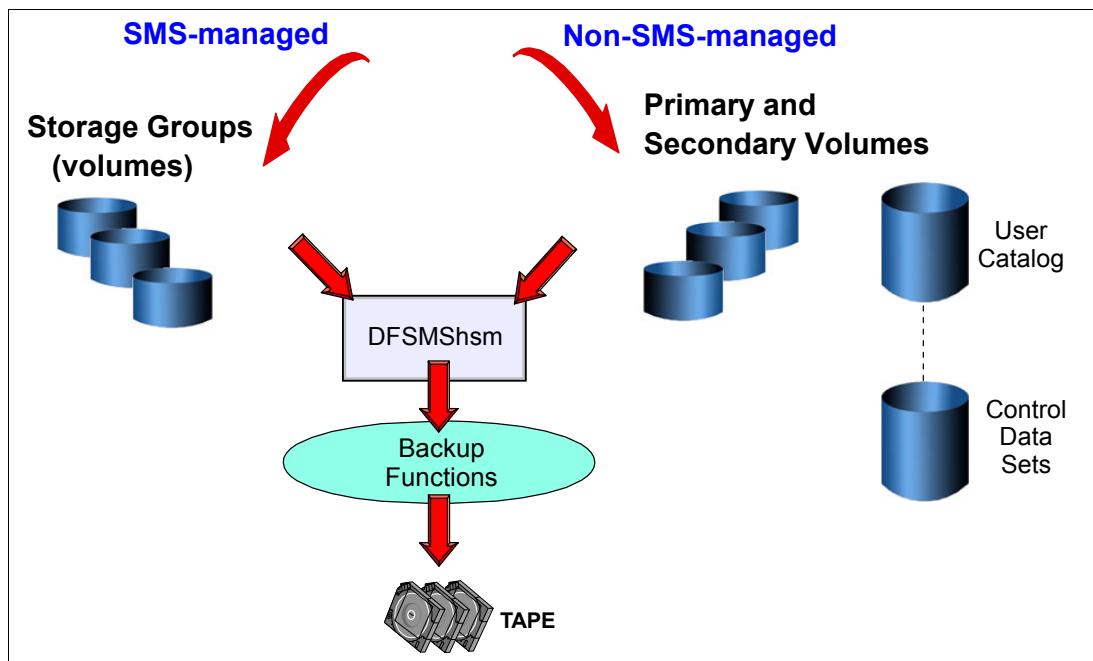


Figure 4-9 DFSMShsm availability management

Note: You must also have DFSMSdss to use the DFSMShsm functions.

Availability management ensures that a recent copy of your DASD data set exists. The purpose of availability management is to ensure that lost or damaged data sets can be retrieved at the most current possible level. DFSMShsm uses DFSMSdss as a fast data mover for backups. Availability management automatically and periodically performs the following functions:

1. Copy all the data sets on DASD volumes to tape volumes
2. Copy the changed data sets on DASD volumes (incremental backup) either to other DASD volumes or to tape volumes

DFSMShsm minimizes the space occupied by the data sets on the backup volume by using compression and stacking.

Tasks for availability management functions

The tasks for controlling automatic availability management of SMS-managed storage require adding DFSSMShsm commands to the ARCCMDxx parmlib member and specifying attributes in the SMS storage classes and management classes. It is assumed that the storage classes and management classes have already been defined.

The attribute descriptions explain the attributes to be added to the previously defined storage groups and management classes. Similarly, the descriptions of DFSSMShsm commands relate to commands to be added to the ARCCMDxx member of SYS1.PARMLIB.

Two groups of tasks are performed for availability management: Dump tasks and backup tasks. Availability management is composed of the following functions:

- ▶ Aggregate backup and recovery (ABARS)
- ▶ Automatic physical full-volume dump
- ▶ Automatic incremental backup
- ▶ Automatic control data set backup
- ▶ Command dump and backup
- ▶ Command recovery
- ▶ Disaster backup
- ▶ Expiration of backup versions
- ▶ Fast replication backup and recovery

Command availability management

Commands cause availability management functions to occur, resulting in the following conditions:

- ▶ One data set to be backed up.
- ▶ All changed data sets on a volume to be backed up.
- ▶ A volume to be dumped.
- ▶ A backed-up data set to be recovered.
- ▶ A volume to be restored from a dump and forward recovered from later incremental backup versions. *Forward recovery* is a process of updating a restored volume by applying later changes as indicated by the catalog and the incremental backup versions.
- ▶ A volume to be restored from a dump copy.
- ▶ A volume to be recovered from backup versions.
- ▶ A specific data set to be restored from a dump volume.
- ▶ All expired backup versions to be deleted.
- ▶ A fast replication backup version to be recovered from a copy pool.

Common dump and recovery queues

Besides the capacity of backing up and restoring data sets and volumes, DFSSMShsm also provides the capability to spread dump and recovery requests across several DFSSMShsm images. By doing so, you can reduce the amount of time that is required to dump and restore your data, thus reducing your backup and recovery windows.

These common queues use XCF messaging along with a master host to receive, and distribute the workload to available hosts. There can be only one master host at a time, and several submitters and processing hosts. If the master host becomes offline, another available host can assume the master role.

For more information about common dump and common recovery queues, including implementation steps, see *z/OS DFSMShsm Implementation and Customization Guide*, SC23-6869.

4.8.4 Space management

Space management is the function of DFSMShsm that allows you to keep DASD space available for users to meet the service level objectives for your system. The purpose of space management is to manage your DASD storage efficiently. To do this, space management automatically and periodically performs these functions:

1. Move low activity data sets (using DFSMSdss) from user-accessible volumes to DFSMShsm volumes
2. Reduce the space that is occupied by data on both the user-accessible volumes and the DFSMShsm volumes

DFSMShsm improves DASD space usage by keeping only active data on fast-access storage devices. It automatically frees space on user volumes by deleting eligible data sets, releasing overallocated space, and moving low-activity data to lower cost-per-byte devices, even if the job did not request tape. Figure 4-10 shows a sample space management processing performed by DFSMShsm.

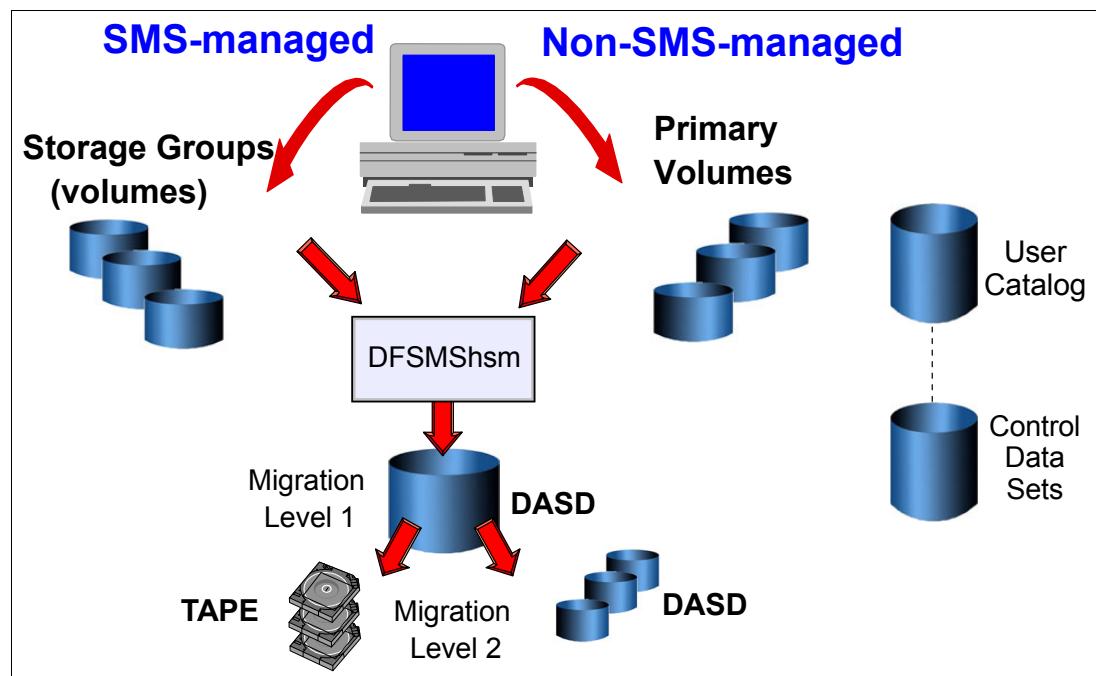


Figure 4-10 DFSMShsm space management

Several DFSMShsm space management functions are available:

- ▶ Automatic primary space management

Run daily, this function cleans L0 volumes by deleting expired and temporary data sets, and releasing allocated and not used space (scratch). During automatic primary space management, DFSMShsm can process a maximum of 15 volume migration tasks concurrently.

This activity consists of the deletion of temporary data sets, deletion of expired data sets, the release of unused and overallocated space, migration, and class transition. Each task processes its own separate user DASD volume. The storage administrator selects the maximum number of tasks that can run simultaneously, and specifies which days and the time of day the tasks are to be performed.

- ▶ Automatic secondary space management

Automatic secondary space management deletes expired data sets from ML1/ML2, then moves data sets (under control of the management class) from ML1 to ML2 volumes. It should complete before automatic primary space management so that the ML1 volumes will not run out of space.

- ▶ Automatic interval migration

This function is initiated when a DFSMShsm-managed volume exceeds a specified threshold. Interval migration can select data sets for migration or transition, depending on the management class and DFSMShsm configuration. The interval migration processing is always performed at the top of the hour. If the threshold is exceeded after this check, the migration will occur only in the next window.

- ▶ On demand migration

This function is initiated when a DFSMShsm-managed volume exceeds a specified threshold. This function differs from interval migration because on demand migration does not perform the top-of-the-hour processing to check the volume utilization. Instead, it is notified by SMS when a volume exceeds the threshold, and the migration is performed against this single volume.

- ▶ Automatic recall

Automatically recalls user data sets back to DASD volumes, when referenced by the application.

- ▶ Space management by command

This function allows you to perform some space management tasks, such as migration and class transition by command, without needing to wait for the next automatic space management.

It is possible to have more than one z/OS image sharing a DFSMShsm policy. In this case, one of the DFSMShsm images is the primary host and the others are secondary. The primary HSM host is identified by HOST= in the HSM startup and is responsible for these tasks:

- ▶ Hourly space checks
- ▶ During auto backup: CDS backup, backup of ML1 data sets to tape
- ▶ During auto dump: Expiration of dump copies and deletion of excess dump VTOC copy data sets
- ▶ During secondary space management (SSM): Cleanup of MCDS, migration volumes, and L1-to-L2 migration

If you are running your z/OS HSM images in sysplex (parallel or basic), you can use *secondary host promotion* to allow a secondary image to assume the primary image's tasks if the primary host fails.

SMS Management Class for tape

With z/OS 2.3, new management class attributes were added to allow tape retention management through SMS management class. The new fields can be used by DFSMSrmm to control data set and tape retention, creating a single policy to manage both disk and tape data.

The following tape attributes were included in management class to allow tape processing:

► Tape volumes attributes

- ‘Retention Method’

The retention method is an attribute of the volume. All volumes in a multivolume set have the same retention method. All data sets on a volume are managed with the same retention method as the volume on which they reside. Retention method is only used if writing to the first data set of the volume chain.

- ‘VRSEL EXCLUDE’ (for VRSEL retention method)

When a data set is created on a tape volume managed by the EXPDT retention method, the data set VRSELEXCLUDE attribute is automatically set as “Y”. If the data set is created on a tape volume managed by the VRSEL retention method, the VRSELEXCLUDE attribute can be set by an SMS MGMTCLAS VRSELEXCLUDE attribute.

► Tape data set attributes

- ‘RETAIN BY’ (for EXPDT retention method)

With the EXPDT retention method, volumes and volume sets can be retained as individual volumes, as volume sets, or based on the expiration date of the first file. The volume attribute related to the retention of a multivolume set is the “RetainBy” attribute. Retain By is only assigned if writing to the first data set of the volume chain.

- ‘WHILECATALOG’ (for EXPDT retention method)

A cataloged data set can prevent the volume from expiring on its expiration date if it has WHILECATALOG(ON). The date might decrease after a data set with WHILECATALOG(UNTILEXPIRED) is uncataloged. The WHILECATALOG attribute is used to set the WHILECATALOG attribute for a new data set on an EXPDT volume only.

To assign a characteristic for a tape volume/data set using an SMS MC, the following conditions must be met:

- MC can apply to both system-managed tape and non-system-managed tape (using &ACSENVIR='RMMVRS').
- The MC should exist and V2R3 is required.
- The created data set should be SMS-supported and be on a TCDB volume.
- MGMTCLAS must be specified in JCL DD statement.
- When MCATTR=NONE, MC will not have any effect on an attribute.
- The following RMM PARMLIB options should be set:
SMSACS(YES) and MCATTR(ALL/VRSELXDI)

For more information about using management class to retain your tape data, see *z/OS DFSMSrmm Managing and Using Removable Media*, SC23-6873.

4.9 Removable media manager (DFSMSrmm)

In your enterprise, you store and manage your removable media in several types of media libraries. For example, in addition to your traditional tape library (a room with tapes, shelves, and drives), you might have several automated and manual tape libraries. You probably also have both onsite libraries and offsite storage locations, also known as *vaults* or *stores*.

With the DFSMSrmm functional component of DFSMS, you can manage your removable media as one enterprise-wide library (single image) across systems. Because of the need for global control information, these systems must have accessibility to shared DASD volumes. DFSMSrmm manages your installation's tape volumes and the data sets on those volumes. DFSMSrmm also manages the shelves where volumes reside in all locations except in automated tape library data servers.

DFSMSrmm manages all tape media, and other removable media you define to it. For example, DFSMSrmm can record the shelf location for optical disks and track their vital record status. However, it does not manage the objects on optical disks. The DFSMSrmm manages different levels of storage components and locations, including these:

- ▶ Library management

DFSMSrmm can manage a removable media library, including system-managed manual tape libraries and automated tape libraries. Non-system-managed or traditional tape libraries, including automated libraries such as a library under Basic Tape Library Support (BTLS) control.

- ▶ Shelf management

DFSMSrmm groups information about removable media by shelves into a central online inventory, and tracks the volumes on those shelves. DFSMSrmm can manage the shelf space that you define in your removable media library and in your storage locations.

- ▶ Volume management

DFSMSrmm manages the movement and retention of tape volumes throughout their lifecycle.

- ▶ Data set management

DFSMSrmm records information about the data sets on tape volumes. DFSMSrmm uses the data set information to validate volumes and to control the retention and movement of those data sets.

For more information about DFSMSrmm, see *z/OS DFSMSrmm Guide and Reference*, SC26-7404 and *z/OS DFSMSrmm Implementation and Customization Guide*, SC26-7405.

4.9.1 Resources managed by DFSMSrmm

A removable media library contains all the tape and optical volumes that are available for immediate use, including the shelves where they reside. A removable media library usually includes other libraries:

- ▶ *System-managed libraries*, such as automated or manual tape library data servers
- ▶ *Non-system-managed libraries*, containing the volumes, shelves, and drives not in an automated or a manual tape library data server

In the removable media library, you store your volumes in “shelves,” where each volume occupies a single shelf location. This shelf location is referred to as a *rack number* in the DFSMSrmm TSO subcommands and ISPF dialog. A rack number matches the volume’s external label.

DFSMSrmm uses the external volume serial number to assign a rack number when adding a volume, unless you specify otherwise. The format of the volume serial that you define to DFSMSrmm must be one to six alphanumeric characters. The rack number must be six alphanumeric or national characters.

System-managed tape library

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

- ▶ An *automated tape library* is a device that consists of robotic components, cartridge storage areas (or shelves), tape subsystems, and controlling hardware and software. It also includes the set of tape volumes that are in the library and can be mounted on the library tape drives.
- ▶ A *manual tape library* is a set of tape drives and the set of system-managed volumes the operator can mount on those drives. The manual tape library provides more flexibility, enabling you to use various tape volumes in a manual tape library. Unlike the automated tape library, the manual tape library does not use the library manager. With the manual tape library, a human operator responds to mount messages that are generated by the host and displayed on a console.

You can have several automated tape libraries or manual tape libraries. You use an installation-defined library name to define each automated tape library or manual tape library to the system. DFSMSrmm treats each system-managed tape library as a separate location or destination.

Non-system-managed tape library

A non-system-managed tape library consists of all the volumes, shelves, and drives not in an automated tape library or manual tape library. You might know this library as the traditional tape library that is not system-managed. DFSMSrmm provides complete tape management functions for the volumes and shelves in this traditional tape library. Volumes in a non-system-managed library are defined by DFSMSrmm as being “shelf-resident.”

All tape media and drives that are supported by z/OS are supported in this environment. Using DFSMSrmm, you can fully manage all types of tapes in a non-system-managed tape library.

Storage location

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

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

You can also use the DFSMSrmm built-in storage locations: LOCAL, DISTANT, and REMOTE. Although the names of these locations imply their purpose, they do not mandate their actual location. All volumes can be in the same or separate physical locations.

For example, an installation can have the LOCAL storage location onsite as a vault in the computer room, the DISTANT storage location can be a vault in an adjacent building, and the REMOTE storage location can be a secure facility across town or in another state.

DFSMSSrmm provides shelf-management for storage locations so that storage locations can be managed at the shelf location level.

4.9.2 Libraries and locations

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

DFSMSSrmm automatically records information about data sets on tape volumes so that you can manage the data sets and volumes more efficiently. When all the data sets on a volume have expired, the volume can be reclaimed and reused. You can optionally move volumes that are to be retained to another location.

DFSMSSrmm helps you manage your tape volumes and shelves at your primary site and storage locations by recording information in a DFSSMSSrmm control data set.

Managing libraries and storage locations

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

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

Figure 4-11 shows a sample library and location management performed by DFSMSrmm.

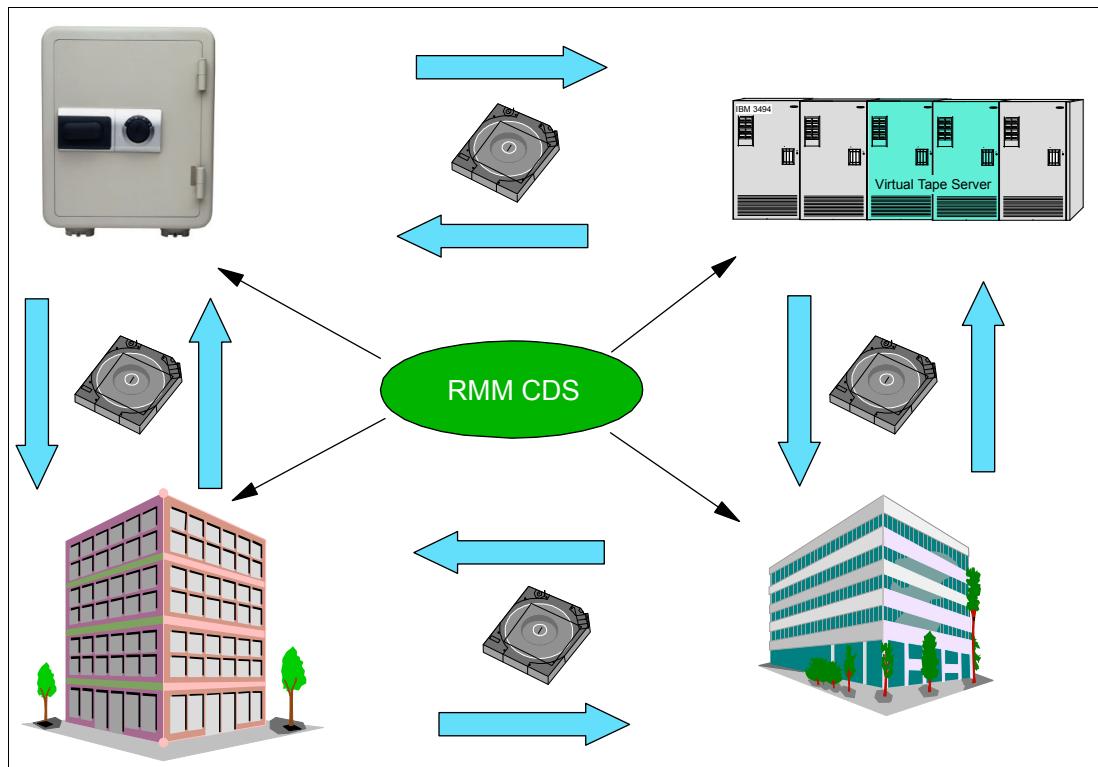


Figure 4-11 DFSMSrmm: Managing libraries and storage locations

DFSMSrmm helps you manage the movement of your volumes and retention of your data over their full life, from initial use to the time they are retired from service. These are some of the functions that DFSMSrmm performs:

- ▶ Automatically initializing and erasing volumes
- ▶ Recording information about volumes and data sets as they are used
- ▶ Expiration processing
- ▶ Identifying volumes with high error levels that require replacement

To make full use of all of the DFSMSrmm functions, you specify installation setup options and define retention and movement policies. DFSMSrmm provides you with utilities to implement the policies that you define. If you want to learn more about DFSMSrmm functions, check *DFSMSrmm Primer*, SG24-5983.

4.9.3 Managing data

Besides the library and location management, DFSMSrmm also allows tape and data set retention management. You can define policies to retain your data, or allow the users to specify the retention on JCL. DFSMSrmm also provides a strong reporting capability, so you can use this information to plan your tape environment based on its usage and retention requirements.

You can control your data set expiration by the following methods:

- ▶ Vital records specification (VRS)

You can set VRS to identify the retention and movement policies, including how long and where you want to keep data sets or volumes. You also use them to define how volumes are to be moved among the libraries that DFSMSrmm supports, and the storage locations defined for vital records and disaster recovery purposes.

- ▶ EXPDT, RETPD

Data sets that are not controlled by VRSs can be controlled by EXPDT and RETPD parameters that are defined by the user on JCL during data set creation.

- ▶ Management Class

The SMS management class also includes information so DFSMSrmm can manage tape retention. For more information about management class and tape retention, see “SMS Management Class for tape” on page 71.

- ▶ External Data Manager (EDM)

DFSMSrmm also provides the option for the tapes to be managed externally. Some programs such as DFMSHsm perform their own tape management, and are capable of identifying when tapes are no longer required and can be scratched. Allowing a tape to be EDM ensures only the application that created the tape will have the authorization to scratch the tape.

When performing tape and data set management, DFSMSrmm also considers the time that the data was created, to ensure that the data is deleted only after the correct retention was reached.



System-managed storage

As your business expands, so does your need for storage to hold your applications and data, and so does the cost of managing that storage. Storage cost includes more than the price of the hardware, with the highest cost being the people needed to perform storage management tasks.

If your business requires transaction systems, the batch window can also be a high cost. Additionally, you must pay for staff to install, monitor, and operate your storage hardware devices, for electrical power to keep each piece of storage hardware cool and running, and for floor space to house the hardware. Removable media, such as optical and tape storage, cost less per gigabyte (GB) than online storage, but they require additional time and resources to locate, retrieve, and mount.

To allow your business to grow efficiently and profitably, you need to find ways to control the growth of your information systems and use your current storage more effectively.

With these goals in mind, this chapter covers these topics:

- ▶ A description of the z/OS storage-managed environment
- ▶ An explanation of the benefits of using a system-managed environment
- ▶ An overview of how SMS manages a storage environment based on installation policies
- ▶ A description of how to maintain an SMS configuration
- ▶ A description of how to use Interactive Storage Management Facility (ISMF), an interface for defining and maintaining storage management policies

This chapter includes the following sections:

- ▶ DFSMS and storage management
- ▶ DFSMS constructs
- ▶ Maintaining and monitoring SMS policies
- ▶ Interactive Storage Management Facility

5.1 DFSMS and storage management

Storage management involves data set allocation, placement, monitoring, migration, backup, recall, recovery, and deletion. These activities can be done either manually or by using automated processes.

5.1.1 Managing storage with DFSMS

The Data Facility Storage Management Subsystem (DFSMS) comprises the base z/OS operating system and performs the essential data, storage, program, and device management functions of the system. DFSMS is the central component of both system-managed and non-system-managed storage environments.

The DFSMS software product, together with hardware products and installation-specific requirements for data and resource management, comprises the key to system-managed storage in a z/OS environment.

The heart of DFSMS is the storage management subsystem (SMS). Using SMS, the storage administrator defines policies that automate the management of storage and hardware devices. These policies describe data allocation characteristics, performance and availability goals, backup and retention requirements, and storage requirements for the system. SMS governs these policies for the system and the ISMF provides the user interface for defining and maintaining the policies.

DFSMS is a set of products, and one of these products, DSFMSdfp, is mandatory for running z/OS. DFSMS is the central component of both system-managed and non-system-managed storage environments.

DFSMS environment

The DFSMS environment consists of a set of hardware and IBM software products that together provide a system-managed storage solution for z/OS installations.

DFSMS uses a set of constructs, user interfaces, and routines (using the DFSMS products) that allow the storage administrator to better manage the storage system. The core logic of DFSMS, such as the automatic class selection (ACS) routines, ISMF code, and constructs, is located in DSFMSdfp. DFSMSHsm and DFSMSdss are involved in the management class construct.

Figure 5-1 shows the products that comprise DFSMS environment, and some of their functions.

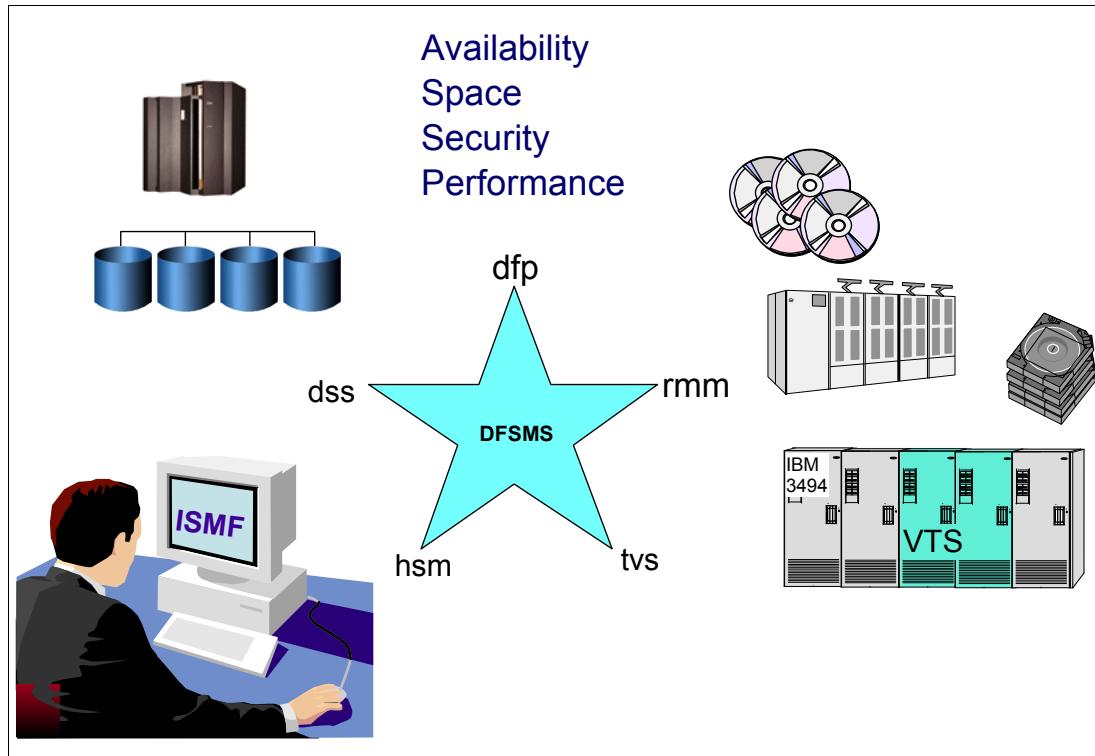


Figure 5-1 Managing storage with DFSMS

In this environment, the Resource Access Control Facility (RACF) and Data Facility Sort (DFSORT) products complement the functions of the base operating system. RACF provides resource security functions, and DFSORT adds the capability for faster and more efficient sorting, merging, copying, reporting, and analyzing of business information.

The DFSMS environment is also called the SMS environment.

5.1.2 Goals and benefits of system-managed storage

With the SMS, you can define performance goals and data availability requirements, create model data definitions for typical data sets, and automate data backup. Based on installation policies, SMS can automatically assign those services and data definition attributes to data sets when they are created. IBM storage management-related products determine data placement, manage data backup, control space usage, and provide data security.

The following are the goals of system-managed storage:

- ▶ To improve the use of the storage media (for example, by reducing out-of-space abends and providing a way to set a free-space requirement).
- ▶ To reduce the labor involved in storage management by centralizing control, automating tasks, and providing interactive controls for storage administrators.
- ▶ To reduce the user's need to be concerned with the physical details, performance, space, and device management. Users can focus on using data, instead of managing data.

Simplified data allocation

System-managed storage enables users to simplify their data allocations. For example, without using the Storage Management Subsystem, a z/OS user must specify the unit and volume on which the system is to allocate the data set. The user also must calculate the amount of space required for the data set in terms of tracks or cylinders. This requirement means the user must know the track size of the device that will contain the data set.

With system-managed storage, users can allow the system to select the specific unit and volume for the allocation. They can also specify size requirements in terms of megabytes or kilobytes. This feature means the user does not need to know anything about the physical characteristics of the devices in the installation.

If the user does not provide the unit parameter for allocation, DFSMS uses the default device geometry to convert space allocation requests into device-independent units, such as KBs and MBs. This quantity is then converted back into device-dependent tracks based on the selected device geometry.

Improved allocation control

System-managed storage enables you to set a requirement for free space across a set of direct access storage device (DASD) volumes. You can then provide adequate free space to avoid out-of-space abends. The system automatically places data on a volume that contains adequate free space. DFSMS offers relief to avoid out-of-space conditions. You can also set a threshold for scratch tape volumes in tape libraries to ensure that enough cartridges are available in the tape library for scratch mounts.

Improved input/output performance management

System-managed storage enables you to improve DASD I/O performance across the installation. At the same time, it reduces the need for manual tuning by defining performance goals for each class of data. You can use cache statistics that are recorded in System Management Facilities (SMF) records to help evaluate performance. You can also improve sequential performance by using *extended* sequential data sets. The DFSMS environment makes the most effective use of the caching abilities of storage controllers.

Automated DASD space management

System-managed storage enables you to automatically reclaim space that is allocated to old and unused data sets or objects. You can define policies that determine how long an unused data set or object will be allowed to be on primary storage (storage devices used for your active data). You can have the system remove obsolete data by migrating the data to other DASD, tape, or optical volumes, or you can have the system delete the data. You can also release allocated but unused space that is assigned to new and active data sets.

Automated tape space management

Mount Management System-managed storage lets you fully use the capacity of your tape cartridges and automate tape mounts. Using tape mount management (TMM) methodology, DFSMShsm can fill tapes to their capacity. With 3490E, 3590, 3591, and 3592 tape devices, you can increase the amount of data that can be written on a single tape cartridge.

Tape System-managed storage lets you use the device technology of new devices without having to change the job control language (JCL) UNIT parameter. In a multi-library environment, you can select the drive based on the library where the cartridge or volume resides. You can use the IBM Automated Tape Library (4500) or IBM Tape Server 7700 family to automatically mount tape volumes and manage the inventory in a physical or virtual tape library. Similar functionality is available in a system-managed manual tape library. If you are not using SMS for tape management, you can still access the tape libraries.

Automated optical space management

System-managed storage enables you to fully use the capacity of your optical cartridges and to automate optical mounts. Using a 3995 Optical Library Dataserver, you can automatically mount optical volumes and manage the inventory in an automated optical library.

Improved data availability management

System-managed storage enables you to provide separate backup requirements to data on the same DASD volume. Thus, you do not have to treat all data on a single volume the same way.

You can use DFSShsm to automatically back up your various types of data sets and use point-in-time copy to maintain access to critical data sets while they are being backed up. Concurrent copy, virtual concurrent copy, SnapShot, and FlashCopy, along with backup-while-open, have an added advantage in that they avoid invalidating a backup of a CICS VSAM KSDS due to a control area or control interval split.

You can also create a logical group of data sets so that the group is backed up at the same time to allow recovery of the application defined by the group. This process is done with the aggregate backup and recovery support (ABARS) provided by DFSShsm.

Simplified conversion of data to other device types

System-managed storage enables you to move data to new volumes without requiring users to update their job control language (JCL). Because users in a DFSSMS environment do not need to specify the unit and volume that contains their data, it does not matter to them if their data is on a specific volume or device type. This feature allows you to easily replace old devices with new ones.

You can also use system-determined block sizes to automatically reblock physical sequential and partitioned data sets that can be reblocked.

Establishing installation standards

Establishing standards such as naming conventions and allocation policies helps you to manage storage more efficiently and improves service to your users. With them, your installation is better prepared to make a smooth transition to system-managed storage.

You can negotiate with your user group representatives to agree on the specific policies for the installation, how soon you can implement them, and how strongly you enforce them. You can also simplify storage management by limiting the number of data sets and volumes that cannot be system-managed.

5.1.3 Service level objectives

To allow your business to grow efficiently and profitably, you want to find ways to control the growth of your information systems and use your current storage more effectively.

In an SMS-managed storage environment, your enterprise establishes centralized policies for how to use your hardware resources. These policies balance your available resources with your users' requirements for data availability, performance, space, and security.

The policies that are defined in your installation represent decisions about your resources, such as the following:

- ▶ What performance objectives are required by the applications that access the data
 - Based on these objectives, you can try to better use cache data striping. By tracking data set I/O activities, you can make better decisions about data set caching policies and improve overall system performance. For object data, you can track transaction activities to monitor and improve OAM's performance.
- ▶ When and how to back up data (incremental or total)
 - Determine the backup frequency, the number of backup versions, and the retention period by consulting user group representatives. Be sure to consider whether certain data backups need to be synchronized. For example, if the output data from application A is used as input for application B, you must coordinate the backups of both applications to prevent logical errors in the data when they are recovered.
- ▶ Whether data sets are to be kept available for use during backup or copy
 - You can store backup data sets on DASD or tape (this does not apply to objects). Your choice depends on how fast the data needs to be recovered, media cost, operator cost, floor space, power requirements, air conditioning, the size of the data sets, and whether you want the data sets to be portable.
- ▶ How to manage backup copies kept for disaster recovery (locally or in a vault)
 - Back up related data sets in aggregated tapes. Each application is to have its own, self-contained aggregate of data sets. If certain data sets are shared by two or more applications, you might want to ensure application independence for disaster recovery by backing up each application that shares the data. This independance is especially important for shared data in a distributed environment.
- ▶ What to do with data that is obsolete or seldom used
 - Data is obsolete when it has exceeded its expiration dates and is no longer needed. To select obsolete data for deletion using DFSMSdss, issue the DUMP command with the DELETE parameter, and force OUTDDNAME to DUMMY.

The purpose of a backup plan is to ensure the prompt and complete recovery of data. A well-documented plan identifies data that requires backup, the levels required, responsibilities for backing up the data, and methods to be used.

5.1.4 SMS configuration

An SMS configuration is composed of these elements:

- ▶ A set of data classes, management classes, storage classes, and storage groups
- ▶ ACS routines to assign the classes and groups
- ▶ Optical library and drive definitions (optional)
- ▶ Tape library definitions (optional)
- ▶ Aggregate group definitions (optional)
- ▶ SMS base configuration, that contains information such as the following:
 - Default management class
 - Default device geometry
 - The systems in the installation for which the subsystem manages storage

SMS complex (SMSplex)

A collection of systems or system groups that share a common configuration is called an SMS complex. All systems in an SMS complex share control data sets called ACDS and a COMMDS. The systems or system groups that share the configuration are defined to SMS in the SMS base configuration. The ACDS and COMMDS must be on a shared volume that is accessible to all systems in the SMS complex.

The SMS configuration is stored in SMS control data sets, which are VSAM linear data sets. You must define the control data sets before activating SMS. SMS uses the following types of control data sets:

- ▶ Source control data set (SCDS)
- ▶ Active control data set (ACDS)
- ▶ Communications data set (COMMDS)

5.1.5 SMS control data sets

Before you use DFSMS in your systems, you need to understand the basic data sets that are necessary to run your SMS environment. All the SMS configuration created to allocate and manage your data, as well as volume and other system information are stored in control data sets, described next.

Source control data set

The SCDS contains SMS classes, groups, and translated ACS routines that define a *single storage management policy*, called an SMS configuration. You can have several SCDSs, but only one can be used to activate the SMS configuration.

Use the SCDS to develop and test your SMS configuration, but before activating a configuration, retain at least one prior configuration if you need to regress to it because of an error. The SCDS is never used to manage allocations.

Active control data set

The ACDS is the system's active copy of the current SCDS. When you activate a configuration, SMS copies the existing configuration from the specified SCDS into the ACDS. By using copies of the SMS classes, groups, volumes, optical libraries, optical drives, tape libraries, and ACS routines rather than the originals, you can change the current storage management policy without disrupting it. For example, while SMS uses the ACDS, you can perform these tasks:

- ▶ Create a copy of the ACDS
- ▶ Create a backup copy of an SCDS
- ▶ Modify an SCDS
- ▶ Define a new SCDS

Generally, have extra ACDSs in case a hardware failure causes the loss of your primary ACDS. These extra ACDSs must be on a shared device that is accessible to all systems that share the SMS configuration to ensure that they share a common view of the active configuration. Do not place the ACDS on the same device as the COMMDS or SCDS. Both the ACDS and COMMDS are needed for SMS operation across the complex. Separation protects against hardware failure. Also, create a backup ACDS in case of hardware failure or accidental data loss or corruption.

Communications data set

The COMMDS data set contains the name of the ACDS and storage group volume statistics. It enables communication between SMS systems in a multisystem environment. The COMMDS also contains space statistics, SMS status, and MVS status for each system-managed volume.

The COMMDS must reside on a shared device accessible to all systems. However, do not allocate it on the same device as the ACDS. Create a spare COMMDS in case of a hardware failure or accidental data loss or corruption. SMS activation fails if the COMMDS is unavailable.

Figure 5-2 shows the relationship between SMS control data sets and DFSMS.

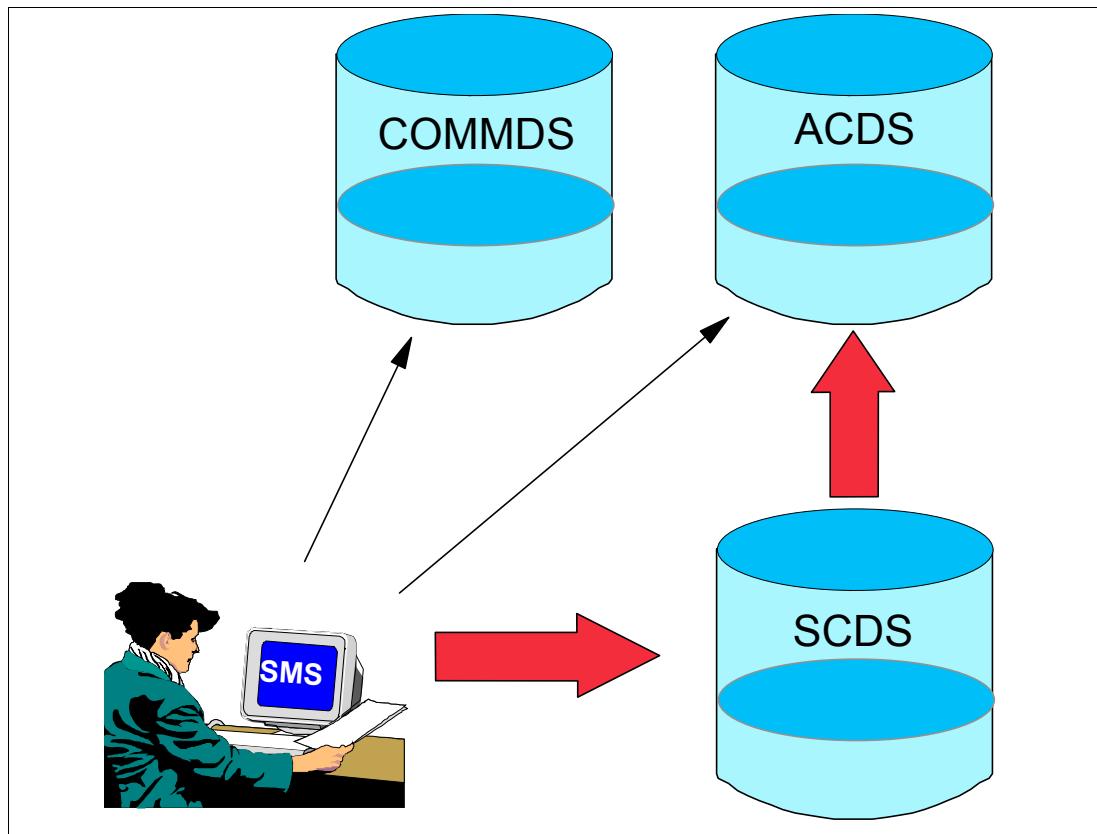


Figure 5-2 SMS control data sets

5.1.6 Data sets eligible for SMS management

The following are common types of data that can be system-managed. For details about how these data types can be system-managed using SMS storage groups, see *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407.

Temporary data Data sets used only for the duration of a job, job step, or terminal session, and then deleted. These data sets can range in size from small to very large.

Permanent data Sequential and partitioned data sets, z/OS UNIX files (such as zNFS, zFS, and so on), basic, or large format data sets can all be SMS-managed if they are catalogued in ICF catalogs, are not unmovable, or have user written access method.

VSAM data	Data organized with all VSAM types (KSDS, ESDS, RRDS, and LDS), including VSAM data sets that are part of an existing database.
Multivolume data	Data sets that span more than one volume.
System data	Some system data sets can be system-managed if they are uniquely named. These data sets include user catalogs. Place other system data sets on non-system managed volumes. The system data sets that are allocated at MVS system initialization are not system-managed because the SMS address space is not active at initialization time.
Object data	Also known as byte-stream data, this data is used in specialized applications such as image processing, scanned correspondence, and seismic measurements. Object data typically has no internal record or field structure and, after it is written, the data is not changed or updated. However, the data can be referenced many times during its lifetime.

5.1.7 Data sets non-eligible for SMS management

DFSMS is capable of managing most data set types available on z/OS with very few exceptions. The following data sets cannot be SMS-managed:

- ▶ Uncataloged data sets
- ▶ Unmovable data sets
- ▶ Data sets with absolute track allocations
- ▶ Tape data sets, except tape data sets on mountable volumes contained in an automated tape library (ATL) data server
- ▶ Data sets with user-written access method

For information about identifying and converting non-eligible data sets, see *z/OS DFSMSdss Storage Administration Guide*, SC35-0423.

5.1.8 Implementing DFSMS

You can implement SMS to fit your specific needs. You do not have to implement and use all of the SMS functions. Rather, you can implement the functions that you are most interested in first. For example, you might choose to create this setup:

- ▶ Set up a storage group to only use the functions provided by extended format data sets, such as striping, system-managed buffering (SMB), partial release, and so on.
- ▶ Put data in a pool of one or more storage groups and assign them policies at the storage group level to implement DFSMShsm operations in stages.
- ▶ Use VSAM record-level sharing (RLS).

This book presents an overview of the functions and features available on DFSMS. For a better understanding on the necessary steps to implement SMS into a system, as well as planning considerations, security requirements, and z/OS configuration, see *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407.

5.2 DFSMS constructs

Data management through DFSMS is performed by the assignment of SMS constructs, which defines attributes for space management performance, and availability requirements for data sets. The storage administrator uses these classes and routines:

Data class	Data classes are used to define model allocation characteristics for data sets.
Storage class	Storage classes are used to define performance and availability goals.
Management class	Management classes are used to define backup and retention requirements.
Storage group	Storage groups are used to create logical groups of volumes to be managed as a unit.
ACS routines	Automatic class selection (ACS) routines are used to assign class and storage group definitions to data sets and objects.

Figure 5-3 shows that SMS constructs are assigned to a data set through the ACS routines, and how each SMS construct provides input to data set management through its lifecycle.

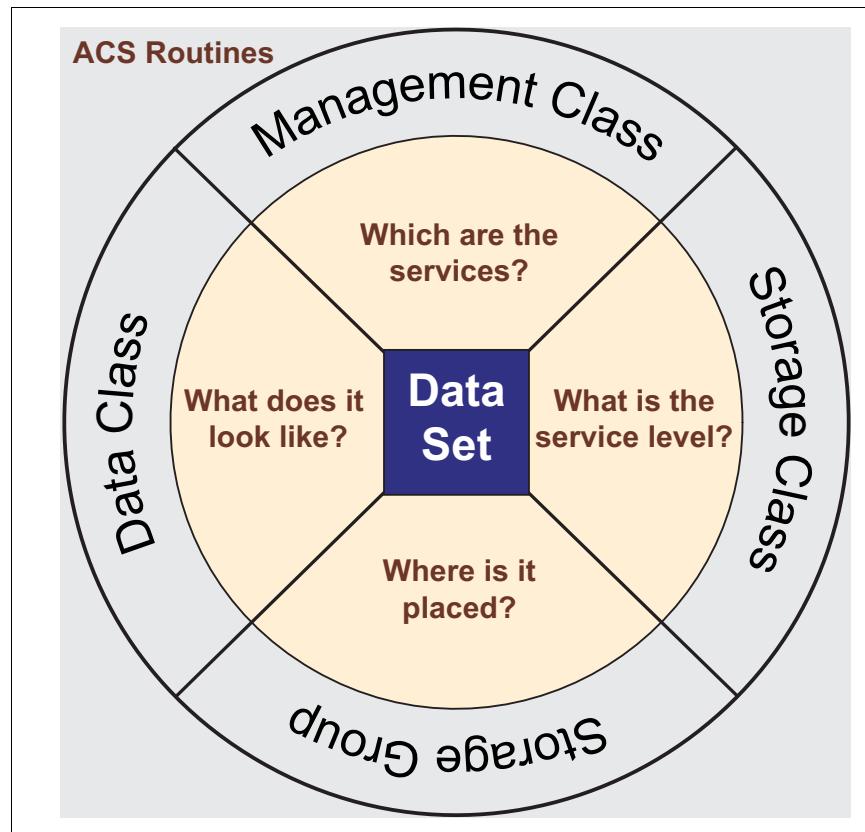


Figure 5-3 Constructs assignment and description

By creating different constructs, you can use the ACS routines to assign different data management aspects for each group of data sets. For example, the administrator can define one storage class for data entities requiring high performance, and another for those requiring standard performance. The administrator then writes ACS routines that use naming conventions or other criteria of your choice to automatically assign the classes that have been defined to data as that data is created. These ACS routines can then be validated and tested.

When the ACS routines are started and the classes (also referred to as constructs) are assigned to the data, SMS uses the policies defined in the classes to apply to the data for the life of the data. Additionally, devices with various characteristics can be pooled together into storage groups so that new data can be automatically placed on devices that best meet the needs for the data.

DFSMS facilitates all of these tasks by providing menu-driven panels with ISMF. ISMF panels make it easy to define classes, test and validate ACS routines, and perform other tasks to analyze and manage your storage. Many of these functions are available in batch through the NaviQuest tool.

5.2.1 Data class

A *data class* is a collection of allocation and space attributes that you define. It is used when data sets are created. You can simplify data set allocation for the users by defining data classes that contain standard data set allocation attributes. You can use data classes with *both* system-managed and non-system-managed data sets. However, various data class characteristics, like extended format, are only available for system-managed data sets.

Data class attributes define space and data characteristics that are normally specified on JCL DD statements, TSO/E ALLOCATE command, IDCAMS DEFINE commands, and dynamic allocation requests. For tape data sets, data class attributes can also specify the type of cartridge and recording method, and if the data is to be compacted. Users then only need to specify the appropriate data classes to create standardized data sets.

You can assign a data class by using these techniques:

- ▶ The DATACLAS parameter of a JCL DD statement using either the ALLOCATE or DEFINE command.
- ▶ Data class ACS routine to automatically assign a data class when the data set is being created. For example, you can use Data Class to set large data sets as extended-format or large format data sets. Data sets with similar allocation requirements, and can all be assigned the same data class.

You can override various data set attributes assigned in the data class, but you cannot change the data class name after it has been assigned to your data set.

Even though data class is optional, generally assign data classes to system-managed and non-system-managed data. Although the data class is not used after the initial allocation of a data set, the data class name is kept in the catalog entry for system-managed data sets for future reference.

Note: The data class name is not saved for non-system-managed data sets, although the allocation attributes in the data class are used to allocate the data set.

For objects on tape, avoid assigning a data class through the ACS routines. To assign a data class, specify the name of that data class on the SETOAM command.

If you change a data class definition, the changes only affect new allocations. Existing data sets allocated with the data class are not changed.

Data class attributes

You can specify the data class space attributes to control DASD space waste; for example:

- ▶ The primary space value is to specify the total amount of space initially required for output processing. The secondary allocation allows automatic extension of additional space as the data set grows and does not waste space by overallocating the primary quantity. You can also use data class space attributes to relieve users of the burden of calculating how much primary and secondary space to allocate.
- ▶ The COMPACTON attribute specifies whether data is to be compressed on DASD if the data set is allocated in the extended format. The COMPACTON attribute alone also allows you to use the improved data recording capability (IDRC) of your tape device when allocating tape data sets. To use the COMPACTON attribute, the data set *must* be system-managed because this attribute demands an extended format data set.
- ▶ The following attributes are used for tape data sets only:
 - MEDIA TYPE allows you to select the mountable tape media cartridge type.
 - RECORDING TECHNOLOGY allows you to select the format to use when writing to that device.
 - The read-compatible special attribute indicator in the tape device selection information (TDSI) allows an 18-track tape to be mounted on a 36-track device for read access. The attribute increases the number of devices that are eligible for allocation when you are certain that no more data will be written to the tape.

For detailed information about specifying data class attributes, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

5.2.2 Storage class

A *storage class* is a collection of performance goals and availability requirements that you define. The storage class is used to select a device to meet those goals and requirements. Only *system-managed* data sets and objects can be assigned to a storage class. Storage classes free users from having to know about the physical performance characteristics of storage devices and manually placing their data on appropriate devices.

Some of the availability requirements that you specify to storage classes (such as dual copy) can only be met by DASD volumes with required features enabled.

Figure 5-4 shows storage class assignment according to performance and availability requirements.

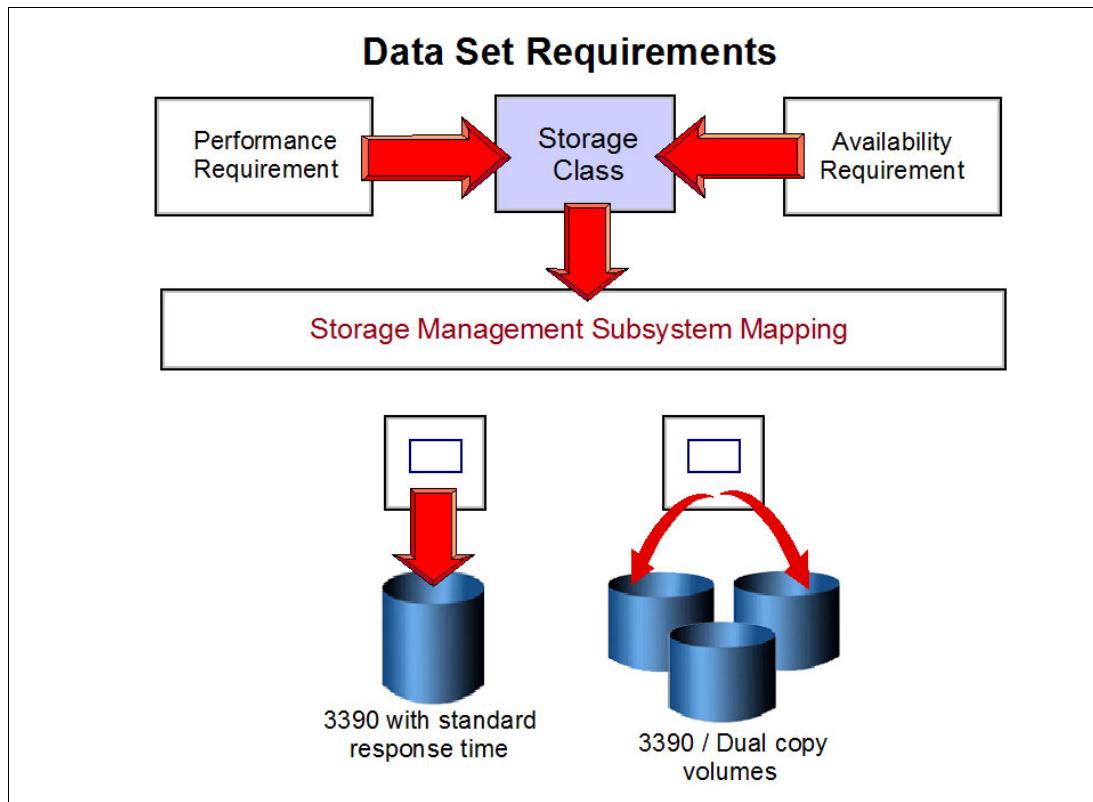


Figure 5-4 Storage Class assignment

With a storage class, you can assign a data set to dual copy volumes to ensure continuous availability for the data set. With dual copy, two current copies of the data set are kept on separate DASD volumes (by the control unit). If the volume containing the primary copy of the data set is damaged, the companion volume is automatically brought online and the data set continues to be available and current. Remote copy is similar, with the two volumes in distinct control units (generally remote).

You can use the ACCESSIBILITY attribute of the storage class to request that concurrent copy be used when data sets or volumes are backed up.

You can specify an I/O response time objective with storage class by using the millisecond response time (MSR) parameter. During data set allocation, the system attempts to select the closest available volume to the specified performance objective. Also, along the data set life, through the use MSR, DFSMS dynamically uses the cache algorithms as DASD fast write (DFW) and inhibit cache load (ICL) to reach the MSR target I/O response time. This DFSMS function is called *dynamic cache management*.

To assign a storage class to a new data set, you can use these techniques:

- ▶ The STORCLAS parameter of the JCL DD statement, or ALLOCATE or DEFINE command
- ▶ Storage class ACS routine

For *objects*, the system uses the performance goals you set in the storage class to place the object on DASD, optical, or tape volumes. The storage class is assigned to an object when it is stored or when the object is moved. The ACS routines can override this assignment.

Note: If you change a storage class definition, the changes affect the performance service levels of *existing* data sets that are assigned to that class when the data sets are later opened. However, the definition changes do not affect the location or allocation characteristics of existing data sets.

For more information about how to set up a storage class, and what attributes it defines, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

5.2.3 Management class

A *management class* is a collection of management attributes that you define. The attributes defined in a management class are related to these characteristics:

- ▶ Expiration date
- ▶ Migration criteria
- ▶ GDG management
- ▶ Backup of data set
- ▶ Class Transition Criteria
- ▶ Aggregate backup

Figure 5-5 shows the management class processing cycle based on data set attributes.

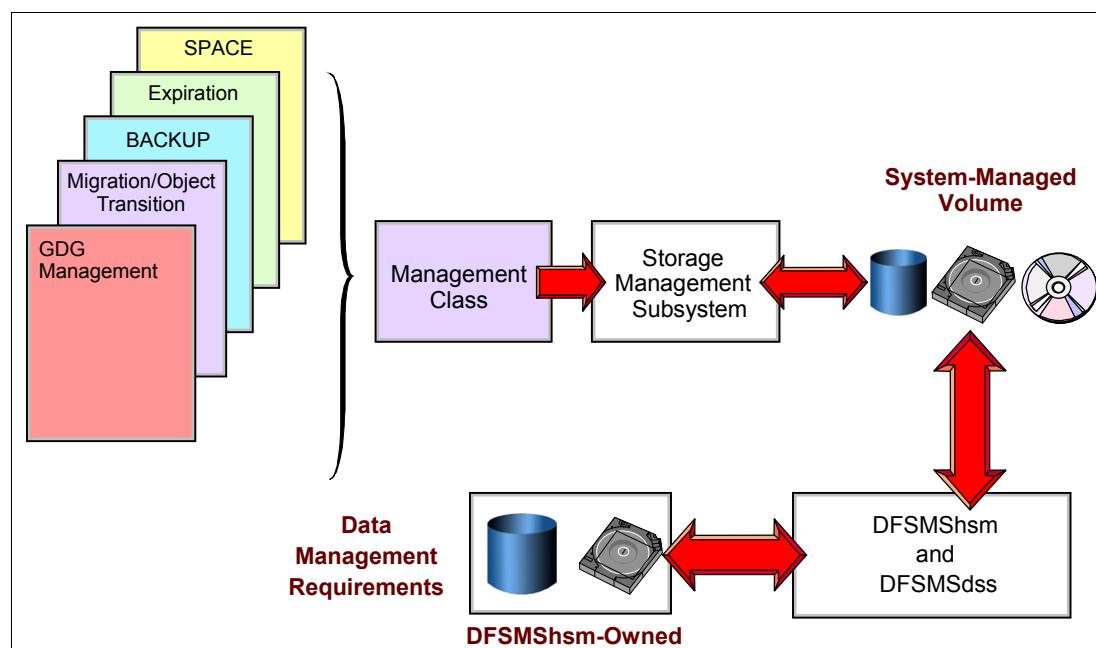


Figure 5-5 Management Class processing cycle

Management classes let you define management requirements for individual data sets, rather than defining the requirements for entire volumes. All the data set functions described in the management class are run by DFSMShsm and DFSMSdss programs.

To assign a management class to a new data set, you can use these techniques:

- ▶ The MGMTCLAS parameter of the JCL DD statement, ALLOCATE or DEFINE command
- ▶ The management class ACS routine to automatically assign management classes to new data sets

The ACS routine can override the management class specified in JCL, or ALLOCATE or DEFINE command. You cannot override management class attributes through JCL or command parameters.

If you do not explicitly assign a management class to a system-managed data set or object, the system uses the *default management class*. You can define your own default management class when you define your SMS base configuration.

Note: If you change a management class definition, the changes affect the management requirements of *existing data sets* and *objects* that are assigned that class. You can reassign management classes when data sets are renamed.

For objects, you can perform these tasks:

- ▶ Assign a management class when it is stored
- ▶ Assign a new management class when the object is moved
- ▶ Change the management class by using the OAM application programming interface (OSREQ CHANGE function)

The ACS routines can override this assignment for objects. For more information about how to set up a management class, and what attributes it defines, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

Management class functions

By classifying data according to management requirements, an installation can define unique management classes to fully automate data set and object management, as in these examples:

- ▶ Control the migration and backup of CICS user databases, DB2 user databases, and archive logs.
- ▶ Apply different retention periods for test systems and their associated data sets.
- ▶ Keep backup copies of a deleted data set for a specified number of days.
- ▶ Specify that DB2 image copies, IMS image copies, and change accumulation logs be written to primary volumes and then migrated directly to migration level 2 tape volumes.
- ▶ For objects, define when an object is eligible for a change in its performance objectives or management characteristics. For example, after a certain number of days, an installation might want to move an object from a high-performance DASD volume to a slower optical volume.

Management class can also be used to specify that the object is to have a backup copy made when the OAM Storage Management Component (OSMC) is running.

When changing a management class definition, the changes affect the management requirements of existing data sets and objects that are assigned to that class.

5.2.4 Storage group

A *storage group* is a collection of storage volumes and attributes that you define. The collection can be a group of these types of volumes:

- ▶ System paging volumes
- ▶ DASD volumes
- ▶ Tape volumes
- ▶ Optical volumes
- ▶ Combination of DASD and optical volumes that look alike
- ▶ DASD, tape, and optical volumes treated as a single object storage hierarchy

Storage groups, along with storage classes, help reduce the requirement for users to understand the physical characteristics of the storage devices that contain their data.

In a tape environment, you can also use tape storage groups to direct a new tape data set to an automated or manual tape library.

DFSMShsm uses various storage group attributes to determine whether the volumes in the storage group are eligible for automatic space or availability management.

Figure 5-6 on page 95 shows an example of how an installation can group storage volumes according to their objective. Note the following characteristics in this example:

- ▶ SMS-managed DASD volumes are grouped into storage groups so that primary data sets, large data sets, DB2 data, IMS data, and CICS data are all separated.
- ▶ The VIO storage group uses system paging volumes for small temporary data sets.
- ▶ The TAPE storage groups are used to group tape volumes that are held in tape libraries.
- ▶ The OBJECT storage group can span optical, DASD, and tape volumes.
- ▶ The OBJECT BACKUP storage group can contain either optical or tape volumes within one OAM invocation.
- ▶ Some volumes are not system-managed.
- ▶ Other volumes are owned by DFSMShsm for use in data backup and migration. DFSMShsm migration level 2 tape cartridges can be system-managed if you assign them to a tape storage group.

Figure 5-6 shows some storage groups definitions available based on different devices availability and usage purposes.

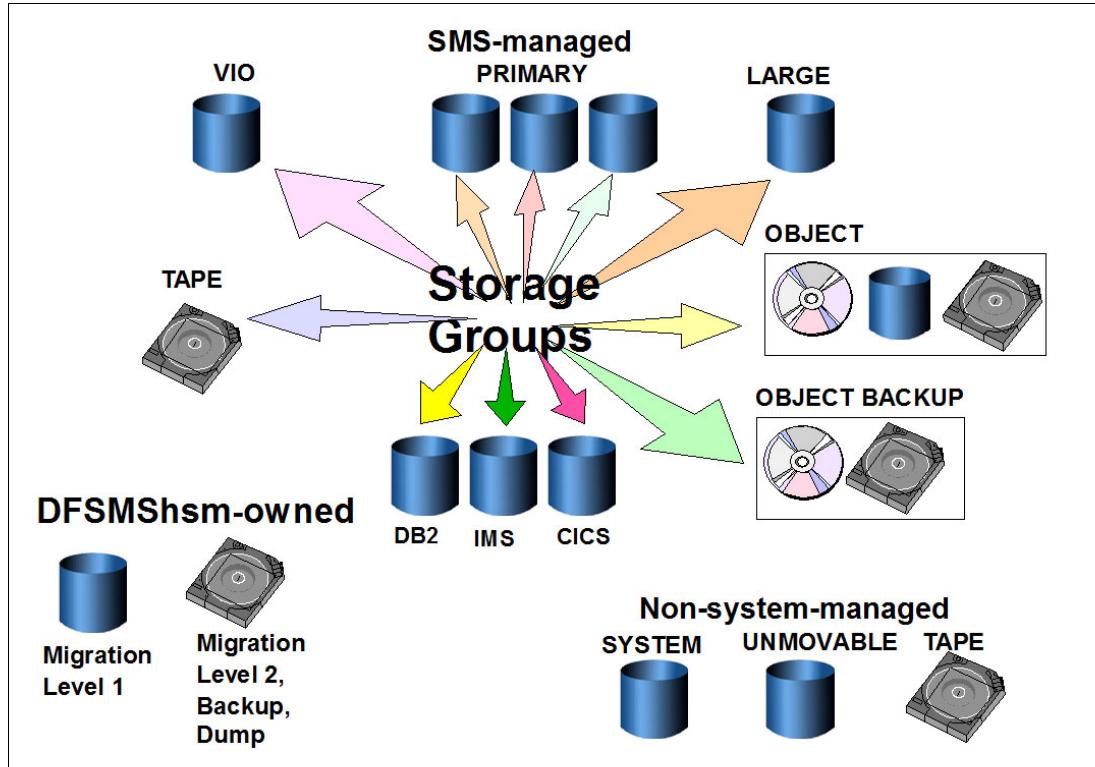


Figure 5-6 Different storage groups definitions and usage

Note: A storage group is assigned to a data set *only* through the storage group ACS routine. Users cannot specify a storage group when they allocate a data set, although they *can* specify a unit and volume.

Whether to accept a user's unit and volume request is an installation decision, but you should generally discourage users from directly requesting specific devices. It is more effective for users to specify the logical storage requirements of their data by storage and management class, which the installation can then verify in the ACS routines.

Objects have two types of storage groups: **OBJECT** and **OBJECT BACKUP**. An **OBJECT** storage group is assigned by OAM when the object is stored. The storage group ACS routine can override this assignment. There is only one **OBJECT BACKUP** storage group, and all backup copies of all objects are assigned to this storage group.

SMS volume selection

SMS determines which volumes are used for data set allocation by developing a list of all volumes from the storage groups assigned by the storage group ACS routine. Volumes are then either removed from further consideration or flagged as the following types:

- | | |
|------------------|---|
| Primary | Volumes online, below threshold, that meet all the specified criteria in the storage class. |
| Secondary | Volumes that do not meet all the criteria for primary volumes. |

Tertiary	When the number of volumes in the storage group is less than the number of volumes that are requested.
Rejected	Volumes that do not meet the required specifications. They are not candidates for selection.

SMS starts volume selection from the primary list. If no volumes are available, SMS selects from the secondary, and, if no secondary volumes are available, SMS selects from the tertiary list.

SMS interfaces with the system resources manager (SRM) to select from the eligible volumes in the primary list. SRM uses device delays as one of the criteria for selection, and does not prefer a volume if it is already allocated in the job step. This technique is useful for batch processing when the data set is accessed immediately after creation.

SMS does not use SRM to select volumes from the secondary or tertiary volume lists. It uses a form of randomization to prevent skewed allocations in instances such as when new volumes are added to a storage group, or when the free space statistics are not current on volumes.

For a striped data set, when multiple storage groups are assigned to an allocation, SMS examines each storage group and selects the one that offers the largest number of volumes attached to unique control units. This is called *control unit separation*. After a storage group has been selected, SMS selects the volumes based on available space, control unit separation, and performance characteristics if they are specified in the assigned storage class.

For more information about how to set up a storage group, and what attributes it defines, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

5.2.5 Using aggregate backup and recovery support

ABARS, also called *application backup application recovery support*, is a command-driven process to back up and recover any user-defined group of data sets that are vital to your business. An *aggregate group* is a collection of related data sets and control information that has been pooled to meet a defined backup or recovery strategy. If a disaster occurs, you can use these backups at a remote or local site to recover critical applications.

The user-defined group of data sets can be those belonging to an application, or any combination of data sets that you want treated as a separate entity. Aggregate processing enables you to perform these tasks:

- ▶ Back up and recover data sets by application to enable business to resume at a remote site if necessary
- ▶ Move applications in a non-emergency situation along with personnel moves or workload balancing
- ▶ Duplicate a problem at another site for problem determination

You can use aggregate groups as a supplement to using management class for applications that are critical to your business. You can associate an aggregate group with a management class. The management class specifies backup attributes for the aggregate group, such as the copy technique for backing up DASD data sets on primary volumes, the number of aggregate versions to retain, and how long to retain versions. Aggregate groups simplify the control of backup and recovery of critical data sets and applications.

Although SMS must be used on the system where the backups are performed, you can recover aggregate groups to systems that are not using SMS, if the groups do not contain data that requires SMS, such as PDSEs. You can use aggregate groups to transfer applications to other data processing installations, or provide data backup and recovery for your critical data. One major advantage of using ABARS for backing up your data is the capacity to backup migrated data sets without recalling them first.

5.2.6 Automatic class selection routines

You use ACS routines to assign SMS constructs (Data Class, Storage Class, Management Class, and Storage Group) definitions to data sets, and objects. You write ACS routines using the ACS language, which is a high-level programming language. After the routines is written, you use the ACS translator to translate them to object form so they can be stored in the SMS configuration.

The ACS language contains a number of read-only variables, which you can use to analyze new data allocations. For example, you can use the read-only variable &DSN to make class and group assignments based on data set or object collection name, or &SIZE to make assignments based on the amount of space requested by the data set.

Note: You cannot alter the value of read-only variables.

Use the four read/write variables to assign the class or storage group you determine for the data set or object, based on the routine you are writing. For example, use the &STORCLAS variable to assign a storage class to a data set or object. These variables are only read/write during their designated class selection, meaning you cannot attribute a value to &STORCLAS during Storage Group class selection.

For a detailed description of the ACS language and its variables, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

For each SMS configuration, you can have active as many as four routines: One each for data class, storage class, management class, and storage group. Use ISMF to create, translate, validate, and test the routines.

Processing order of ACS routines

Figure 5-7 shows the order in which ACS routines are processed. Data can become system-managed if the storage class routine assigns a storage class to the data, or if it allows a user-specified storage class to be assigned to the data. If this routine does not assign a storage class to the data, the data cannot reside on a system-managed volume.

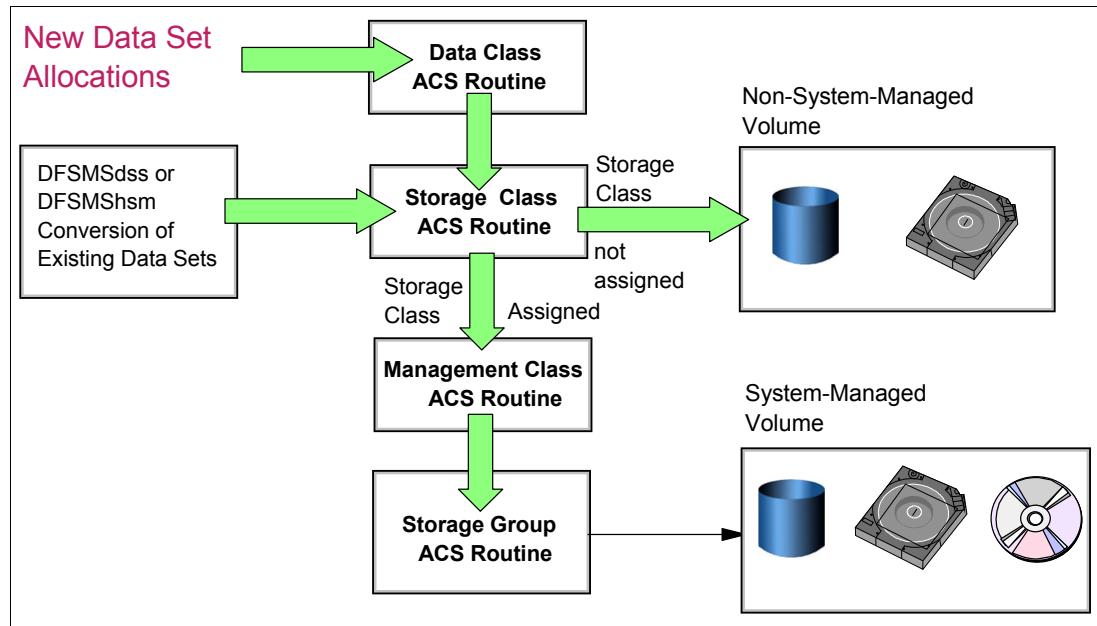


Figure 5-7 ACS routines processing

Because data allocations, whether dynamic or through JCL, are processed through ACS routines, you can enforce installation standards for data allocation on system-managed and non-system-managed volumes. ACS routines also enable you to override user specifications for data, storage, and management class, and requests for specific storage volumes.

You can use the ACS routines to determine the SMS classes for data sets created by the Distributed FileManager/MVS. If a remote user does not specify a storage class, and if the ACS routines decide that the data set is not to be system-managed, then the Distributed FileManager/MVS terminates the creation process immediately and returns an error reply message to the source. Therefore, when you construct your ACS routines, consider the potential data set creation requests of remote users.

Using ACS routines to enforce standards

The ACS routine provides an automatic method for enforcing standards during data set allocations. Standards are enforced automatically at allocation time, rather than through manual techniques after allocation.

Enforcing standards optimizes data processing resources, improves service to users, and positions you for implementing system-managed storage. You can fail requests or issue warning messages to users who do not conform to standards. Consider enforcing the following standards in your DFSMS environment:

- ▶ Prevent extended retention or expiration periods.
- ▶ Prevent specific volume allocations, unless authorized. For example, you can control allocations to spare, system, database, or other volumes.

Require valid naming conventions before implementing DFSMS system management for permanent data sets.

Activating ACS routines

Before you can implement a new routine, or any changes to your ACS routines, you need to ensure that the code is error free, and the results are as expected. Complete these steps:

1. Translate: After any changes to your ACS code, you need to translate it to check for syntax errors, and translate it to object format, so it can be stored into SCDS data set.
2. Validate: After translation, validate your ACS routines to make sure that they do not assign any non-existent construct. If there is an error in the validation, the activation step fails.
3. Test: Although not required, run a few test cases against your ACS code to make sure that the correct constructs are assigned, and no gaps exists.
4. Activate: Last, activate your SCDS configuration so your ACS routines can take effect.

For more information about ACS processing and activation, see *z/OS DFSMS: Using the Interactive Storage Management Facility*, SC26-7411.

5.3 Maintaining and monitoring SMS policies

After storage administrators have established the installation's service levels and implemented policies based on those levels, they can use DFSMS facilities to see whether the installation objectives have been met. Information on past use can help to develop more effective storage administration policies and manage growth effectively. The DFSMS Optimizer feature can be used to monitor, analyze, and tune the policies.

At the same time, changes in business requirements, law enforcements, application response time requirements, and other changes demand constant maintenance to SMS policies. This section describes some maintenance and monitoring activities.

5.3.1 Starting SMS

To start SMS, which starts the SMS address space, use either of these methods:

- ▶ With SMS=xx defined in IEASYSxx and SMS defined as a valid subsystem, IPL the system. This action starts SMS automatically.
- ▶ With SMS defined as a valid subsystem to z/OS, IPL the system. Start SMS later by using the **SET SMS=yy** MVS operator command.

For detailed information, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

5.3.2 Activating a new SMS configuration

Activating a new SMS configuration means to copy the configuration from SCDS to ACDS and to the SMS address space. The SCDS itself is never considered active. Attempting to activate an ACDS that is not valid results in an error message.

You can manually activate a new SMS configuration in two ways. SMS must be active before you use one of these methods:

1. Activating an SMS configuration from ISMF:
 - a. From the ISMF Primary Option Menu panel, select **Control Data Set**.
 - b. In the CDS Application Selection panel, enter your SCDS data set name and select **5 Activate**, or enter the ACTIVATE command on the command line.

- Activating an SMS configuration from the operator console. From the operator console, enter the following command:

```
SETSMS {ACDS(YOUR.OWN.ACDS)} {SCDS(YOUR.OWN.SCDS)}
```

Activating the configuration means that information is brought into the SMS address space from the ACDS.

To update the current ACDS with the contents of an SCDS, specify the SCDS parameter only.

If you want to both specify a new ACDS and update it with the contents of an SCDS, enter the SETSMS command with both the ACDS and SCDS parameters specified.

The ACTIVATE command, which runs from the ISMF CDS application, is equivalent to the SETSMS operator command with the SCDS keyword specified.

If you use RACF, you can enable storage administrators to activate SMS configurations from ISMF by defining the facility STGADMIN.IGD.ACTIVATE.CONFIGURATION and issuing permit commands for each storage administrator.

5.3.3 Controlling SMS processing using operator commands

The DFSMS environment provides a set of z/OS operator commands to control SMS processing. The VARY, DISPLAY, DEVSERV, and SET commands are MVS operator commands that support SMS operation.

SETSMS	This command changes a subset of SMS parameters from the operator console without changing the active IGDSMSxx PARMLIB member. For example, you can use this command to activate a new configuration from an SCDS. The MVS operator must use SETSMS to recover from ACDS and COMMDS failures. For an explanation about how to recover from ACDS and COMMDS failures, see <i>z/OS DFSMSdfp Storage Administration Reference</i> , SC26-7402.
SET SMS=zz	This command starts SMS, if it has not already been started, and is defined as a valid MVS subsystem. The command also does these things: <ul style="list-style-type: none">– Changes options set on the IGDSMSxx PARMLIB member.– Restarts SMS if it has terminated.– Updates the SMS configuration.
VARY SMS	This command changes storage group, volume, library, or drive status. You can use this command to accomplish these tasks: <ul style="list-style-type: none">– Limit new allocations to a volume or storage group.– Enable a newly-installed volume for allocations.
DISPLAY SMS	Shows volumes, storage groups, libraries, drives, SMS configuration information, SMS trace parameters, SMS operational options, OAM information, OSMC information, and cache information.
DEVSERV	This command displays information for a device. Use it to display the status of extended functions in operation for a specific volume that is attached to a cache-capable storage control.

Table 5-1 lists the differences between the SETSMS and SET SMS commands.

Table 5-1 Comparison of SETSMS and SET SMS commands

Difference	SET SMS=xx	SETSMS
When and how to use the command.	Initializes SMS parameters and starts SMS if SMS is defined but not started at IPL. Changes SMS parameters when SMS is running.	Changes SMS parameters only when SMS is running.
Where the parameters are entered.	IGDSMSxx PARMLIB member.	At the console.
What default values are available.	Default values are used for non-specified parameters.	No default values. Parameters non-specified remain unchanged.

For more information about operator commands, see *z/OS MVS System Commands*, SA22-7627.

5.4 Interactive Storage Management Facility

ISMF helps you analyze and manage data and storage interactively. ISMF is an Interactive System Productivity Facility (ISPF) application.

ISMF provides interactive access to the space management, backup, and recovery services of the DFSMShsm and DFSMSdss functional components of DFSMS, to the tape management services of the DFSMSrmm functional component, as well as to other products. DFSMS introduces the ability to use ISMF to define attributes of tape storage groups and libraries.

A storage administrator uses ISMF to define the installation's policy for managing storage by defining and managing SMS classes, groups, and ACS routines. ISMF then places the configuration in an SCDS. You can activate an SCDS through ISMF or an operator command.

ISMF is menu-driven, with fast paths for many of its functions. ISMF uses the ISPF Dialog Tag Language (DTL) to give its functional panels on workstations the look of Common User Access (CUA) panels and a graphical user interface (GUI).

5.4.1 ISMF: What you can do with ISMF

ISMF is a panel-driven interface. Use the panels in an ISMF application to accomplish these tasks:

- ▶ Display lists with information about specific data sets, DASD volumes, mountable optical volumes, and mountable tape volumes
- ▶ Generate lists of data, storage, and management classes to determine how data sets are being managed
- ▶ Display and manage lists saved from various ISMF applications

ISMF generates a data list based on your selection criteria. After the list is built, you can use ISMF entry panels to perform space management or backup and recovery tasks against the entries in the list.

As a user performing data management tasks against individual data sets or against lists of data sets or volumes, you can use ISMF to accomplish these tasks:

- ▶ Edit, browse, and sort data set records
- ▶ Delete data sets and backup copies
- ▶ Protect data sets by limiting their access
- ▶ Recover unused space from data sets and consolidate free space on DASD volumes
- ▶ Copy data sets or DASD volumes to the same device or another device
- ▶ Migrate data sets to another migration level
- ▶ Recall data sets that have been migrated so that they can be used
- ▶ Back up data sets and copy entire volumes for availability purposes
- ▶ Recover data sets and restore DASD volumes, mountable optical volumes, or mountable tape volumes

You cannot allocate data sets from ISMF. Data sets are allocated from ISPF, from TSO, or with JCL statements. ISMF provides the DSUTIL command, which enables users to get to ISPF and toggle back to ISMF.

5.4.2 Accessing ISMF

How you access ISMF depends on your site:

- ▶ You can create an option on the ISPF Primary Option Menu to access ISMF. Then, access ISMF by typing the appropriate option after the arrow on the Option field in the ISPF Primary Option Menu. This action starts an ISMF session from the ISPF/PDF Primary Option Menu.
- ▶ To access ISMF directly from TSO, use the command:

```
ISPSTART PGM(DGTFMD01) NEWAPPL(DGT)
```

There are two Primary Option Menus: One for storage administrators, and another for users. The menu available to storage administrators includes additional applications not available to users. Option 0 controls the user mode or the type of Primary Option Menu to be displayed.

The ISMF Primary Option Menu example assumes installation of DFSMS at the current release level. For information about adding the DFSORT option to your Primary Option Menu, see *DFSORT Installation and Customization Release 14*, SC33-4034.



Catalogs

Knowing where your data is stored is vital to perform any data-related tasks, from a simple open for I/O, to providing backups for availability management. Whether you are a small shop with a single LPAR and a few data sets, or a large system with dozens LPARs and millions of data sets to manage, catalogs provide you a simple, easy, and efficient way to organize information about your data sets.

In z/OS, catalogs are controlled by an embedded DFSMSdfp component that is called Catalog Management. Catalog Management has one address space for itself named catalog address space (CAS). This control allows a good catalog performance, while managing multiple requests to catalogs, and ensure catalog integrity. This chapter introduces you to z/OS catalogs structures, usage, and considerations.

This chapter includes the following sections:

- ▶ The integrated catalog facility structure
- ▶ Aliases
- ▶ Catalog search order
- ▶ Using multiple catalogs
- ▶ Sharing catalogs across systems
- ▶ Catalog caching
- ▶ Catalog address space
- ▶ Maintaining your catalogs

6.1 The integrated catalog facility structure

Catalogs are data sets containing information about other data sets. They provide users with the ability to locate a data set by name, without knowing the volume where the data set resides. This feature means that data sets can be moved from one device to another, without requiring a change in job control language (JCL) DD statements that refer to an existing data set.

Cataloging data sets also simplifies backup and recovery procedures. Catalogs are the central information point for VSAM data sets, so all VSAM data sets must be cataloged. In addition, all SMS-managed data sets must be cataloged. Activity towards the catalog is much more intense in a batch/TSO workload than in a CICS/Db2 workload, where most data sets are allocated at CICS/Db2 initialization time.

An integrated catalog facility (ICF) catalog is a structure that replaced the former MVS CVOL catalog. As a catalog, it describes data set attributes and indicates the volumes on which a data set is located. ICF catalogs are allocated by the CAS, a system address space for the DFSMSdfp catalog function.

A catalog consists of two separate kinds of data sets:

- ▶ A basic catalog structure (BCS). The BCS can be considered the catalog.
- ▶ A VSAM volume data set (VVDS). The VVDS can be considered an extension of the volume table of contents (VTOC).

Figure 6-1 shows how BCS and VVDS work together to create the ICF catalogs structure.

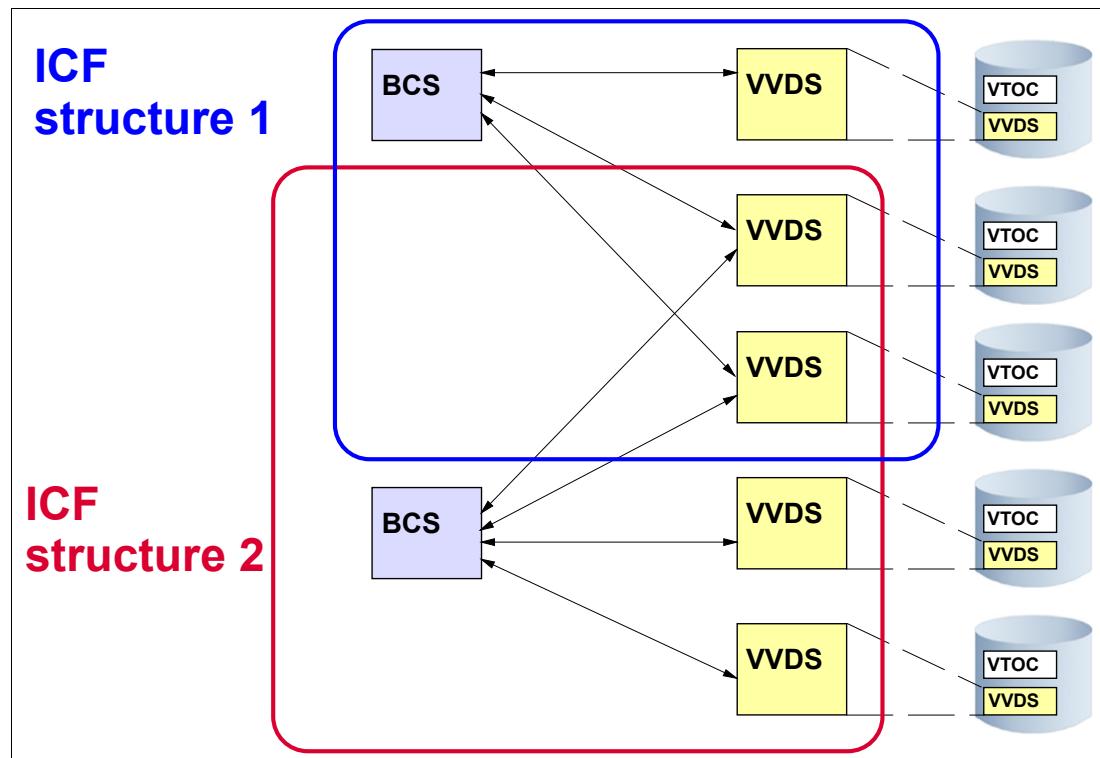


Figure 6-1 The ICF catalog structure

6.1.1 Basic catalog structure

The BCS is a VSAM key-sequenced data set that contains the information about where a data set resides. That can be a DASD volume, tape, or other storage medium. Related information in the BCS is grouped into logical, variable-length, spanned records related by key. The BCS uses keys that are the *data set names* (plus one character for extensions).

One control interval can contain multiple BCS records. To reduce the number of I/Os necessary for catalog processing, logically-related data is consolidated in the BCS.

A catalog can have data sets cataloged on any number of volumes. The BCS can have as many as 123 extents on one volume. One volume can have multiple catalogs on it. All the necessary control information is recorded in the VVDS on that volume.

The catalogs (BCSs) can be classified as *master* catalogs and *user* catalogs. A particular case of a user catalog is the *volume* catalog, which is a user catalog that contains only the tape library and tape volume entries.

There is no structural difference between a master catalog and a user catalog. What sets a master catalog apart is how it is used, and what data sets are cataloged in it. For example, the same catalog can be master in one z/OS and user in the other z/OS. They are described in more detail in the following sections.

The master catalog

Each system has one active master catalog. One master catalog can be shared between various MVS images. It does not have to be on the system residence volume (the one that is restarted).

The master catalog for a system must contain entries for all user catalogs and their aliases that the system uses. Also, all SYS1 data sets must be cataloged in the master catalog for proper system initialization.

Important: To minimize update activity to the master catalog, and to reduce the exposure to breakage, plan to place *only* SYS1 data sets, user catalog connector records, and the aliases pointing to those connectors in the master catalog.

During a system initialization, the master catalog is read so that system data sets and catalogs can be located. You can identify the master catalog in a system by performing one of the following tasks:

- ▶ Check SYS1.NUCLEUS member SYSCATxx (default is SYSCATLG).
- ▶ Check SYS1.PARMLIB/SYSn.IPLPARM member LOADxx.
- ▶ Issue a LISTCAT command against a SYS1 data set, and verify the catalog name.

User catalogs

The difference between the master catalog and the user catalogs is in the *function*. User catalogs are to be used to contain information about your installation cataloged data sets other than SYS1 data sets. There are no set rules as to how many or what size to have. The configuration depends entirely on your environment.

Cataloging data sets for two unrelated applications in the same catalog might increase the impact to your business during a failure. Assessing the impact of outage of a specific catalog can help to determine whether it is too large or can affect too many applications.

Volume catalog

A volume catalog (VOLCAT) is a catalog that contains only tape library and tape volume entries. A general VOLCAT contains all tape library entries and any tape volume entries that do not point to a specific VOLCAT. A specific VOLCAT cannot contain tape library entries. It contains a specific group of tape volume entries based on the tape volume serial numbers (tape volters).

6.1.2 VSAM volume data set

The VVDS is a data set that describes the characteristics of VSAM and system-managed data sets on a DASD volume. It is part of an ICF catalog structure.

The VVDS contains VSAM volume records (VVRs) that hold information about VSAM data sets on the volume. The VVDS also contains non-VSAM volume records (NVRs) for SMS-managed non-VSAM data sets on the volume. If an SMS-managed non-VSAM data set spans volumes, then only the first volume contains an NVR for that data set.

If not previously defined, the system automatically defines a VVDS to your volume when the first VSAM or SMS-managed data set is allocated into the device. If not changed by the user, the default size for VVDS data sets is 10 tracks primary and 10 tracks secondary space.

There are three types of entries in a VVDS:

- ▶ VSAM volume control records (VPCR)
 - First logical record in a VVDS
 - Contain information for management of DASD space and the names of the BCSs that have data sets on the volume
- ▶ VSAM volume records (VVR)
 - Contain information about a VSAM data set on the volume
 - Number of VVRs varies according to the type of data set and the options specified for the data set
 - Also includes data set characteristics, SMS data, and extent information
 - There is one VVR describing the VVDS itself
- ▶ Non-VSAM volume record (NVR)
 - Equivalent to a VVR for SMS-managed non-VSAM data sets
 - Contains SMS-related information

VVDS characteristics

The VVDS is a VSAM entry-sequenced data set (ESDS) that has a 4 KB control interval size. The hexadecimal RBA of a record is used as its key or identifier.

A VVDS is recognized by the restricted data set name:

SYS1.VVDS.Vvolser

Volser is the volume serial number of the volume on which the VVDS resides.

You can *explicitly* define the VVDS using IDCAMS, or it is *implicitly* created after you define the first VSAM or SMS-managed data set on the volume. Generally, plan allocating your VVDS before any data sets are allocated to reduce volume fragmentation.

An explicitly defined VVDS is not related to any BCS until a data set or catalog object is defined on the volume. Because data sets are allocated on the VVDS volume, each BCS with VSAM data sets or SMS-managed data sets on that volume is related to the VVDS.

Although VVDS data sets start with SYS1 high-level qualifier, they are not necessarily cataloged to the system master catalog. Instead, each catalog having at least one VSAM or SMS-managed data set in a volume will also have a pointer to the volume VVDS. Plan to remove these VVDS entries when the volumes are formatted with a new name.

6.2 Aliases

Aliases are used to tell catalog management which user catalog your data set is cataloged in. First, you place a pointer to a user catalog in the master catalog through the IDCAMS DEFINE UCAT command. Next, you define an appropriate alias name for a user catalog in the master catalog. Then, match the high-level qualifier (HLQ) of your data set with the alias. This qualifier identifies the appropriate user catalog to be used to satisfy the request.

In Figure 6-2, all data sets with an HLQ of PAY have their information in the user catalog UCAT1 because in the master catalog there is an alias PAY that points to UCAT1.

The data sets with an HLQ of DEPT1 and DEPT2 have their information in the user catalog UCAT2 because in the master catalog, there are aliases DEPT1 and DEPT2 pointing to UCAT2.

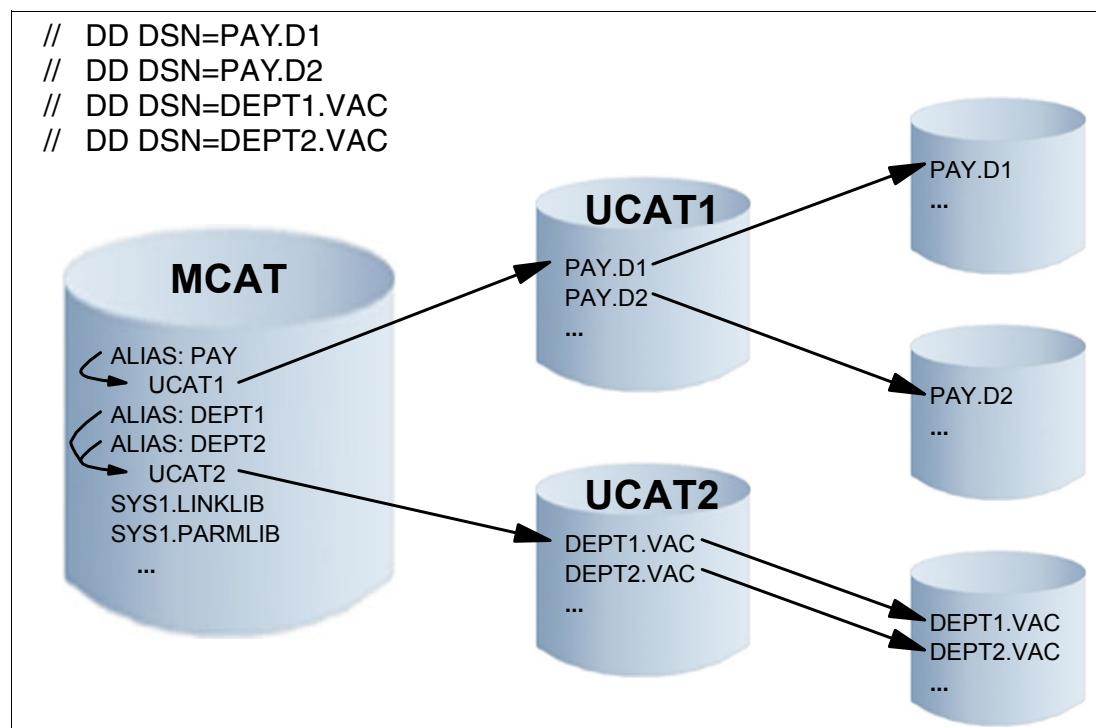


Figure 6-2 Sample alias processing

Note: Aliases can also be used with non-VSAM data sets to create alternate names to the same data set. Those aliases are *not* related to a user catalog.

6.2.1 Multilevel aliases

You can augment the standard catalog search order by defining multilevel catalog aliases. A multilevel catalog alias is an alias of two or more high-level qualifiers. You can define aliases of up to four high-level qualifiers.

However, the multilevel alias facility is only to be used when a better solution cannot be found. The need for the multilevel alias facility can indicate data set naming conventions problems.

For more information about the multilevel alias facility, see *z/OS DFSMS: Managing Catalogs*, SC26-7409.

6.3 Catalog search order

Whenever you reference a data set and do not provide the data set volume name, the system performs a catalog search to locate your data set within the system devices. Performing a catalog search to locate and access a data set is the most common way to reference data in z/OS because manually specifying the volume name for each request is impractical today.

When a search for a data set is performed, a LOCATE SVC calls the catalog management asking for a data set name search. Most catalog searches are based on catalog aliases.

These searches are performed when you are either trying to allocate a new data set, or access an existing data set for read, write, update, or delete. You can also explicitly code the catalog name to force the search against a specific catalog using the IDCAMS CATALOG keyword. Figure 6-3 is a diagram with standard search order for a LOCATE request.

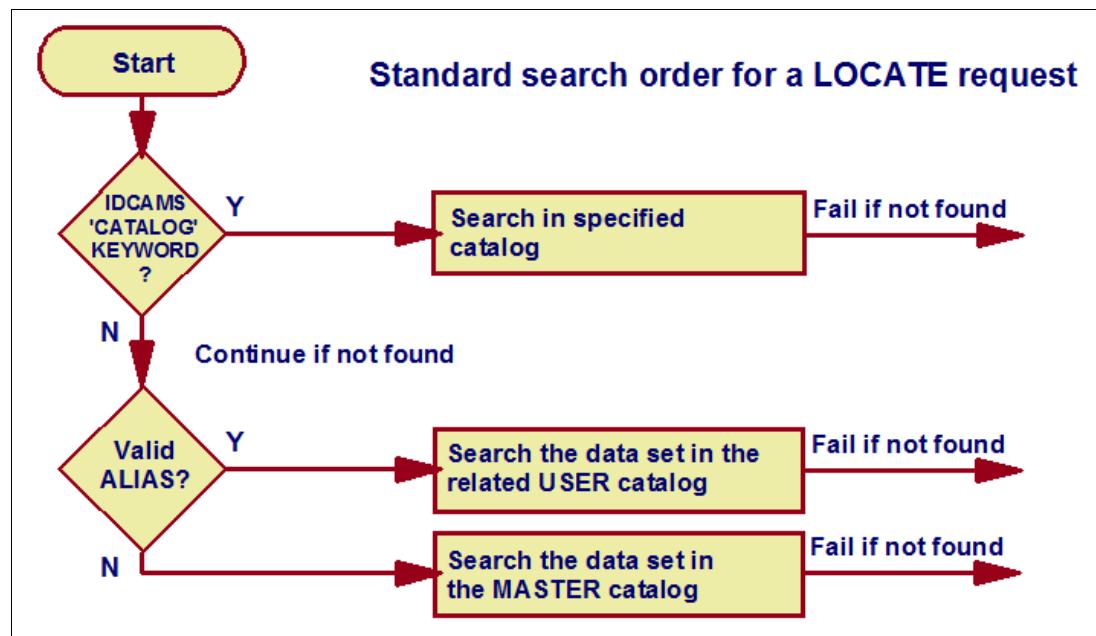


Figure 6-3 Catalog search order for a LOCATE request

6.3.1 Search order for catalogs for a data set define request

For the system to determine where a data set is to be cataloged, the following search order is used to find the catalog:

1. Use the catalog named in the IDCAMS CATALOG parameter, if coded.
2. If the high-level qualifier is a catalog alias, use the catalog identified by the alias or the catalog whose name is the same as the high-level qualifier of the data set.
3. If no catalog has been identified yet, use the master catalog.

6.3.2 Search order for locating a data set

Base catalog searches on the catalog aliases. When appropriate aliases are defined for catalogs, the high-level qualifier of a data set name is identical to a catalog alias and identifies the appropriate catalog to be used to satisfy the request.

However, alternatives to catalog aliases are available for directing a catalog request, specifically the CATALOG parameter of access method services and the name of the catalog.

The following search order is used to locate the catalog for an already cataloged data set:

1. Use the catalog named in IDCAMS CATALOG parameter, if coded. If the data set is not found, fail the job.
2. If the CATALOG statement is not found, and the high-level qualifier is an alias for a catalog, search the catalog. If the data set is not found, fail the job.
3. Otherwise, search the master catalog.

To use an alias to identify the catalog to be searched, the data set must have more than one data set qualifier. For information about the catalog standard search order, see *z/OS DFSMS: Managing Catalogs*, SC26-7409.

6.4 Using multiple catalogs

Multiple catalogs on multiple volumes can perform better than fewer catalogs on fewer volumes. This difference is because of the interference between requests to the same catalog, such as a single shared catalog being locked out by another system in the sysplex. This situation can occur if another application issues a RESERVE against the volume that has nothing to do with catalog processing. Another reason can be that there is more competition to use the available volumes, and thus more I/O can be in progress concurrently.

Tip: Convert all intra-sysplex RESERVES in global ENQs through the conversion RNL.

Multiple catalogs can reduce the impact of the loss of a catalog for these reasons:

- ▶ Reducing the time necessary to re-create any catalog
- ▶ Allowing multiple catalog recovery jobs to be in process at the same time

Recovery from a pack failure depends on the total amount of catalog information about a volume, regardless of whether this information is stored in one catalog or in many catalogs.

When using multiple user catalogs, consider grouping data sets under different high-level qualifiers. Independent of the number of catalogs, use the available caching options to increase your catalog search performance. The caching options are discussed in more detail in 6.6, “Catalog caching” on page 111.

Based on your current system usage, you might find that some catalogs are growing at a large pace, while other catalogs are underutilized. You can analyze your catalog utilization and split or merge catalogs to achieve your performance and maintenance requirements.

6.4.1 Splitting catalogs

You can split a catalog to create two catalogs or to move a group of catalog entries if you determine that a catalog is either unacceptably large or that it contains too many entries for critical data sets.

If the catalog is unacceptably large (that is, a catalog failure would leave too many entries inaccessible), then you can split the catalog into two catalogs. If the catalog is of an acceptable size but contains entries for too many critical data sets, then you can simply move entries from one catalog to another. For more information about splitting catalogs, including step-by-step instructions for splitting catalogs, see *z/OS DFSMS: Managing Catalogs*, SC26-7409.

6.4.2 Merging catalogs

You might find it beneficial to merge catalogs if you have many small or seldom-used catalogs. An excessive number of catalogs can complicate recovery procedures and waste resources such as CAS storage, tape mounts for backups, and system time performing backups.

Merging catalogs is accomplished in much the same way as splitting catalogs. The only difference between splitting catalogs and merging them is that in merging, you want all the entries in a catalog to be moved to another catalog, so that you can delete the obsolete catalog. For more information about merging catalogs, including step-by-step instructions for merging catalogs, see *z/OS DFSMS: Managing Catalogs*, SC26-7409.

6.5 Sharing catalogs across systems

A shared catalog is a BCS that is eligible to be used by more than one system. It must be defined with SHAREOPTIONS(3 4), and be on a shared volume. A DASD volume is initialized as shared using the MVS hardware configuration definition (HCD) facility.

Note: The device must be defined as shared to all systems that access it.

If several systems have the device defined as shared and other systems do not, then catalog corruption will occur. Check with your system programmer to determine shared volumes. Tape volume catalogs can be shared in the same way as other catalogs.

By default, catalogs are defined with SHAREOPTIONS(3 4). You can specify that a catalog is not to be shared by defining the catalog with SHAREOPTIONS(3 3). Only define a catalog as unshared if you are certain it will not be shared. Preferably place unshared catalogs on volumes that have been initialized as unshared. Catalogs that are defined as unshared and that reside on shared volumes will become damaged if referred to by another system.

If you need to share data sets across systems, it is advisable that you share the catalogs that contain these data sets. A BCS catalog is considered shared when *both* of the following are true:

- ▶ It is defined with SHAREOPTIONS (3 4).
- ▶ It resides on a shared device, as defined at HCD.

Attention: To avoid catalog corruption, define a catalog volume on a shared UCB and set catalog SHAREOPTIONS to (3 4) on *all* systems that share a catalog.

Note that CAS considers a catalog shared when both values mentioned are true, regardless if the catalog is actually accessed by other systems or not.

Using SHAREOPTIONS 3 means that VSAM does not issue the ENQ SYSVSAM SYSTEMS for the catalog. SHAREOPTIONS 4 means that the VSAM buffers need to be refreshed.

You can check whether a catalog is shared by running the operator command:

```
MODIFY CATALOG,ALLOCATED
```

A flag in the catalog indicates whether the catalog is shared.

The VVR entry for a shared catalog is used as a log by all the catalog management that accesses such a catalog. This log is used to help ensure the coherency of each catalog buffer in each z/OS system.

Note: Checking VVR entry for a catalog can increase the I/O ratio for the volume and impact catalog performance. Plan on using the caching options to prevent I/Os and check catalog coherency. For more information about caching, see 6.6, "Catalog caching" on page 111

6.6 Catalog caching

Most data set allocations by jobs, users, and applications are performed by catalog searches. This fact causes catalogs to be highly accessed, and any performance issues with a specific catalog can affect all applications that access data sets within that catalog. For that reason, careful planning must be done to ensure good catalog performance.

Caching catalogs can assist you in providing good catalog response times, and even keep the CAS CPU utilization down. There are several caching options that can be used for catalogs. These cache options are explained in more detail in the following sections.

6.6.1 In-storage cache

In-storage cache (ISC) is the virtual memory in CAS reserved for caching. The amount of memory that is used is fixed. When the cache fills, entries are removed from the cache based on a least recently used (LRU) basis. This caching is used (performance-wise) by MCAT control intervals.

This caching option does not deal with data buffer modifications. If these modifications occur, the catalog management flushes the cache in memory, and performs I/Os to retrieve the new information from catalog. For that reason, use ISC only when other caching options are not available.

6.6.2 Catalog data space caching

Catalog data space caching (CDSC), or virtual lookaside facility (VLF) caching, is based in a data space that is owned by the VLF, which is a z/OS component. It is defined through the COFVLFx member in SYS1.PARMLIB. Catalog entries are not limited by a piece of the data space storage size at a catalog level, but use whatever storage is assigned to the data space in total. At the point when the storage becomes saturated, an LRU algorithm starts removing entries from the VLF cache.

6.6.3 Enhanced Catalog Sharing

Enhanced Catalog Sharing (ECS) differs from other cachings by not caching catalog entries. Instead, it uses Coupling Facility (CF) structure resources to keep the VVR for the catalog on memory. By doing that, CAS does not need to perform an I/O operation to check VVR, accessing the CF structure for coherency checking. ECS can be used along with ISC and VLF caching to improve performance.

6.6.4 VSAM record-level sharing

VSAM record-level sharing (RLS) allows data sharing at record level. This feature reduces the chances of delays related to one CAS trying to access records locked by other CAS in another logical partition (LPAR). It also uses the SMSVSAM data space to store records, and removes the need to check VVR records for catalog updates. The master catalog cannot use RLS.

To learn more about catalog caching and implementation steps, check *z/OS DFSMS: Managing Catalogs*, SC26-7409.

6.7 Catalog address space

Catalog functions are performed in the CAS. The jobname of the catalog address space is CATALOG.

As soon as a user requests a catalog function (for example, to locate or define a data set), the CAS gets control to handle the request. When it has finished, it returns the requested data to the user. A catalog task that handles a single user request is called a *service task*. A service task is assigned to each user request. The minimum number of available service tasks is specified in the SYSCATxx member of SYS1.NUCLEUS (or the LOADxx member of SYS1.PARMLIB). A table called the CRT tracks these service tasks.

The CAS contains all information necessary to handle a catalog request, like control block information about all open catalogs, alias tables, and buffered BCS records.

During the initialization of an MVS system, all user catalog names identified in the master catalog, their aliases, and their associated volume serial numbers are placed in tables in CAS.

You can use the MODIFY CATALOG operator command to work with the catalog address space. For more information, see 6.8.6, “Working with the catalog address space” on page 118.

6.8 Maintaining your catalogs

After your catalogs are defined, you need to perform periodic tasks to ensure that the catalogs are being properly used, response times are good, and the catalogs are recoverable during a failure.

This section describes some activities that can assist you in maintaining your catalog environment. For a deeper view on catalogs maintenance and activities, see *A Practical Guide to ICF Catalogs*, SG24-8262 and *z/OS DFSMS: Managing Catalogs*, SC26-7409.

6.8.1 Listing a catalog

You can list catalog records by using the IDCAMS LISTCAT command, or the ISMF line operator command CATLIST. CATLIST produces the same output as LISTCAT.

You can use the LISTCAT output to monitor VSAM data sets including catalogs. The statistics and attributes listed can be used to help determine whether you reorganize, re-create, or otherwise alter a VSAM data set, including catalogs, to improve performance or avoid problems.

The LISTCAT command can be used in many variations to extract information about a particular entry in the catalog. It extracts the data from the BCS and VVDS.

The following are some examples of using LISTCAT for monitoring catalogs:

- ▶ List all ALIAS entries in the master catalog:

```
LISTCAT ALIAS CAT(master.catalog.name)
```

This command provides a list of all aliases that are currently defined in your master catalog. If you need information only about one specific alias, use the keyword **ENTRY(aliasname)** and specify ALL to get detailed information.

- ▶ List a user catalog connector in the master catalog:

```
LISTCAT ENT(user.catalog.name) ALL
```

You can use this command to display the volser and the alias associations of a user catalog as it is defined in the master catalog.

- ▶ List the catalog's self-describing record:

```
LISTCAT ENT(user.catalog.name) CAT(user.catalog.name) ALL
```

This command provides detailed information about a user catalog, such as attributes, statistics, extent information, and so on. Because the self-describing record is in the user catalog, you must specify the name of the user catalog in the CAT statement. If you do not use the CAT keyword, only the user catalog connector information from the master catalog is listed as in the previous example.

- ▶ Listing a VSAM or non-VSAM data set:

```
LISTCAT ENT(data.set.name) ALL
```

The output for a VSAM data set looks the same as the catalog's LISTCAT output. For a non-VSAM data set, the output is much shorter.

You can use the LISTCAT command to list information for other catalog entries as well. For information about LISTCAT, see *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

6.8.2 Backup procedures

The two parts of an ICF catalog, the BCS and the VVDS, require separate backup techniques. The BCS can be backed up like any other data set. Only back up the VVDS as part of a volume dump. The entries in the VVDS and VTOC are backed up when the data sets they describe are acted upon in these ways:

- ▶ Exported with IDCAMS
- ▶ Logically dumped with DFSMSdss
- ▶ Backed up with DFSMShsm

Important: Because catalogs are essential system data sets, it is important that you maintain backup copies. The more recent and accurate a backup copy, the less effect a catalog outage will have on your installation.

To back up a BCS, use one of the following methods:

- ▶ The access method services EXPORT command
- ▶ The DFSMSdss logical DUMP command
- ▶ The DFSMShsm BACKDS command

You can later recover the backup copies using the same utility used to create the backup:

- ▶ The access method services IMPORT command for exported copies
- ▶ The DFSMSdss RESTORE command for logical dump copies
- ▶ The DFSMShsm RECOVER command for DFSMShsm backups

The copy created by these utilities is a *portable* sequential data set that can be stored on a tape or direct access device, which can be of another device type than the one containing the source catalog.

When these commands are used to back up a BCS, the aliases of the catalog are saved in the backup copy. The source catalog is not deleted, and remains as a fully functional catalog. The relationships between the BCS and VVDSs are unchanged.

You cannot permanently export a catalog by using the PERMANENT parameter of EXPORT. The TEMPORARY option is used even if you specify PERMANENT or allow it to default. Figure 6-4 shows you an example for an IDCAMS EXPORT.

```
//EXPTCAT JOB ...
//STEP1 EXEC PGM=IDCAMS
//RECEIVE DD DSNAME=CATBACK,UNIT=(TAPE,,DEFER),
//          DISP=(NEW,KEEP),VOL=SER=327409,LABEL=(1,SL)
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
      EXPORT -
          USER.CATALOG      -
          OUTFILE(RECEIVE) -
      TEMPORARY
/*
```

Figure 6-4 JCL to create a backup of a BCS using IDCAMS EXPORT

Note: You cannot use IDCAMS REPRO or other copying commands to create and recover BCS backups.

You can also make periodic volume dumps of the master catalog's volume. This dump can later be used by the stand-alone version of DFSMSdss to restore the master catalog if you cannot access the volume from another system.

For information about defining and using catalog back-up options, see *z/OS DFSMS: Managing Catalogs*, SC26-7409.

Backing up a VVDS

Do not back up the VVDS as a data set to provide for recovery. To back up the VVDS, back up the volume containing the VVDS, or back up all data sets described in the VVDS (all VSAM and SMS-managed data sets). If the VVDS ever needs to be recovered, recover the entire volume, or all the data sets described in the VVDS.

You can use either DFSMSdss or DFSMShsm to back up and recover a volume or individual data sets on the volume.

6.8.3 Recovery procedures

Normally, a BCS is recovered separately from a VVDS. A VVDS usually does not need to be recovered, even if an associated BCS is recovered. However, if you need to recover a VVDS, and a BCS is on the VVDS's volume, you must recover the BCS as well. If possible, export the BCS before recovering the volume, and then recover the BCS from the exported copy. This process ensures a current BCS.

Before recovering a BCS or VVDS, try to recover single damaged records. If damaged records can be rebuilt, you can avoid a full recovery.

The way that you recover a BCS depends on how it was saved (see 6.8.2, "Backup procedures" on page 114). When you recover a BCS, you do not need to delete and redefine the target catalog unless you want to change the catalog's size or other characteristics, or unless the BCS is damaged in such a way as to prevent the usual recovery.

Aliases to the catalog can be defined if you use DFSMSdss, DFSMShsm, or if you specify ALIAS on the IMPORT command. If you have not deleted and redefined the catalog, all existing aliases are maintained, and any aliases defined in the backup copy are redefined if they are not already defined.

Lock the BCS before you start recovery so that no one else has access to it while you recover the BCS. If you do not restrict access to the catalog, users might be able to update the catalog during recovery or maintenance and create a data integrity exposure. The catalog also will be unavailable to any system that shares the catalog. You cannot lock a master catalog.

After you recover the catalog, update the BCS with any changes that have occurred since the last backup. You can use the access method services DIAGNOSE command to identify certain unsynchronized entries.

The Integrated Catalog Forward Recovery Utility

You also can use the Integrated Catalog Forward Recovery Utility (ICFRU) to recover a damaged catalog to a correct and current status. This utility uses SMF records that record changes to the catalog, updating the catalog with changes made since the BCS was backed up. The SMF records are used by ICFRU as a log database. Use ICFRU to avoid the loss of catalog data even after recovery.

For further information about recovery procedures, see *z/OS DFSMS: Managing Catalogs*, SC26-7409. For information about the IDCAMS facility, see *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

6.8.4 Checking the integrity on an ICF structure

An error in the ICF structure can be caused either by a structural integrity error or a data structure error. A structural integrity error usually means that the data set is broken (logically or physically). The data set no longer has a valid structure that the VSAM component can handle.

Alternatively, an error within the data structure of a BCS or VVDS means that the VSAM structure of the BCS or VVDS is still valid. VSAM has no problems accessing the data set. However, the content of each of the single records in the BCS or VVDS does not conform to catalog standards. The information in the BCS and VVDS for a single data set can be unsynchronized, making the data set inaccessible.

Two kinds of VSAM errors can occur with your BCS or VVDS:

- ▶ Logical errors

The records on the DASD volume still have valid physical characteristics like record size or CI size. The VSAM information in those records is wrong, like pointers from one record to another or the end-of-file information.

- ▶ Physical errors

The records on the DASD volume are invalid. For example, they might be a wrong length. Reasons can be an overlay of physical DASD space or wrong extent information for the data set in the VTOC or VVDS.

When errors in the VSAM structure occur, they are in most cases logical errors for the BCS. Because the VVDS is an ESDS, it has no index component. Logical errors of an ESDS are unlikely.

You can use the IDCAMS EXAMINE command to analyze the structure of the BCS. As explained previously, the BCS is a VSAM key-sequenced data set (KSDS). Before running the EXAMINE, run an IDCAMS VERIFY to make sure that the VSAM information is current, and ALTER LOCK the catalog to prevent update by others while you are inspecting it.

With the parameter INDEXTEST, you analyze the integrity of the index. With parameter DATATEST, you analyze the data component. After you have the results of INDEXTEST and DATATEST, you can take the necessary steps to successfully recover your catalog. For more information about the steps to fix a catalog error, see *z/OS DFSMS: Managing Catalogs*, SC26-7409.

Additionally, you can use the IDCAMS DIAGNOSE command to validate the contents of a BCS or VVDS. You can use this command to check a single BCS or VVDS and to compare the information between a BCS and multiple VVDSs.

For various DIAGNOSE examples, see *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

6.8.5 Catalog performance

The main factors affecting catalog performance are the amount of I/O required for the catalog and the subsequent amount of time it takes to perform the I/O. These factors can be reduced by buffering catalogs in special buffer pools, or applying RLS to your catalogs. Other factors are the size and usage of the master catalog, options that were used to define a catalog, and the way you share catalogs between systems.

The simplest method of improving catalog performance is to use caching to maintain catalog records in memory and prevent physical I/O. Three types of buffer are available for catalogs:

- ▶ The ISC buffer is contained within the CAS.
- ▶ The CDSC is separate from CAS and uses the z/OS VLF component, which stores the buffered records in a data space.
- ▶ The RLS uses the SMSVSAM address space and coupling facility structures to control and share data between LPARs.

All types of buffer are optional, and each can be canceled and restarted without an IPL.

The three types of buffer are used to keep catalog records in the storage. This technique avoids I/Os that are necessary to read the records from DASD. There are several things that you need to take into considerations to decide what kind of buffer to use for which catalog. See *z/OS DFSMS: Managing Catalogs*, SC26-7409, for more information about buffering.

Another kind of caching is using enhanced catalog sharing to avoid I/Os to read the catalog VVR. Refer to 6.6.3, “Enhanced Catalog Sharing” on page 112 for more information about this topic.

Master catalog

If the master catalog only contains entries for catalogs, catalog aliases, and system data sets, the *entire* master catalog is read into main storage during system initialization. Because the master catalog, if properly used, is rarely updated, the performance of the master catalog is not appreciably affected by I/O requirements. For that reason, keep the master catalog small and do not define user data sets into it.

For more information about these values, see *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

Convert SYSIGGV2 resource

If the catalog does not use RLS, catalog management uses the SYSIGGV2 reserve when serializing access to catalogs. The SYSIGGV2 reserve is used to serialize the entire catalog BCS component across all I/O and to serialize access to specific catalog entries.

If the catalog is shared only within one GRSplex, convert the SYSIGGV2 resource to a global enqueue to avoid reserves on the volume on which the catalog resides. If you are not converting SYSIGGV2, you can have ENQ contentions on those volumes and even run into deadlock situations.

Important: If you share a catalog with a system that is not in the same GRS complex, do *not* convert the SYSIGGV2 resource for this catalog. Sharing a catalog outside the complex requires reserves for the volume on which the catalog resides. Otherwise, you will break the catalog. For more information, see *z/OS MVS Planning: Global Resource Serialization*, SA22-7600.

6.8.6 Working with the catalog address space

You can use the command MODIFY CATALOG to extract information from the CAS and to interact with the CAS. This command can be used in many variations. This section provides an overview of parameters to be aware of when maintaining your catalog environment.

For a discussion about the entire functionality of the MODIFY CATALOG command, see *z/OS DFSMS: Managing Catalogs*, SC26-7409.

Examples of the MODIFY CATALOG command:

▶ **MODIFY CATALOG,REPORT**

The catalog report lists information about the CAS, like the service level, the catalog address space ID, service tasks limits, and so on. Since z/OS 1.7, this command generates the IEC392I that identifies the top three holders of the CATALOG service tasks, possibly indicating a lockup.

▶ **MODIFY CATALOG,OPEN**

This command lists all catalogs that are currently open in the CAS. It shows whether the catalog is locked or shared, and the type of buffer used for the catalog. A catalog is opened after an IPL or catalog restart when it is referenced for the first time. It remains open until it is manually closed by the MODIFY CATALOG command or it is closed because the maximum number of open catalogs has been reached.

▶ **MODIFY CATALOG,LIST**

The LIST command shows you all service tasks that are currently active handling a user request. Normally the active phase of a service task is only of a short duration, so no tasks are listed. This command helps you to identify tasks that might be the reason for catalog problems if performance slows down.

▶ **MODIFY CATALOG,REPORT,PERFORMANCE**

The output of this command shows you significant catalog performance numbers about number and duration of ENQs and DEQs for the BCS and VVDS and more. Use the command to identify performance problems.

▶ **MODIFY CATALOG,REPORT,CACHE**

The cache report lists cache type and statistics for the single catalogs that are open in the CAS.

Restarting the catalog address space

Consider restarting the CAS only as a final option before restarting a system. Never try restarting CAS unless an IPL is your only other option. A system failure that is caused by catalogs, or a CAS storage shortage due to FREEMAIN failures, might require you to use MODIFY CATALOG,RESTART to restart CAS in a new address space.

Never use RESTART to refresh catalog or VVDS control blocks, or to change catalog characteristics. Restarting CAS is a drastic procedure, and if CAS cannot restart, you must IPL the system.

When you issue MODIFY CATALOG,RESTART, the CAS mother task is abended with abend code 81A, and any catalog requests that in process at that time are redriven.

The restart of CAS in a new address space is transparent to all users. However, even when all requests are redriven successfully and receive a return code of zero (0), the system might produce indicative dumps. There is no way to suppress these indicative dumps.



DFSMS Transactional VSAM Services

DFSMS Transactional Virtual Storage Access Method (VSAM) Services (DFSMStvs) is an enhancement to VSAM record-level sharing (RLS) access that enables multiple batch update jobs and customer information control system (CICS) to share access to the same data sets. DFSMStvs provides two-phase commit and backout protocols, as well as backout logging and forward recovery logging. DFSMStvs provides transactional recovery directly within VSAM.

As an extension of VSAM RLS, DFSMStvs enables any job or application that is designed for data sharing to read-share or write-share VSAM recoverable data sets. VSAM RLS provides a server for sharing VSAM data sets in a sysplex. VSAM RLS uses Coupling Facility-based locking and data caching to provide sysplex-scope locking and data access integrity. DFSMStvs adds logging, commit, and backout processing.

To understand DFSMStvs, it is necessary to first review base VSAM information and VSAM RLS.

In this chapter we cover the following topics:

- ▶ Introduction to VSAM RLS
- ▶ Introduction to DFSMStvs

This chapter includes the following sections:

- ▶ VSAM record-level sharing introduction
- ▶ VSAM RLS locking
- ▶ Buffering under VSAM RLS
- ▶ VSAM RLS/CICS data set recovery
- ▶ Backup and recovery of VSAM data sets
- ▶ The SMSVSAM address space
- ▶ DFSMStvs introduction
- ▶ Accessing a data set with DFSMStvs
- ▶ Application considerations
- ▶ The DFSMStvs instance

7.1 VSAM record-level sharing introduction

VSAM RLS is a method of access to your existing VSAM files that provides full read and write integrity at the *record level* to any number of users in your Parallel Sysplex.

This capability enables different tasks to access the same data set at the same time, while ensuring data integrity. VSAM RLS reduces the contention between different tasks by locking data at a record level, leaving the other logical records within a control interval (CI) available for read/write by other tasks.

Figure 7-1 shows a sample implementation of VSAM RLS with multiple CICS.

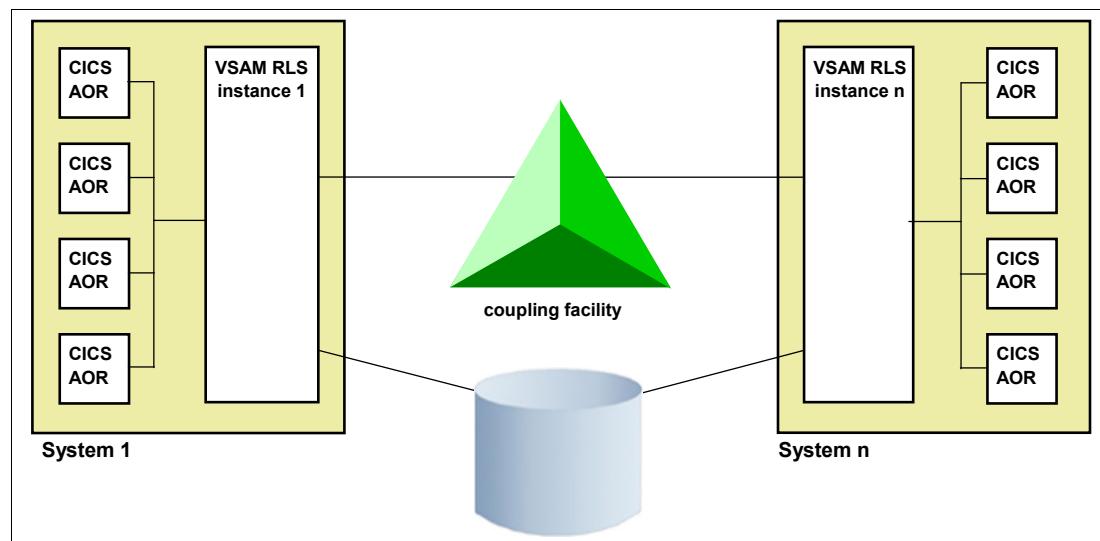


Figure 7-1 Parallel Sysplex CICS with VSAM RLS

VSAM RLS provides these benefits:

- ▶ Enhances cross-system data sharing when the scope is sysplex.
- ▶ Improves performance and availability in CICS and non-CICS VSAM environments.
- ▶ Provides data protection after a system failure.
- ▶ Provides automation for data recovery.
- ▶ Provides full read/write integrity to your existing VSAM files. The user does not need to serialize using ENQ/DEQ macros.
- ▶ Allows CICS to register as a recoverable subsystem, which automates recovery processing and protects the data records to be recovered.

The level of sharing that is allowed between applications is determined by whether a data set is recoverable:

- ▶ Both CICS and non-CICS jobs can have concurrent read or write access to unrecoverable data sets. There is no coordination between CICS and non-CICS, so data integrity can be compromised.
- ▶ Non-CICS jobs can have read-only access to recoverable data sets concurrently with CICS jobs, which can have read or write access.

7.1.1 Supported data set types

VSAM RLS supports access to these types of data sets:

- ▶ Key-sequenced data set (KSDS)
- ▶ Entry-sequenced data set (ESDS)
- ▶ Relative record data set (RRDS)
- ▶ Variable-length relative-record data set cluster (VRRDS)

VSAM RLS also supports access to a data set through an alternate index, but it does not support opening an alternate index directly in VSAM RLS mode. Also, VSAM RLS does not support access through an alternate index to data stored under z/OS UNIX System Services.

Extended format, extended addressability, and spanned data sets are supported by VSAM RLS. Compression is also supported.

VSAM RLS does not support these objects:

- ▶ Linear data sets (LDS)
- ▶ Keyrange data sets
- ▶ KSDS with an imbedded index (defined with IMBED option)
- ▶ Temporary data sets
- ▶ Striped data sets
- ▶ Master catalogs and VVDSs

Keyrange data sets and the IMBED attribute for a KSDS are obsolete. You cannot define new data sets as keyrange or with an imbedded index anymore. However, there still might be old data sets with these attributes in your installation.

7.1.2 Coupling facility overview

The coupling facility (CF) is a shareable storage medium. It is Licensed Internal Code (LIC) running in a special type of IBM PR/SM™ logical partition (LPAR) in certain zSeries processors. It can be shared by the systems in one sysplex only. A CF makes data sharing possible by allowing data to be accessed throughout a sysplex with assurance that the data will not be corrupted and that the data will be consistent among all sharing users.

VSAM RLS uses a CF to perform data-set-level locking, record locking, and data caching. VSAM RLS uses the conditional write and cross-invalidate functions of the CF cache structure, avoiding the need for control interval (CI) level locking.

VSAM RLS uses the CF caches as store-through caches. When a control interval of data is written, it is written to both the Coupling Facility cache and the direct access storage device (DASD). This feature ensures that problems occurring with a CF cache do not result in the loss of VSAM data.

7.1.3 VSAM RLS sharing control data sets

The sharing control data set (SHCDS) is designed to contain the information required for DFSMS to continue processing with a minimum of unavailable data and no corruption of data when failures occur. The SCDS data can be either SMS or non-SMS managed.

The SHCDS contains the following data:

- ▶ Name of the CF lock structure in use
- ▶ System status for each system or failed system instance
- ▶ Time that the system failed
- ▶ List of subsystems and their status

- ▶ List of open data sets that use the CF
- ▶ List of data sets with unbound locks
- ▶ List of data sets in permit non-VSAM RLS state

Both the primary and secondary SHCDS contain the same data. With the duplexing of the data, VSAM RLS ensures that processing can continue in case VSAM RLS loses connection to one SHCDS or the control data set got damaged. In that case, you can switch the spare SHCDS to active.

To calculate the size of the sharing control data sets, follow the guidelines that are provided in *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

Tip: Place the SHCDSs on separate volumes to maximize availability. Avoid placing SHCDSs on volumes for which there might be extensive volume reserve activity.

7.1.4 Data set sharing under VSAM RLS

The VSAM share options specification applies only when non-VSAM RLS access like NSR, LSR, or GSR is used. They are ignored when the data set is open for VSAM RLS access. Record-level sharing always assumes multiple readers and writers to the data set. VSAM RLS ensures full data integrity. When a data set is open for VSAM RLS access, non-VSAM RLS requests to open the data set fail.

Exception: SHAREOPTIONS(2,x)

For non-VSAM RLS access, SHAREOPTIONS(2,x) handled the same as VSAM RLS access. One user can have the data set open for read/write access and multiple users can have it open for read access only. VSAM does not provide data integrity for the readers.

If the data set is open for VSAM RLS access, non-VSAM RLS opens for *read* are possible. These are the only share options, where a non-VSAM RLS request to open the data set will not fail if the data set is already open for VSAM RLS processing. VSAM does not provide data integrity for non-VSAM RLS readers.

Non-CICS access

VSAM RLS access from batch jobs to data sets that are open by CICS depends on whether the data set is recoverable or not. For recoverable data sets, non-CICS access from other applications (that do not act as recoverable resource manager) is not allowed.

7.2 VSAM RLS locking

The granularity under LSR, NSR, or GSR is a *control interval*, whereas VSAM RLS serializes on a *record level*. With VSAM RLS, it is possible to concurrently update separate records in the same control interval. Record locks for UPDATE are always exclusive. Record locks for read depend on the level of read integrity.

Read integrity has three levels:

- ▶ NRI (no read integrity)

This level tells VSAM not to obtain a record lock on the record accessed by a GET or POINT request. This level avoids the overhead of record locking. This is sometimes referred to as a *dirty read* because the reader might see an uncommitted change made by another transaction.

Even with this option specified, VSAM RLS still performs buffer validity checking and refreshes the buffer when the buffer is invalid.

- ▶ CR (consistent read)

This level tells VSAM to obtain a *shared lock* on the record that is accessed by a GET or POINT request. It ensures that the reader does not see an uncommitted change made by another transaction. Instead, the GET or POINT request waits for the change to be committed or backed out. The request also waits for the exclusive lock on the record to be released.

- ▶ CRE (consistent read explicit)

This level has a meaning similar to that of CR, except that VSAM RLS holds the shared lock on the record until the end of the unit of recovery, or unit of work. This option is only available to CICS or DFSMStvs transactions. VSAM RLS does not understand end-of-transaction for non-CICS or non-DFSMStvs usage.

Figure 7-2 shows the following situation:

1. CICS transaction Tran1 obtains an exclusive lock on Record B for update processing.
2. Transaction Tran2 obtains an exclusive lock for update processing on Record E, which is in the same CI.
3. Transaction Tran3 needs a shared lock also on Record B for consistent read. It must wait until the exclusive lock by Tran1 is released.
4. Transaction Tran4 does a dirty read (NRI). It does not have to wait because in that case, no lock is necessary.

With NRI, Tran4 can read the record even though it is held exclusively by Tran1. There is no read integrity for Tran4.

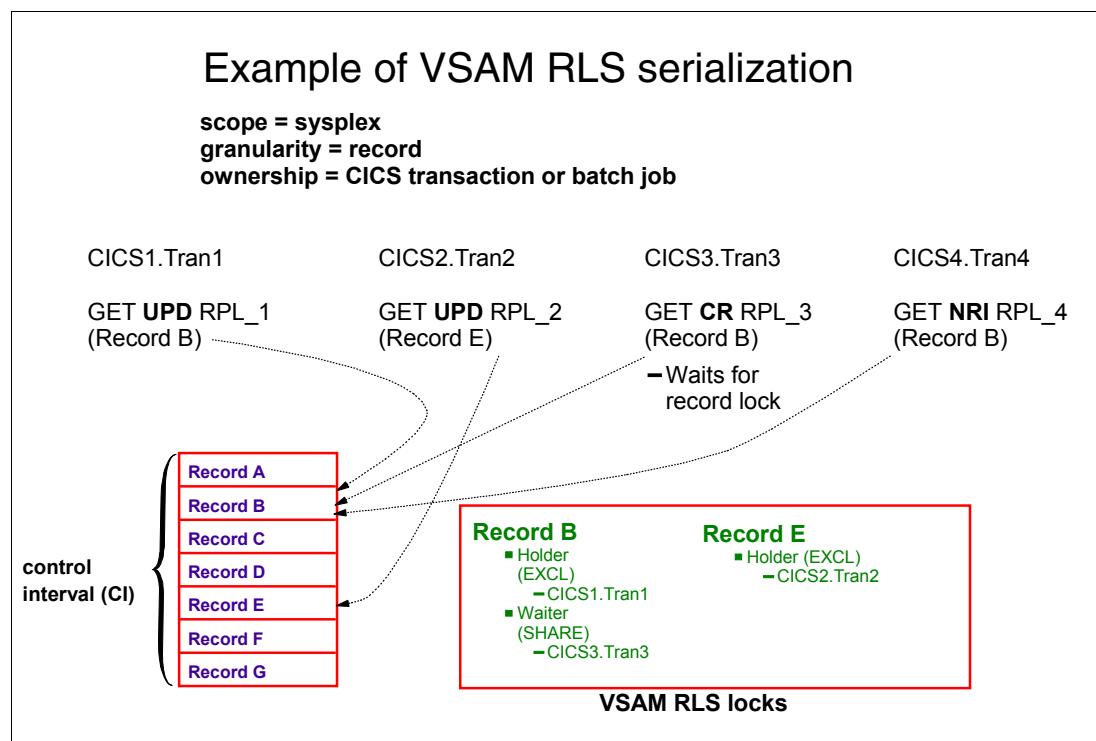


Figure 7-2 Example of VSAM RLS serialization

VSAM RLS locking is performed in the coupling facility by using a CF lock structure (IGWLOCK00) and the XES locking services.

When contention occurs on a VSAM record, the request that encountered the contention waits for the contention to be removed. The lock manager provides deadlock detection. When a lock request is in deadlock, the request is rejected, resulting in the VSAM record management request completing with a deadlock error response.

7.3 Buffering under VSAM RLS

VSAM RLS is another method of buffering that you can specify in the MACRF parameter of the ACB macro. VSAM RLS and NSR/LSR/GSR are mutually exclusive. Unlike NSR, LSR, or GSR, the VSAM buffers are in a data space and not in private or global storage of a user address space. Each image in the sysplex has one large local buffer pool in the data space.

The first request for a record after data set open for VSAM RLS processing causes an I/O operation to read in the CI that contains this record. A copy of the CI is stored into the cache structure of the coupling facility and in the buffer pool in the data space.

Buffer coherency is maintained by using CF cache structures and the cross-system coupling facility (XCF) cross-validation function. For the example in Figure 7-3 on page 125, that means these steps:

1. System 1 opens the VSAM data set for read/write processing.
2. System 1 reads in CI1 and CI3 from DASD. Both CIs are stored in the cache structure in the Coupling Facility.
3. System 2 opens the data set for read processing.
4. System 2 needs CI1 and CI4. CI1 is read from the CF cache, and CI4 from DASD.
5. System 1 updates a record in CI1 and CI3. Both copies of these CIs in the CF are updated.
6. XCF notices the change of these two CIs and invalidates the copy of CI1 for System 2.
7. System 2 needs another record from CI1. It notices that its buffer was invalidated and reads in a new copy of CI1 from the CF.

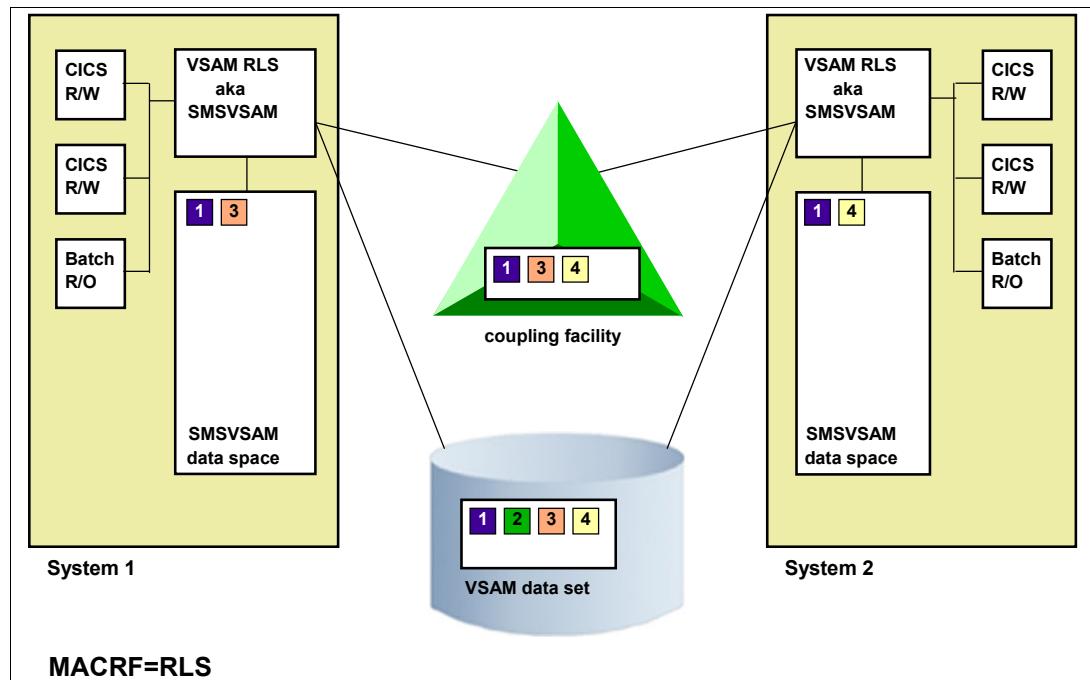


Figure 7-3 Buffering under VSAM RLS

For further information about cross-invalidation, see *z/OS MVS Programming: Sysplex Services Guide*, SA22-7617.

7.4 VSAM RLS/CICS data set recovery

VSAM record-level sharing introduces a VSAM data set attribute called LOG. With this attribute, a data set can be defined as *recoverable* or *non-recoverable*. A data set whose log parameter is undefined or NONE is considered non-recoverable. A data set whose log parameter is UNDO or ALL is considered recoverable. For recoverable data sets, a log of changed records is maintained to commit and back out transaction changes to a data set.

A data set is considered recoverable if the LOG attribute has one of the following values:

- ▶ UNDO

The data set is *backward* recoverable. Changes made by a transaction that does not succeed (no commit was done) are backed out. This is also referred as transactional recovery.

- ▶ ALL

The data set is both *backward* and *forward* recoverable. In addition to the logging and recovery functions provided for backout (transactional recovery), CICS records the image of changes to the data set after they were made.

The forward recovery log records are used by forward recovery programs and products such as CICS VSAM Recovery (CICSVR) to reconstruct the data set in the event of hardware or software damage to the data set. This is referred to as *data set recovery*. For LOG(ALL) data sets, both types of recovery are provided: Transactional recovery and data set recovery. For LOG(ALL), you need to define a logstream in which changes to the data sets are logged.

A data set whose LOG parameter is undefined or NONE is considered as non-recoverable.

7.5 Backup and recovery of VSAM data sets

DFSMSSdss supports backup-while-open (BWO) serialization, which can perform backups of data sets that are open for update for long periods of time. It can also perform a logical data set dump of these data sets even if another application has them serialized.

Backup-while-open is a better method than using SHARE or TOLERATE(ENQFAILURE) for dumping VSAM data sets that are in use and open for update.

To allow DFMSDSS to take a backup while your data set is open, you need to define the data set with the BWO attribute or assign a data class with this attribute.

If you use the DSS BWO processing for a forward recoverable data set, CICS will log the start and end of the copy/backup operation. The data set can then be fully recovered from this backup.

For information about BWO processing, see *z/OS DFMSDSS Storage Administration Reference*, SC35-0424.

7.5.1 Recover a VSAM data set

Sometimes it is necessary to recover a data set, such as if it becomes broken. Recovery differs based on the type of data set:

- ▶ Recovery of a non-recoverable data set

Data sets with LOG(NONE) are considered non-recoverable. To recover such a data set, restore the last backup of the data set. All updates to this data set after the backup was taken are lost. If the backup was taken after a transaction failed (did not commit), the data in the backup might be inconsistent.

- ▶ Recovery of a backward recoverable data set

Data sets with the LOG(UNDO) attribute are considered backward recoverable. To recover such a data set, restore the last backup of the data set. All updates to this data set after the backup was taken are lost. The data in the backup is consistent.

- ▶ Recovery of a forward recoverable data set

Data sets with LOG(ALL) and a log stream assigned are forward recoverable. Restore the last backup of the data set. Then, run a tool like CICSRV, which uses the forward recovery log to redrive all committed updates until the data set became broken. No updates are lost.

7.5.2 Transactional recovery

During the life of a transaction, its changes to recoverable resources are *not* seen by other transactions. The exception is if you are using the NRI option, in which case you might see uncommitted changes.

Exclusive locks that VSAM RLS holds on the modified records cause other transactions that have read-with-integrity requests and write requests for these records to wait. After the modifying transaction is committed or backed out, VSAM RLS releases the locks and the other transactions can access the records.

If the transaction fails, its changes are backed out. This capability is called *transactional recovery*.

The CICS backout function removes changes made to the recoverable data sets by a transaction. When a transaction abnormally ends, CICS performs a backout implicitly.

In the example in Figure 7-4, transaction Trans1 is complete (committed) after Record 1 and Record 2 are updated. Transactional recovery ensures that either both changes are made or no change is made. When the application requests *commit*, both changes are made atomically. In the case of a failure after updating Record 1, the change to this record is backed out. This feature applies only for recoverable data sets, not for non-recoverable ones.

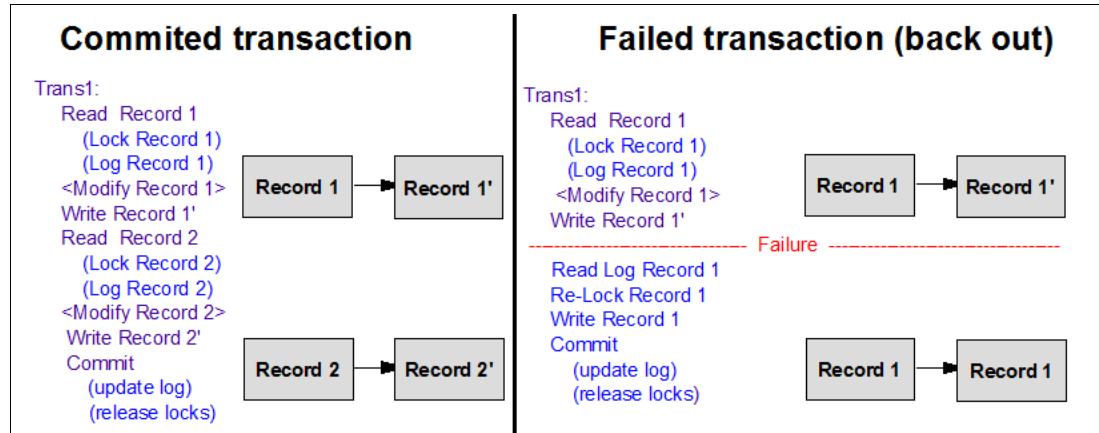


Figure 7-4 Sample committed and failed transactions

7.6 The SMSVSAM address space

SMSVSAM is the MVS job name of the VSAM RLS address space. It is started automatically at IPL time if RLSINIT(YES) is specified in the IGDSMSxx member of SYS1.PARMLIB, or was started by operator command.

The SMSVSAM address space needs to be started on each system where you want to use VSAM RLS. It is responsible for centralizing all processing necessary for cross-system sharing, which includes one connect per system to XCF lock, cache, and VSAM control block structures.

The SMSVSAM address space owns two data spaces:

- ▶ **SMSVSAM**
This space contains VSAM RLS control blocks and a system-wide buffer pool.
- ▶ **MMFSTUFF**
This space collects activity monitoring information that is used to produce SMF records.

The following terms are used to describe a VSAM RLS environment:

- ▶ **RLS server**
The SMSVSAM address space is also referred to as the RLS server.
- ▶ **RLS client**
Any address space that calls an RLS function that results in a program call to the SMSVSAM address space is called an RLS client. Those address spaces can be online applications and batch jobs.
- ▶ **Recoverable subsystem**

A subsystem is an RLS client space that *registers* with the SMSVSAM address space as an address space that provides transactional and data set recovery. CICS, for example, is a *recoverable subsystem*.

- ▶ Batch job

An RLS client space that does *not* first register with SMSVSAM as a recoverable subsystem is called a batch job. An example of such a batch job is HSM.

7.7 DFSMStvs introduction

The objective of DFSMStvs is to provide transactional recovery directly within VSAM. It is an extension to VSAM RLS. It allows any job or application that is designed for data sharing to read/write share VSAM recoverable files.

DFSMStvs is a follow-on project/capability based on VSAM RLS. VSAM RLS supports CICS and IMS as transaction managers. This feature provides sysplex data sharing of VSAM recoverable files when accessed through CICS or IMS. They provide the necessary unit-of-work management, undo/redo logging, and commit/back out functions. VSAM RLS provides the underlying sysplex-scope locking and data access integrity.

DFSMStvs adds logging and commit/back out support to VSAM RLS. DFSMStvs requires and supports the recoverable resource management services (RRMS) component as the commit or sync point manager.

DFSMStvs provides a level of data sharing with built-in transactional recovery for VSAM recoverable files that is comparable with the data sharing and transactional recovery support for databases provided by DB2 and IMSDB.

DFSMStvs enhances VSAM RLS to perform data recovery in these forms:

- ▶ Transactional recovery (see 7.5.2, “Transactional recovery” on page 126)
- ▶ Data set recovery (see 7.5, “Backup and recovery of VSAM data sets” on page 126)

Before DFSMStvs, those two types of recovery were only supported by CICS.

CICS performs the transactional recovery for data sets defined with the LOG parameter UNDO or ALL.

For forward recoverable data sets (LOG(ALL)), CICS also records updates in a log stream for forward recovery. CICS itself does not perform forward recovery. It only performs logging. For forward recovery, you need a utility like CICSVR.

Like CICS, DFSMStvs also provides transactional recovery and logging.

Without DFSMStvs, batch jobs cannot perform transactional recovery and logging. That is the reason batch jobs were granted only read access to a data set that was opened by CICS in RLS mode. A batch window was necessary to run batch updates for CICS VSAM data sets.

With DFSMStvs, batch jobs can perform transactional recovery and logging concurrently with CICS processing. Batch jobs can now update data sets while they are in use by CICS. No batch window is necessary anymore.

Like CICS, DFSMStvs does not perform data set forward recovery.

7.7.1 Components used by DFSMStvs

The following components are involved in DFSMStvs processing:

- ▶ RRMS to manage the unit of recovery

DFSMStvs uses the RRMS to manage the unit of recovery that is needed for transactional recovery. For more information about RRMS, see 7.7.2, “DFSMStvs use of z/OS RRMS” on page 129.

- ▶ System logger to manage log streams

There are three kinds of logs used by DFSMStvs:

- Undo log: Used for transactional recovery (back out processing)
- Forward recovery log: Used to log updates against a data set for forward recovery
- Shunt log: Used for long running or failed units of recovery

You can also define a log of logs for use to automate forward recovery.

These logs are maintained by the MVS system logger. For information about the various log types, see 7.7.5, “DFSMStvs logging” on page 131.

- ▶ VSAM RLS

Used for record locking and buffering. The need for VSAM RLS is discussed in prior sections.

Atomic commit of changes

DFSMStvs allows atomic commit of changes. That means, if multiple updates to various records are necessary to complete a transaction, either all updates are committed or none are, if the transaction does not complete. See also 7.5.2, “Transactional recovery” on page 126 and 7.7.3, “Atomic updates” on page 130.

Peer recovery

Peer recovery allows DFSMStvs to recover for a failed DFSMStvs instance to clean up any work that was left in an incomplete state and to clear retained locks that resulted from the failure. For more information about peer recovery, see *z/OS DFSMStvs Planning and Operation Guide*, SC26-7348.

7.7.2 DFSMStvs use of z/OS RRMS

z/OS provides RRMS that includes these features:

- ▶ Registration services
- ▶ Context services
- ▶ Resource Recovery Services (RRS), which acts as sync point manager

RRS provides the sync point services and is the most important component from a DFSMStvs use perspective.

DFSMStvs is a *recoverable* resource manager. It is *not* a commit or sync point manager. DFSMStvs interfaces with the z/OS sync point manager (RRS).

When an application issues a commit request directly to z/OS or indirectly through a sync point manager that interfaces with the z/OS sync point manager, DFSMStvs is called to participate in the two-phase commit process.

Other resource managers (like DB2) whose recoverable resources were modified by the transaction are also called by the z/OS sync point manager. This process provides a commit scope across the multiple resource managers.

RRS is a system-wide commit coordinator. It enables transactions to update protected resources managed by many resource managers.

It is RRS that provides the means to implement two-phase commit, but a resource manager must also use registration services and context services with Resource Recovery Services.

Two-phase commit

The two-phase commit protocol is a set of actions used to make sure that an application program either makes *all* changes to the resources represented by a single unit of recovery (UR), or it makes *no* changes at all. This protocol verifies that either all changes or no changes are applied even if one of the elements (such as the application, the system, or the resource manager) fails. The protocol allows for restart and recovery processing to take place after system or subsystem failure.

For a discussion of the term *unit of recovery*, see 7.7.4, “Unit of work and unit of recovery” on page 131.

7.7.3 Atomic updates

A transaction is known as *atomic* when an application changes data in multiple resource managers as a single transaction, and all of those changes are accomplished through a single commit request by a sync point manager. If the transaction is successful, all the changes are committed. If any piece of the transaction is not successful, then all changes are backed out.

An *atomic instant* occurs when the sync point manager in a two-phase commit process logs a commit record for the transaction. Figure 7-5 shows a sample atomic update.

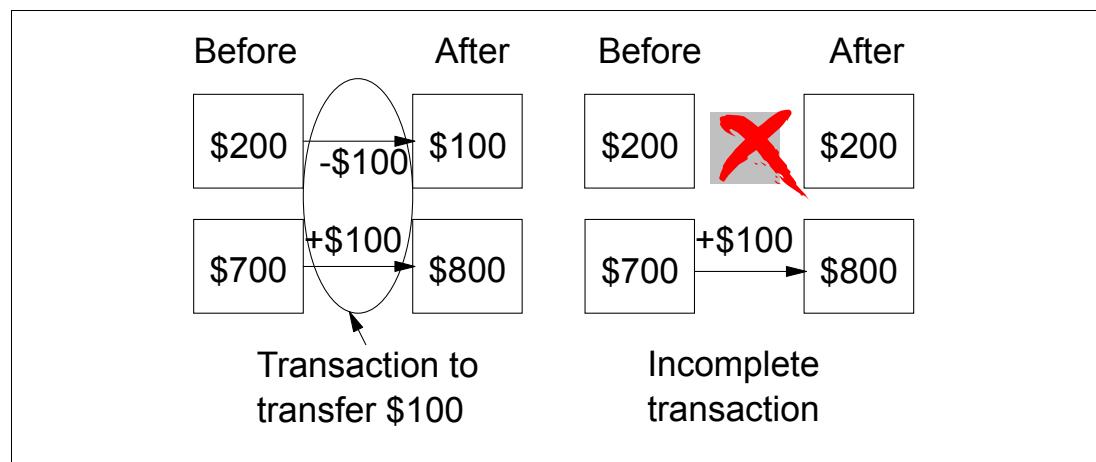


Figure 7-5 Example of an atomic update

For more information about recovering an uncompleted transaction, 7.5.2, “Transactional recovery” on page 126.

7.7.4 Unit of work and unit of recovery

A unit of work (UOW) is the term used in CICS publications for a set of updates that are treated as an atomic set of changes.

RRS uses unit of recovery to mean much the same thing. Thus, a unit of recovery is the set of updates between synchronization points. There are implicit synchronization points at the start and at the end of a transaction. Explicit synchronization points are requested by an application within a transaction or batch job. It is preferable to use explicit synchronization for greater control of the number of updates in a unit of recovery.

Changes to data are durable after a synchronization point. That means that the changes survive any subsequent failure.

Figure 7-6 shows three units of recovery, noted as A, B, and C. The synchronization points between the units of recovery can be one of these types:

- ▶ Implicit: At the start and end of the program
- ▶ Explicit: When requested by commit

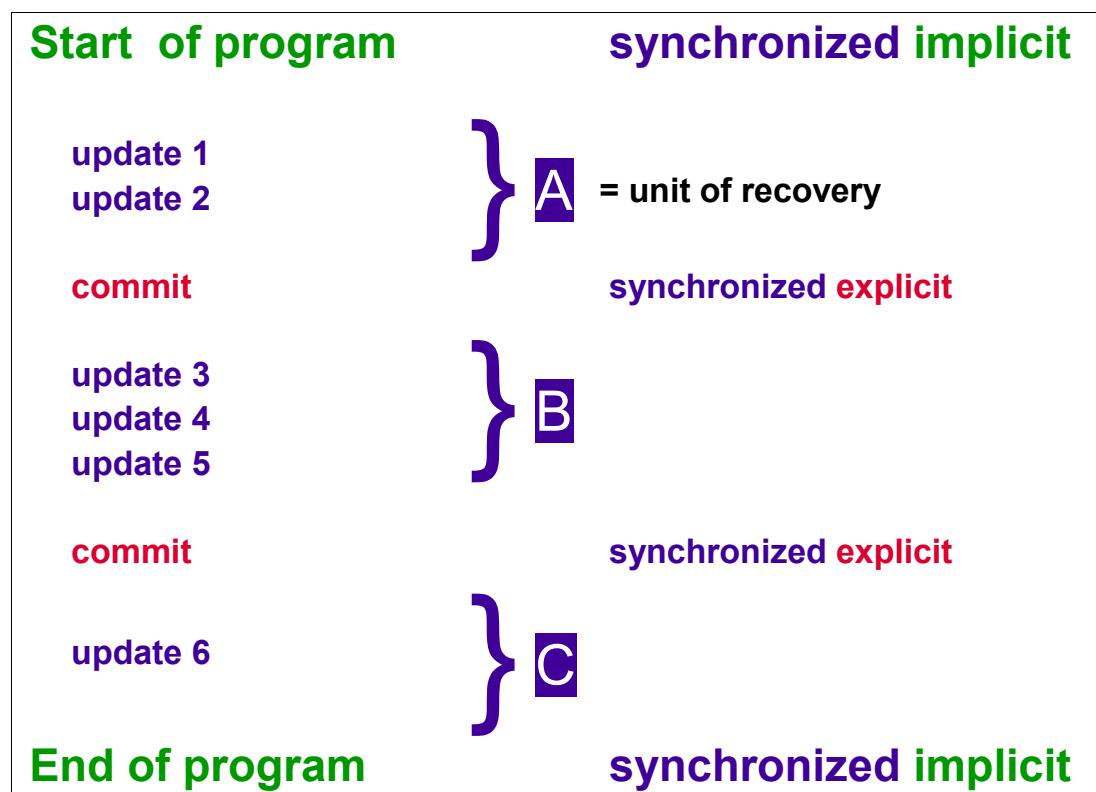


Figure 7-6 Unit of recovery example

7.7.5 DFSMStvs logging

DFSMStvs logging uses the z/OS system logger. The design of DFSMStvs logging is similar to the design of CICS logging. Forward recovery logstreams for VSAM recoverable files will be shared across CICS and DFSMStvs. CICS will log changes made by CICS transactions, while DFSMStvs will log changes made by its callers.

The system logger

The system logger (IXGLOGR) is an MVS component that provides a rich set of services that allow another component or an application to write, browse, and delete log data. The system logger is used because it can merge log entries from many z/OS images to a single log stream, where a log stream is simply a set of log entries.

Types of logs

Various types of logs are involved in DFSMStvs (and CICS) logging:

- ▶ Undo logs (mandatory, one per image) - tvsname.IGWLOG.SYSLOG
 - The backout or undo log contains images of changed records for recoverable data sets as they existed before being changed. It is used for transactional recovery to back out uncommitted changes if a transaction failed.
- ▶ Shunt logs (mandatory, one per image) - tvsname.IGWSHUNT.SHUNTLOG
 - The shunt log is used when backout requests fail and for long running units of recovery.
- ▶ Log of logs (optional, shared with CICS and CICSVR) - Default name is CICSUSER.CICSVR.DFHLOGLOG
 - The log of logs contains copies of log records that are used to automate forward recovery.
- ▶ Forward recovery logs (optional, shared with CICS) - Name of your choice
 - The forward recovery log is used for data sets defined with LOG(ALL). It is used for forward recovery. It needs to be defined in the LOGSTREAMID parameter of the data set.

The system logger writes log data to log streams. The log streams are put in list structures in the Coupling Facility (except for DASDONLY log streams).

As Figure 7-7 shows, you can merge forward recovery logs for use by CICS and DFSMStvs. You can also share it by multiple VSAM data sets. You cannot share an undo log by CICS and DFSMStvs; you need one per image.

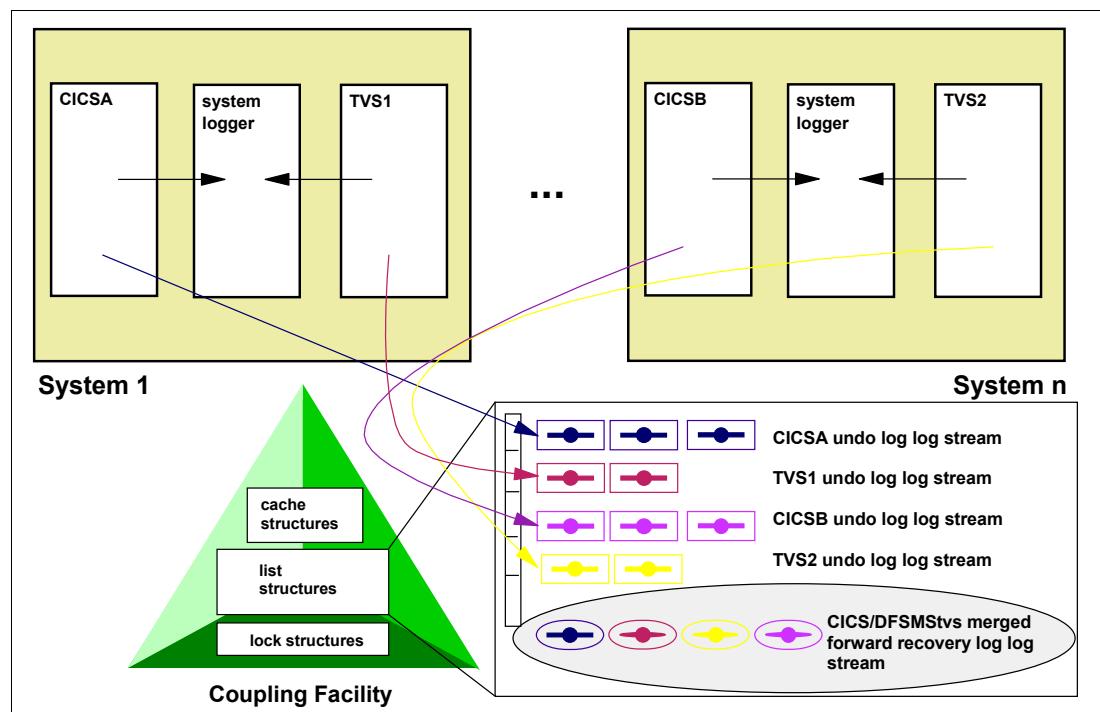


Figure 7-7 DFSMStvs logging

7.8 Accessing a data set with DFSMStvs

The MACRF options NRI, CR, and CRE are described in 7.2, “VSAM RLS locking” on page 122. CRE gives DFSMStvs access to VSAM data sets open for input or output. CR or NRI gives DFSMStvs access to VSAM recoverable data sets only for output.

You can modify an application to use DFSMStvs by specifying RLS in the job control language (JCL) or the ACB and having the application access a recoverable data set using either open for input with CRE or open for output from a batch job.

Table 7-1 shows the cases in which DFSMStvs is called.

Table 7-1 DFSMStvs and VSAM RLS usage

Data set type and Type of OPEN	NRI	CR	CRE
Recoverable open for input	VSAM RLS	VSAM RLS	DFSMStvs
Recoverable open for output	DFSMStvs	DFSMStvs	DFSMStvs
Non-recoverable open for input	VSAM RLS	VSAM RLS	DFSMStvs
Non-recoverable open for output	VSAM RLS	VSAM RLS	DFSMStvs

DFSMStvs is always called in these circumstances:

- ▶ RLS and CRE are specified in the JCL or MACRF parameter of the ACB macro
CRE is also known as *repeatable read*.
- ▶ RLS is specified in the JCL or MACRF parameter of the ACB macro *and* the data set is recoverable *and* it is opened for output (update processing)
This feature allows DFSMStvs to provide the necessary transactional recovery for the data set.

7.9 Application considerations

For an application to participate in transactional recovery, it must first understand the concept of a transaction. It is *not* a good idea to modify an existing batch job to use DFSMStvs with no further change, as doing so causes the entire job to be seen as a single transaction. As a result, locks would be held and log records would need to exist for the entire life of the job. This circumstance can cause a tremendous amount of contention for the locked resources. It can also cause performance degradation as the undo log becomes exceedingly large.

7.9.1 Break processing into a series of transactions

To use the DFSMStvs capabilities, break down application processing into a series of transactions. Have the application issue frequent sync points by calling RRS commit and backout processing (MVS callable services SRRCMIT and SRRBACK).

VSAM RLS and DFSMStvs provide isolation until commit/backout. Consider the following rules:

- ▶ Share locks on records accessed with *repeatable read*.
- ▶ Hold write locks on changed records until the end of a transaction.
- ▶ Use commit to apply all changes and release all locks.

- ▶ Information extracted from shared files must not be used across commit/backout for the following reasons:
 - Need to reaccess the records.
 - Cannot get a record before a sync point and update it after.
 - Do not position to a record before a sync point and access it after.

7.9.2 Modify the program or JCL to request DFSMStvs access

You need to change the ACB MACRF or JCL specifications for a data set to request DFSMStvs access. See the previous section for more information.

Prevent contention within a unit of recovery

If the application uses multiple RPLs, care must be taken in how they are used. Using various RPLs to access the same record can cause lock contention within a UR.

Handle potential loss of positioning at sync point

The batch application must have a built-in method of tracking its processing position within a series of transactions. One potential method of doing this is to use a VSAM recoverable file to track the job's commit position.

Do not use file backup/restore as a job restart technique

Today's batch applications that update VSAM files in a non-shared environment can create backup copies of the files to establish a restart/recovery point for the data. If the batch application fails, the files can be restored from the backup copies, and the batch jobs can be executed again. This restart/recovery procedure *cannot* be used in a data sharing DFSMStvs environment because restoring to the point-in-time backup erases changes made by other applications that share the data set.

Instead, the batch application must have a built-in method of tracking its processing position within a series of transactions. One potential method of doing this is to use a VSAM recoverable file to track the job's commit position. When the application fails, any uncommitted changes are backed out.

The already-committed changes cannot be backed out because they are already visible to other jobs or transactions. In fact, it is possible that the records that were changed by previously-committed UR were changed again by other jobs or transactions. Therefore, when the job is rerun, it is important that it determines its restart point and not attempt to redo any changes it had committed before the failure.

For this reason, it is important that jobs and applications using DFSMStvs be written to run as a series of transactions and use a commit point tracking mechanism for restart.

7.10 The DFSMStvs instance

DFSMStvs runs in the SMSVSAM address space as an instance. The name of the DFSMStvs address space is always IGWTV nnn , where nnn is a unique number per system as defined for TVSNAME in the PARMLIB member IGDSMSxx. This number can be in the range 0 - 255. You can define up to 32 DFSMStvs instances, one per system. An example is IGWTV001.

As soon as an application that does not act as a recoverable resource manager has VSAM RLS access to a recoverable data set, DFSMStvs is called (see also 7.8, “Accessing a data set with DFSMStvs” on page 133). DFSMStvs calls VSAM RLS (SMSVSAM) for record locking and buffering. With DFSMStvs built on top of VSAM RLS, full sharing of recoverable files becomes possible. Batch jobs can now update the recoverable files without first quiescing CICS’s access to them.

As a recoverable resource manager, CICS interacts directly with VSAM RLS.

7.10.1 Interacting with DFSMStvs

This section provides a brief overview of a few commands that were introduced particularly for displaying and changing DFSMStvs-related information.

There are a few display commands you can use to get information about DFSMStvs:

- ▶ To display common DFSMStvs information:

```
DISPLAY SMS,TRANVSAM{,ALL}
```

This command lists information about the DFSMStvs instance on the system where it was issued. To get information from all systems, use ALL. This information includes name and state of the DFSMStvs instance, values for AKP, start type, and the quiesce timeout value, and also the names, types, and states of the used log streams.

- ▶ To display information about a particular job that uses DFSMStvs:

```
DISPLAY SMS,JOB(jobname)
```

The information about the particular job includes the current job step, the current ID, and status of the unit of recovery used by this job.

- ▶ To display information about a particular unit of recovery currently active within the sysplex:

```
DISPLAY SMS,URID(urid|ALL)
```

This command provides information about a particular UR in the sysplex or about all URs of the system on which this command was issued. If ALL is specified, you do not obtain information about shunted URs and URs that are restarting. The provided information includes age and status of the UR, the jobname with which this UR is associated, and the current step within the job.

- ▶ To display entries currently contained in the shunt log:

```
DISPLAY SMS,SHUNTED,{SPHERE(sphere)|  
URID(urid|ALL)}
```

Entries are moved to the shunt log when DFSMStvs is unable to finish processing a sync point for them. There are several reasons that this might occur, including an I/O error. Depending on what was specified, you get information for a particular VSAM sphere, a particular UR, or for all shunted URs in the sysplex.

- ▶ To display information about logstreams that DFSMStvs is currently using:

```
DISPLAY SMS,LOG(logstream|ALL)
```

If ALL is specified, information about all log streams in use is provided from the system on which the command is issued. The output includes the status and type of the log stream, the job name, and URID of the oldest unit of recovery using the log. It also includes a list of all DFSMStvs instances that are using the log.

- ▶ To display information about a particular data set:

```
DISPLAY SMS,DSNAME(dsn)
```

Use this command to display jobs currently accessing the data set using DFSMStvs access on the systems within the sysplex.

IDCAMS command SHCDS

The IDCAMS command **SHCDS** was enhanced to display and modify recovery information related to DFSMStvs. For more information about this command, see *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

Important: This chapter simply provided an overview of new operator commands to understand how to work with DFSMStvs. Before using these commands (other than the DISPLAY command), read the official z/OS manuals carefully.



Storage management hardware

The use of Data Facility Storage Management Subsystem (DFSMS) requires storage management hardware that includes both direct access storage device (DASD) and tape device types. This chapter provides an overview of both storage device categories and a brief introduction to RAID technology.

For many years, DASDs have been the most used storage devices on IBM eServer™ zSeries systems and their predecessors. DASDs deliver the fast random access to data and high availability that customers have come to expect. This chapter covers the following types of DASD:

- ▶ Traditional DASD
- ▶ DS8000 family

The era of tapes began before DASD was introduced. During that time, tapes were used as the primary application storage medium. Today, customers use tapes for such purposes as backup, archiving, and data transfer between companies. The following types of tape devices are described:

- ▶ IBM TS1155
- ▶ IBM TS4500
- ▶ IBM TS7700

This chapter also briefly explains the storage area network (SAN) concept.

This chapter includes the following sections:

- ▶ Overview of DASD types
- ▶ Redundant Array of Independent Disks
- ▶ IBM DS8000 series
- ▶ IBM TotalStorage Resiliency Family
- ▶ DS8000 performance features
- ▶ Introduction to tape processing
- ▶ IBM TS1155 tape drive
- ▶ IBM TS4500 tape library
- ▶ Introduction to IBM TS7700
- ▶ Introduction to TS7700 Grid configuration
- ▶ Storage area network

8.1 Overview of DASD types

In the era of traditional DASD, the hardware consisted of controllers like 3880 and 3990, which contained the necessary intelligent functions to operate a storage subsystem. The controllers were connected to S/390 systems through parallel or ESCON channels. Behind a controller, there were several model groups of the 3390 that contained the disk drives. Based on the models, these disk drives had various capacities per device.

Within each model group, the various models provide either four, eight, or twelve devices. All A-units come with four controllers, providing a total of four paths to the 3990 Storage Control. At that time, you were not able to change the characteristics of a given DASD device.

8.1.1 DASD based on RAID technology

With the introduction of the RAMAC Array in 1994, IBM first introduced storage subsystems for S/390 systems based on RAID technology. The various RAID implementations are shown in Figure 8-1 on page 139.

The more modern IBM DASD products, such as DS8000 family, and DASD from other vendors, emulate IBM 3380 and 3390 volumes in geometry, capacity of tracks, and number of tracks per cylinder. This emulation makes all the other entities think they are dealing with real 3380s or 3390s, reducing the needs from programmers to deal with different DASD technologies and architecture.

One benefit of this emulation is that it allows DASD manufacturers to implement changes in the real disks, including the geometry of tracks and cylinders, without affecting the way those components interface with DASD. From an operating system point of view, device types always will be 3390s, sometimes with much higher numbers of cylinders, but 3390s nonetheless.

Note: This publication uses the terms *disk* or *head disk assembly (HDA)* for the real devices, and the terms *DASD volumes* or *DASD devices* for the logical 3380/3390s.

8.2 Redundant Array of Independent Disks

Redundant Array of Independent Disks (RAID) is a direct-access storage architecture where data is recorded across multiple physical disks with parity separately recorded so that no loss of access to data results from the loss of any one disk in the array.

RAID breaks the one-to-one association of volumes with devices. A logical volume is now the addressable entity presented by the controller to the attached systems. The RAID unit maps the logical volume across multiple physical devices. Similarly, blocks of storage on a single physical device can be associated with multiple logical volumes. Because a logical volume is mapped by the RAID unit across multiple physical devices, it is now possible to overlap processing for multiple cache misses to the same logical volume because cache misses can be satisfied by separate physical devices.

The RAID concept involves many serial-attached SCSI (SAS) disks replacing a big one. RAID provides the following major RAID advantages:

- ▶ Performance (due to parallelism)
- ▶ Cost (SAS are commodities)
- ▶ zSeries compatibility
- ▶ Environment (space and energy)

Figure 8-1 shows some sample RAID configurations, which are explained in more detail next.

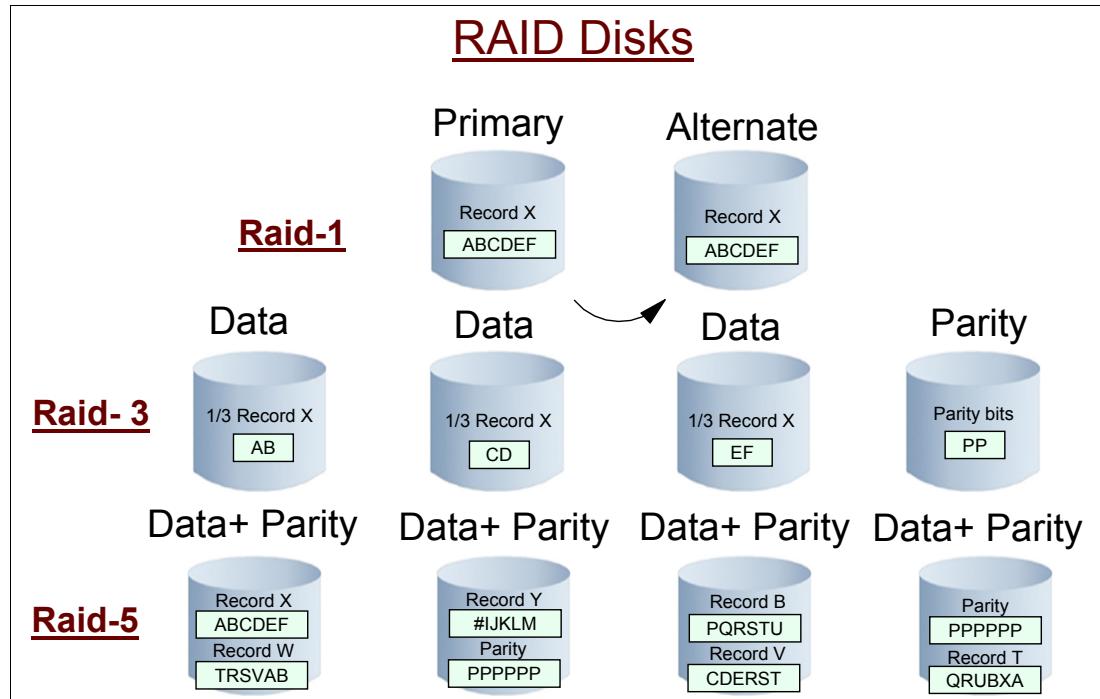


Figure 8-1 Redundant Array of Independent Disks (RAID)

However, RAID increased the chances of malfunction due to media and disk failures and the fact that the logical device is on many physical disks. The solution was redundancy, which wastes space and causes performance problems as “write penalty” and “free space reclamation.” To address this performance issue, large caches are implemented.

A *disk array* is a group of disk drive modules (DDMs) that are arranged in a relationship, for example, a RAID 5 or a RAID 10 array. For the DS8000, the arrays are built upon the disks of storage enclosures.

Note: The DS8000 storage controllers use the RAID architecture that enables multiple logical volumes to be mapped on a single physical RAID group. If required, you can still separate data sets on a physical controller boundary for availability.

8.2.1 RAID implementations

Except for RAID-1, each manufacturer sets the number of disks in an array. An *array* is a set of logically related disks, where a parity applies.

The following implementations are certified by the RAID Architecture Board:

- RAID-1** This implementation has simple disk mirroring, like dual copy.
- RAID-3** This implementation has an array with one dedicated parity disk and just one I/O request at a time, with intra-record striping. It means that the written physical block is striped and each piece (together with the parity) is written in parallel in each disk of the array. The access arms move together. It has a high data rate and a low I/O rate.

- RAID-5** This implementation has an array with one distributed parity (there is no dedicated disk for parities). It does I/O requests in parallel with extra-record striping, meaning each physical block is written in each disk. The access arms move independently. It has strong caching to avoid write penalties, involving four disk I/Os per write. RAID-5 has a high I/O rate and a medium data rate. RAID-5 does the following:
- ▶ It reads data from an undamaged disk. This is one single disk I/O operation.
 - ▶ It reads data from a damaged disk, which implies $(n-1)$ disk I/Os, to re-create the lost data where n is the number of disks in the array.
 - ▶ For every write to an undamaged disk, RAID-5 does four I/O operations to store a correct parity block. This is called a write penalty. This penalty can be relieved with strong caching and a slice triggered algorithm (coalescing disks updates from cache into a single parallel I/O).
 - ▶ For every write to a damaged disk, RAID-5 does $n-1$ reads and one parity write.
- RAID-6** This implementation has an array with two distributed parity and I/O requests in parallel with extra-record striping. Its access arms move independently (Reed/Salomon P-Q parity). The write penalty is greater than RAID-5, with six I/Os per write.
- RAID-6+** This implementation is without write penalty (due to large file storage (LFS)), and has background free-space reclamation. The access arms all move together for writes.
- RAID-10** RAID-10 is also known as RAID 1+0 because it is a combination of RAID 0 (striping) and RAID 1 (mirroring). The striping optimizes the performance by striping volumes across several disk drives. RAID 1 is the protection against a disk failure provided by having a mirrored copy of each disk. By combining the two, RAID 10 provides data protection and I/O performance.

Spare disks

The DS8000 requires that a device adapter loop have a minimum of two spare disks to enable sparing to occur. The sparing function of the DS8000 is automatically initiated whenever a DDM failure is detected on a device adapter loop and enables regeneration of data from the failed DDM onto a hot spare DDM.

Note: Data striping (stripe sequential physical blocks in separate disks) is sometimes called RAID-0, but it is not a real RAID because there is no redundancy, that is, no parity bits.

8.3 IBM DS8000 series

IBM DS8000 series is a high-performance, high-capacity series of disk storage that is designed to support continuous operations. DS8000 series models (machine type 2107) use the IBM POWER8® server technology that is integrated with the IBM Virtualization Engine technology. DS8000 series models consist of a storage unit and one or two management consoles, two being the preferred configuration. The graphical user interface (GUI) or the command-line interface (CLI) allows you to logically partition storage (create storage LPARs) and use the built-in Copy Services functions. For high availability, hardware components are redundant.

The DS8000 series system can scale to over 5.22 PB of raw drive capacity, with a wide range of capacity available for configuration, supporting any range of customer for smaller to very large storage systems.

The DS8000 series offers various choices of base and expansion models, so you can configure storage units that meet your performance and configuration needs:

- ▶ **DS8884 and DS8886**

DS8884 and DS8886 provide great scalability and performance, with over 5 PB of raw capacity drive with expansion frames, and up to 2 TB of system memory.

- ▶ **DS8884F, DS8886F, and DS8888F**

The DS888xF models feature all-flash drivers, and are optimized for performance and throughput by maximizing the number of paths to the storage enclosures.

The DS8000 expansion frames expand the capabilities of the base models, allowing greater capacity levels for your storage controller.

8.3.1 DS8000 hardware overview

This section provides an overview of available DS8000 hardware.

DS8884 (Model 984)

The IBM TotalStorage DS8884, which is Model 984, is a business class machine that offers the following features, among others:

- ▶ 6-core processor complex
- ▶ Up to 256 GB of system memory
- ▶ Up to two HPF Enclosures pairs (Gen-2)
- ▶ Up to eight standard drive enclosures
- ▶ Up to 16 host adapters
- ▶ Single-phase power
- ▶ Up to 192 2.5-inch disk drives
- ▶ Up to 48 flash cards
- ▶ Maximum capacity of up to 730 TB
- ▶ Up to 32 Fibre Channel/FICON ports

The DS8884 model can support two expansion frames. With two expansion frames, you can expand the capacity of the Model 984 as follows:

- ▶ Up to 768 small form-factor (SFF) drives, and 96 flash cards, for a maximum capacity of 2.61 PB

The DS8884 also supports an all-flash configuration, which is named DS8884F.

DS8886 (Models 986)

The IBM TotalStorage DS8886, which is Model 986, is a business class machine that offers the following features, among others:

- ▶ Up to 24-core processor complex
- ▶ Up to 2 TB of system memory
- ▶ Up to four HPF Enclosures pairs (Gen-2)
- ▶ Up to six standard drive enclosures
- ▶ Up to 16 host adapters
- ▶ Single-phase or three-phase power
- ▶ Up to 144 2.5-inch disk drives

- ▶ Up to 96 flash cards
- ▶ Maximum capacity of up to 739 TB
- ▶ Up to 64 Fibre Channel/FICON ports

The DS8886 model can support four expansion frames. With four expansion frames, you can expand the capacity of the Model 986 as follows:

- ▶ Up to 1536 SFF drives, and 192 flash cards, for a maximum capacity of 5.22 PB

The DS8886 also supports an all-flash configuration, which is named DS8886F.

DS888F (Models 988)

The IBM TotalStorage DS8886, which is Model 988, is a business class machine that offers the following features, among others:

- ▶ Up to 48-core processor complex
- ▶ Up to 2 TB of system memory
- ▶ Up to eight HPF Enclosures pairs (Gen-2)
- ▶ Up to 128 host adapters
- ▶ Three-phase power
- ▶ Up to 192 2.5-inch flash cards
- ▶ Maximum capacity of up to 614 TB
- ▶ Up to 64 Fibre Channel/FICON ports
- ▶ Up to 2.5 billion IOPS

The DS8888F model can support one expansion frame. With one expansion frame, you can expand the capacity of the Model 986 as follows:

- ▶ Up to 384 flash cards, for a maximum capacity of 1.23 PB
- ▶ Up to 128 Fibre Channel/FICON ports

8.3.2 DS8000 major components

Figure 8-2 shows an IBM TDS8886 Model 985 and its major components. As you can see, the machine that is displayed consists of two frames, each with its own power supplies, batteries, I/O enclosures, HPF enclosures, and standard drive enclosures. The base frame also includes the two IBM POWER8 processor-based servers, and the Hardware Management Console (HMC).

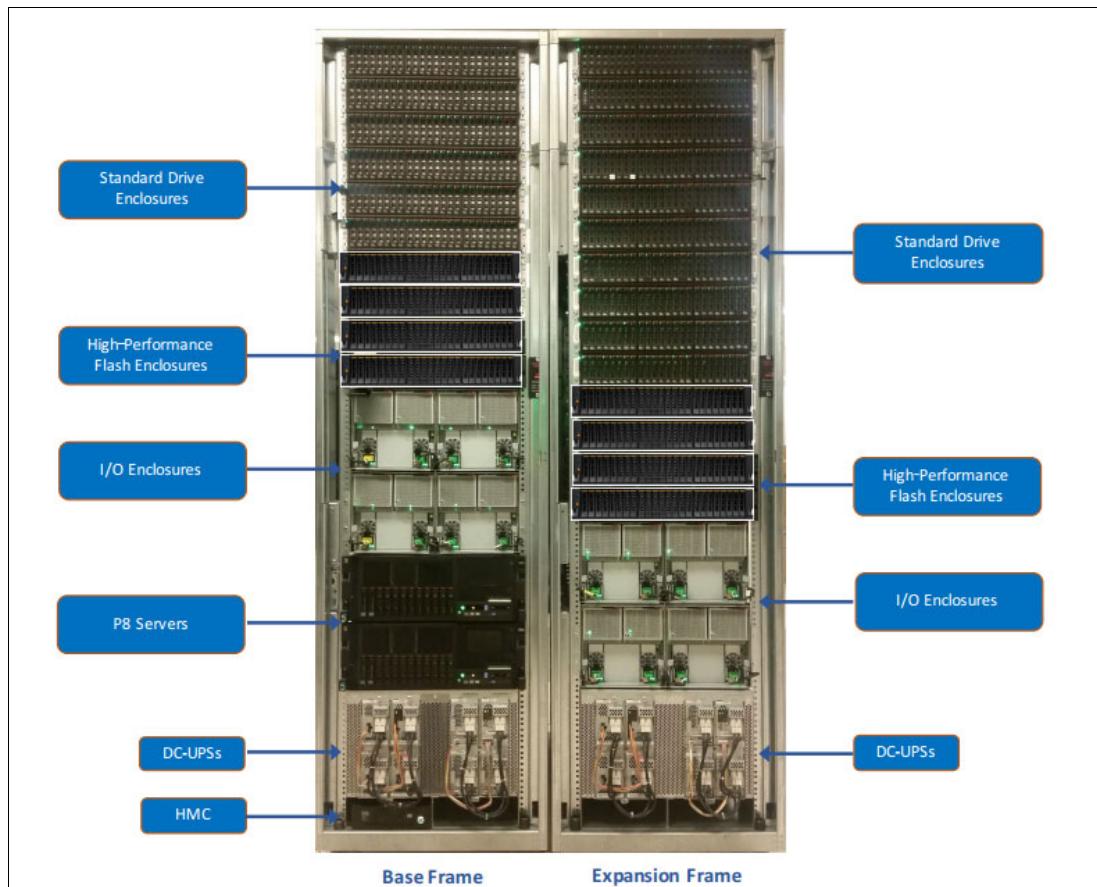


Figure 8-2 DS8886 front view, with one expansion frame shown

The major DS8000 components will be briefly described next. For an in-depth description of the DS8000 major components, see *IBM DS8880 Product Guide (Release 8.3)*, REDP-5344.

IBM POWER8 servers

A pair of POWER8 based servers, also known as central processor complexes, are at the heart of the IBM DS8880 models.

These servers share the load of receiving and moving data between the attached hosts and storage arrays. They can provide redundancy, so, if either server fails, the system operations are processed by the remaining server, without any disruption to the service.

Hardware Management Console

The mini PC HMC is a Linux-based server that enables users to interact with the DS8880 by using the HMC GUI (for service purposes) or DS Storage Manager/DS CLI (for storage administration or configuration).

Power subsystem

The power architecture used on DS8000 is based on a direct current uninterruptible power supply (DC-UPS). The power subsystems available depend on the DS8000 model in use. The DS8884 uses single-phase power, which supports 200-240V, whereas DS8886 also includes the option of having three-phase power supporting voltages of 220-240 V and 380-415 V. For DS8888, three-phase power is the standard.

Each DC-UPS has its own battery-backup functions. Therefore, the battery system also provides 2N redundancy. The battery of a single DC-UPS is able to preserve non-volatile storage (NVS) in a complete power outage.

High-performance flash enclosure

The high-performance flash enclosure (HPFE) Gen-2 is a 2U storage enclosure, with additional hardware at the rear (microbay). HPFEs need to be installed in pairs.

Each PHFE Gen-2 pair contains the following hardware components:

- ▶ 16, 32, 48 2.5-inch SAS Flash Cards.
- ▶ A pair of Microbays, where each Microbay contains these components, among others:
 - A Flash RAID adapter that is dual-core IBM PowerPC® based and can do RAID parity processing as well as provide speed and an amount of I/O that goes far beyond of what a usual Device Adapter could handle.
 - A PCIe switch card. The switch cards carry the signal forward from the Flash RAID adapters over through the PCIe Gen-3 into the processor complexes. The Flash RAID adapters go with eight SAS connections per Microbay pair to a pair of the specific flash enclosures that hold SAS expanders and the flash cards.

All pathing is redundant. The HPFE Microbays are directly attached to the I/O enclosures by using the PCIe bus fabric, which increases bandwidth and transaction-processing capability compared to Fibre Channel attached standard drives.

Standard drive enclosure

The IBM DS8880 offers 2U FC attached standard drive enclosures in two types. SFF drive enclosures are installed in pairs, and can contain up to 24 2.5-inch SFF SAS drives.

Large-Form factor (LFF) drive enclosures are also installed in pairs, and can contain up to 12 3.5-inch LFF SAS drives.

The SFF drives can be Flash Drives (SSDs) or rotational hard disk drives (HDDs), also known as disk drive modules (DDMs). Although Flash Drives and rotational drive types can both be used in the same storage system, they are not intermixed within the same standard drive enclosure pair.

Host adapters

The IBM DS8880 offers 16 Gbps Host Adapters (HAs) with four ports, and 8 Gbps HAs with either four or eight ports. Each HA port can be individually configured for FC or FICON connectivity. The 8 Gbps adapters also support FC-AL connections. Configuring multiple host connections across multiple HAs in multiple I/O enclosures provides the best combination of throughput and fault tolerance.

NVS cache

NVS is used to store a second copy of write data to ensure data integrity if there is a power failure or a cluster failure and the cache copy is lost. The NVS of cluster 1 is located in cluster 2 and the NVS of cluster 2 is located in cluster 1. In this way, during a cluster failure, the write data for the failed cluster will be in the NVS of the surviving cluster. This write data is then de-staged at high priority to the disk arrays. At the same time, the surviving cluster will start to use its own NVS for write data, ensuring that two copies of write data are still maintained. This process ensures that no data is lost even during a component failure.

Drive options

The IBM DS8880 offers five different drives types to meet the requirements of various workloads and configurations:

- ▶ 400 GB, 800 GB, 1.6 TB, and 3.2 TB flash cards for the highest performance requirements
- ▶ 400 GB, 800 GB, and 1.6 TB flash drives (SSDs) for higher performance requirements
- ▶ 300 GB, 600 GB, 15,000 RPM Enterprise drives for high-performance requirements
- ▶ 600 GB, 1.2 TB, and 1.8 TB 10,000 RPM drives for standard performance requirements
- ▶ 4 TB, and 6 TB 7,200 RPM Nearline drives for large-capacity requirements.

Additional drive types are in qualification. Flash cards and flash drives are treated as the same tier, and the IBM Easy Tier® intra-tier auto-rebalance function enables you to use the higher IOPS capability of flash cards.

For more information about DS8000 family major components, see *IBM DS8880 Product Guide (Release 8.3)*, REDP-5344.

8.4 IBM TotalStorage Resiliency Family

The IBM TotalStorage Resiliency Family is a set of products and features that are designed to help you implement storage solutions that keep your business running 24 hours a day, 7 days a week.

These hardware and software features, products, and services are available on the IBM DS8000 series. In addition, a number of advanced Copy Services features that are part of the IBM TotalStorage Resiliency family are available for the IBM DS6000™ and DS8000 series. The IBM TotalStorage DS Family also offers systems to support enterprise-class data backup and disaster recovery capabilities.

As part of the IBM TotalStorage Resiliency Family of software, IBM TotalStorage FlashCopy point-in-time copy capabilities back up data in the background and allow users nearly instant access to information about both source and target volumes. Metro and Global Mirror capabilities create duplicate copies of application data at remote sites. High-speed data transfers help to back up data for rapid retrieval.

8.4.1 Copy Services

Copy Services is a collection of functions that provides disaster recovery, data migration, and data duplication functions. Copy Services runs on the DS8000 series and supports open systems and zSeries environments.

Copy Services functions also are supported on the previous generation of storage systems, the IBM TotalStorage Enterprise Storage Server®.

Copy Services include the following types of functions:

- ▶ FlashCopy, which is a point-in-time copy function
- ▶ Remote mirror and copy functions (previously known as Peer-to-Peer Remote Copy or PPRC), which include the following:
 - IBM Metro Mirror (previously known as Synchronous PPRC)
 - IBM Global Copy (previously known as PPRC Extended Distance)
 - IBM Global Mirror (previously known as Asynchronous PPRC)
 - IBM Metro/Global Mirror
- ▶ z/OS Global Mirror (previously known as Extended Remote Copy or XRC)

8.4.2 FlashCopy

FlashCopy creates a copy of a source volume on the target volume. This copy is called a point-in-time copy. When you initiate a FlashCopy operation, a FlashCopy relationship is created between a source volume and target volume. A FlashCopy relationship is a “mapping” of the FlashCopy source volume and a FlashCopy target volume.

This mapping allows a point-in-time copy of that source volume to be copied to the associated target volume. The FlashCopy relationship exists between this volume pair from the time that you initiate a FlashCopy operation until the storage unit copies all data from the source volume to the target volume or you delete the FlashCopy relationship, if it is a persistent FlashCopy.

8.4.3 Metro Mirror function

Metro Mirror is a copy service that provides a continuous, synchronous mirror of one volume to a second volume. The different systems can be up to 300 kilometers apart, so by using Metro Mirror you can make a copy to a location off-site or across town. Because the mirror is updated in real time, no data is lost if a failure occurs. Metro Mirror is generally used for disaster-recovery purposes, where it is important to avoid data loss.

8.4.4 Global Copy function

Global Copy is a nonsynchronous mirroring function and is an alternative mirroring approach to Metro Mirror operations. Host updates to the source volume are not delayed by waiting for the update to be confirmed by a storage unit at your recovery site. The source volume sends a periodic, incremental copy of updated tracks to the target volume instead of a constant stream of updates.

There is no guarantee that dependent write operations are transferred in the same sequence that they have been applied to the source volume. This nonsynchronous operation results in a “fuzzy copy” at the recovery site. However, through operational procedures, you can create a point-in-time consistent copy at your recovery site that is suitable for data migration, backup, and disaster recovery purposes.

The Global Copy function can operate at very long distances, well beyond the 300 km distance that is supported for Metro Mirror, and with minimal impact to applications. The distance is limited only by the network and the channel extended technology.

8.4.5 Global Mirror function

Global Mirror is a copy service that is very similar to Metro Mirror. Both provide a continuous mirror of one volume to a second volume. However, with Global Mirror, the copy is asynchronous. You do not have to wait for the write to the secondary system to complete. For long distances, performance is improved compared to Metro Mirror. However, if a failure occurs, you might lose data. Global Mirror uses one of two methods to replicate data:

- ▶ Multicycling Global Mirror is designed to replicate data while adjusting for bandwidth constraints, and is appropriate for environments where it is acceptable to lose a few minutes of data if a failure occurs.
- ▶ For environments with higher bandwidth, noncycling Global Mirror can be used so that less than a second of data is lost if a failure occurs.

Global Mirror works well for data protection and migration when recovery sites are more than 300 kilometers away.

8.4.6 Metro/Global Mirror function

Metro/Global Mirror is a three-site, high availability disaster recovery solution that uses synchronous replication to mirror data between a local site and an intermediate site, and asynchronous replication to mirror data from an intermediate site to a remote site. The DS8000 series supports the Metro/Global Mirror function on open systems and IBM Z. You can set up and manage your Metro/Global Mirror configurations using DS CLI and TSO commands.

In a Metro/Global Mirror configuration, a Metro Mirror volume pair is established between two nearby sites (local and intermediate) to protect from local site disasters. The Global Mirror volumes can be located thousands of miles away and can be updated if the original local site has suffered a disaster but has performed failover operations to the intermediate site. In a local-site-only disaster, Metro/Global Mirror can provide zero-data-loss recovery at the remote site and the intermediate site.

8.4.7 z/OS Global Mirror function

z/OS Global Mirror (previously known as XRC) provides a long-distance remote copy solution across two sites for open systems and IBM Z data with asynchronous technology.

The DS8000 series supports the z/OS Global Mirror function on IBM Z hosts, only. The Global Mirror function mirrors data on the storage system to a remote location for disaster recovery. It protects data consistency across all volumes that you define for mirroring. The volumes can be on several different storage systems. The z/OS Global Mirror function can mirror the volumes over several thousand kilometers from the source site to the target recovery site.

With Global Mirror, you can suspend or resume service during an outage. You do not have to end your current data-copy session. You can suspend the session, and then restart it. Only data that changed during the outage must be synchronized again between the copies.

8.5 DS8000 performance features

This section covers DS8000 series performance features.

8.5.1 I/O priority queuing

Before I/O priority queuing, IOS kept the UCB I/O pending requests in a queue named IOSQ. The priority order of the I/O request in this queue, when the z/OS image is in goal mode, is controlled by Workload Manager (WLM), depending on the transaction owning the I/O request. There was no concept of priority queuing within the internal queues of the I/O control units. Instead, the queue regime was FIFO.

It is now possible to have this queue concept internally. I/O Priority Queuing in DS8000 has the following properties:

- ▶ I/O can be queued with the DS8000 in priority order.
- ▶ WLM sets the I/O priority when running in goal mode.
- ▶ There is I/O priority for systems in a sysplex.
- ▶ Each system gets a fair share.

8.5.2 Custom volumes

Custom volumes provide the possibility of defining DASD volumes with any size, from 1 - 268,434,453 cylinders, though current z/OS versions support DASD volumes with up to 1,182,006 cylinders. This capability gives storage administrators more flexibility to create the DASD volumes, and reduce wasted space, while reducing the number of UCBs necessary to provide the required capacity.

8.5.3 Improved caching algorithms

With its effective caching algorithms, DS8000 series can minimize wasted cache space and reduce disk drive utilization, reducing its back-end traffic. The current DS8886 has a maximum cache size of 2 TB, and the NVS size can be up to 2 TB.

The DS8000 can manage its cache in 4 KB segments, so for small data blocks (4 KB and 8 KB are common database block sizes), minimum cache is wasted. In contrast, large cache segments can exhaust cache capacity when filling up with small random reads.

This efficient cache management, together with the optional flash cards and flash drives, HPFEs, and caching size improvements, all integrate to give greater throughput while sustaining cache speed response times.

8.5.4 I/O Priority manager

I/O Priority Manager is a feature that provides application-level quality of service (QoS) for workloads that share a storage pool. This feature provides a way to manage QoS for I/O operations that are associated with critical workloads, and gives them priority over other I/O operations that are associated with non-critical workloads. For IBM z/OS, the I/O Priority Manager allows increased interaction with the host side.

8.5.5 Easy Tier

Easy Tier offers capabilities including manual volume capacity rebalance, auto performance rebalancing in both homogeneous and hybrid pools, hot-spot management, rank depopulation, manual volume migration, and thin provisioning support.

Easy Tier determines the appropriate tier of storage based on data access requirements, and then automatically and nondisruptively moves data, at the volume or extent level, to the appropriate tier in the storage system.

8.5.6 Thin provisioning

The DS8000 series allows you to create thin-provisioned volumes in your z/OS systems, maximizing your storage capacity, and allowing you to increase your storage capacity on demand with no disruption to z/OS systems that access the storage devices. To achieve that, a few features were introduced on DS8000 series. They are described in the next sections.

Small extents

In an alternative to the standard CKD extents of 1,113 cylinders, you can create volumes using small extents of 21 cylinders, allowing a better granularity, specially for thin-provisioned volumes.

Extent Space Efficient volumes

Extent Space Efficient (ESE) volumes are the replacement for the old Track Space Efficient (TSE) volumes, using allocations of 21 cylinders extents to back-end your data. These extents are allocated to the volumes from a defined extent pool when new data needs to be stored in the specified DASD. Space that is no longer used by the logical volume can be reclaimed by the storage administrator.

ESE volumes provide a better performance than TSE volumes.

8.6 Introduction to tape processing

The term *tape* refers to volumes that can be physically moved. You can only store sequential data sets on tape. Tape volumes can be sent to a safe, or to other data processing centers. Internal labels are used to identify magnetic tape volumes and the data sets on those volumes. You can process tape volumes with these types of labels:

- ▶ IBM standard labels
- ▶ Labels that follow standards published by these organizations:
 - International Organization for Standardization (ISO)
 - American National Standards Institute (ANSI)
- ▶ Nonstandard labels
- ▶ No labels

Note: Your installation can install a bypass for any type of label processing. However, the use of labels is preferred as a basis for efficient control of your data.

IBM standard tape labels consist of volume labels and groups of data set labels. The volume label, identifying the volume and its owner, is the first record on the tape. The data set label, identifying the data set and describing its contents, precedes and follows each data set on the volume:

- ▶ The data set labels that precede the data set are called *header* labels.
- ▶ The data set labels that follow the data set are called *trailer* labels. They are almost identical to the header labels.
- ▶ The data set label groups can include standard user labels at your option.

Usually, the formats of ISO and ANSI labels, which are defined by their respective organizations, are similar to the formats of IBM standard labels.

Nonstandard tape labels can have any format and are processed by routines that you provide. Unlabeled tapes contain only data sets and tape marks.

8.6.1 SL and NL format

Figure 8-3 illustrates the format differences between the following label types:

- ▶ No labels (NL)
- ▶ IBM standard labels (SL)

Other types are described in Table 8-1 on page 151.

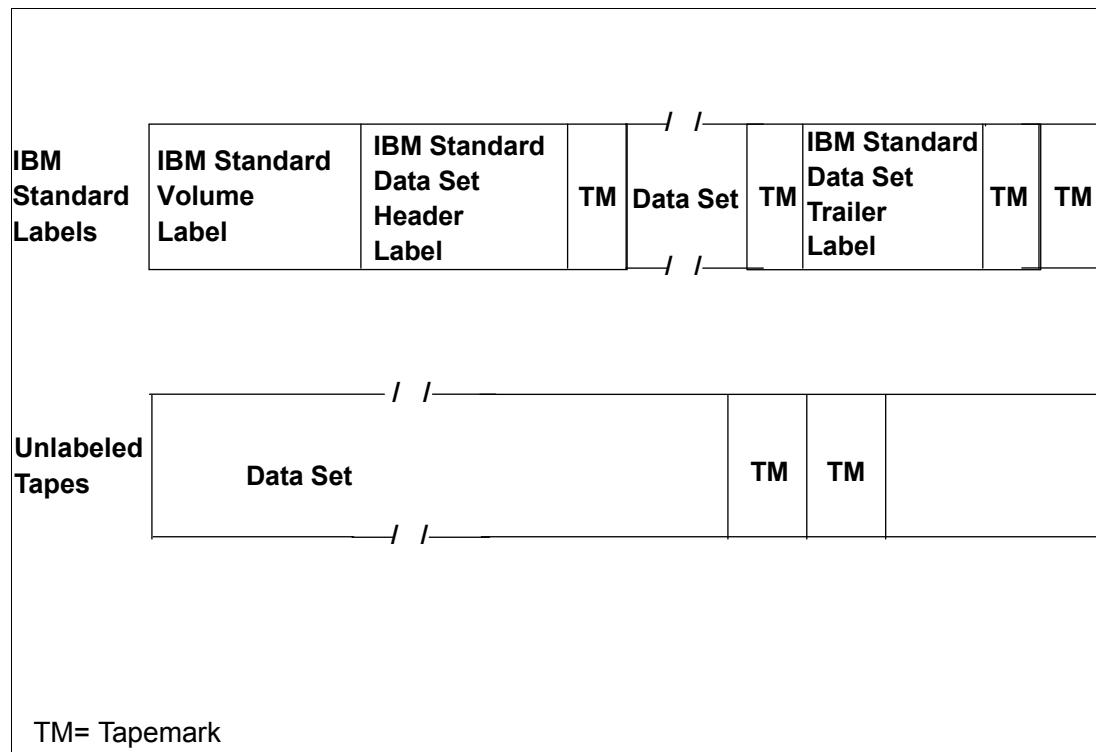


Figure 8-3 SL and NL format

Using tape with JCL

In the job control statements, you must provide a data definition (DD) statement for each data set to be processed. The LABEL parameter of the DD statement is used to describe the data set's labels.

Other parameters of the DD statement identify the data set, give volume and unit information and volume disposition, and describe the data set's physical attributes. You can use a data class to specify all of your data set's attributes (such as record length and record format), but not data set name and disposition. Specify the name of the data class using the job control language (JCL) keyword DATACLAS. If you do not specify a data class, the automatic class selection (ACS) routines assign a data class based on the defaults defined by your storage administrator.

An example of allocating a tape data set using DATACLAS in the DD statement of the JCL statements follows. In this example, TAPE01 is the name of the data class.

```
//NEW DD DSN=DATASET.NAME,UNIT=TAPE,DISP=(,CATLG,DELETE),DATACLAS=TAPE01,LABEL=(1,SL)
```

Describing the labels

You specify the type of labels by coding one of the subparameters of the LABEL parameter as shown in Table 8-1.

Table 8-1 Types of labels

Code	Meaning
SL	IBM Standard Label.
AL	ISO/ANSI/FIPS labels.
SUL	Both IBM and user header or trailer labels.
AUL	Both ISO/ANSI/FIPS and user header or trailer labels.
NSL	Nonstandard labels.
NL	No labels, but the existence of a previous label is verified.
BLP	Bypass label processing. The data is treated in the same manner as though NL had been specified, except that the system does not check for an existing volume label. The user is responsible for the positioning. If your installation does not allow BLP, the data is treated exactly as though NL had been specified. Your job can use BLP only if the Job Entry Subsystem (JES) through Job class, RACF through TAPEVOL class, or DFSMSrmm(*) allow it.
LTM	Bypass a leading tape mark. If encountered, on unlabeled tapes from VSE.

Note: If you do not specify the label type, the operating system assumes that the data set has IBM standard labels.

8.6.2 Tape cartridges

The operating system supports several IBM magnetic tape subsystems, such as the IBM 3590 and 3592 Tape Subsystem, as well as virtual tape, all of which use tape cartridges.

Tape mount management

Using DFSMS and tape mount management can help you reduce the number of both tape mounts and tape volumes that your installation requires. The volume mount analyzer reviews your tape mounts and creates reports that provide you with the information that you need to effectively implement the tape mount management methodology suggested by IBM.

Tape mount management allows you to efficiently fill a tape cartridge to its capacity and gain full benefit from compression. By filling your tape cartridges, you reduce your tape mounts and even the number of tape volumes that you need.

With an effective tape cartridge capacity of 15 TB using 3592 JD cartridges on TS1155 tape drives, DFSMS can intercept all but extremely large data sets and manage them with tape mount management. By implementing tape mount management with DFSMS, you can reduce your tape mounts with little or no additional hardware required.

Tape mount management also improves job throughput because jobs are no longer queued up on tape drives. A large portion of all tape data sets queued up on drives are less than 20 MB. With tape mount management, these data sets are on DASD while in use. This feature frees up the tape drives for other allocations.

Tape mount management recommends that you use DFMSHsm to do interval migration to SMS storage groups. You can use ACS routines to redirect your tape data sets to a tape mount management DASD buffer storage group. DFMSHsm scans this buffer regularly and migrates the data sets to migration level 1 DASD or migration level 2 tape as soon as possible, based on the management class and storage group specifications.

8.7 IBM TS1155 tape drive

The IBM TS1155 tape drive is the sixth generation of high capacity and high-performance tape systems. It was announced in May 2017 and connects to IBM Z through Fibre Channel links. The 3592 system is the successor of the IBM Magstar® 3590 family of tape drives and controller types.

The IBM 3592 tape drive can be used as a stand-alone solution or as an automated solution within a TS4500 tape library.

The native rate for data transfer increases up to 360 MBps. The uncompressed amount of data that fits on a single cartridge increases to 15 TB and is used for scenarios where high capacity is needed. The tape drive has a second option, where you can store a maximum of 10 TB per tape. This option is used whenever fast access to tape data is needed.

Unlike other tape drives, TS1155 provides two different methods of connecting to the host. The first option is using 8-Gbps Fibre Channel interface, allowing connections to the host or switched fabric environment. An alternative model (TS1155 model 55F) has a dual-ported 10 Gb Ethernet port for host attachment, which is optimized for cloud-based and large, open computer environments.

8.8 IBM TS4500 tape library

Tape storage media can provide low-cost data storage for sequential files, inactive data, and vital records. Because of the continued growth in tape use, *tape automation* has been seen as a way of addressing an increasing number of challenges.

The IBM TS4500 offers a wide range of features that include the following:

- ▶ Up to 128 tape drives
- ▶ Support through the Library Control Unit for attachment of up to 17 additional frames
- ▶ Cartridge storage capacity of over 17,500 3592 tape cartridges
- ▶ Data storage capacity of up to 263 PB

- ▶ Support for the High Availability unit that provides a high level of availability for tape automation
- ▶ Support for the IBM TS7700 through Fibre Channel switches
- ▶ Support for the IBM Total Storage Peer-to-Peer VTS
- ▶ Support for the following tape drives:
 - IBM TS1155 Model 55F tape drive
 - IBM TS1155 Model 55E tape drive
 - IBM TS1150 Model E08 tape drives
 - IBM TS1140 Model E07 tape drives
 - IBM Linear Tape-Open (LTO) Ultrium 7 tape drives
- ▶ Attachment to and sharing by multiple host systems, such as IBM Z, iSeries, pSeries, AS/400, HP, and Sun processors
- ▶ Data paths through Fibre Channels
- ▶ Library management commands through RS-232, a local area network (LAN), and parallel, ESCON, and FICON channels

8.9 Introduction to IBM TS7700

The IBM TS7700 series, integrated with the IBM TS4500, delivers an increased level of storage capability beyond the traditional storage products hierarchy. The TS7700 emulates the function and operation of IBM 3490 Enhanced Capacity (3490E). This virtualization of both the tape devices and the storage media to the host allows for transparent utilization of the capabilities of the IBM 3592 tape technology.

With virtualization, a new view of virtual volumes and drives was introduced, as users and applications using TS7700 do not access directly the physical volumes and drives associated with it. Instead, they use virtual drives and volumes that are managed by the library. Using a TS7700 subsystem, the host application writes tape data to virtual devices. The volumes created by the hosts are called *virtual volumes*. They are physically stored in a tape volume cache that is built from RAID DASD, being destaged to physical tapes when attached to a TS4500, and the library cache reaches the defined threshold.

8.9.1 IBM TS7700 models

Two TS7700 models are currently available:

- ▶ IBM TS7760

The IBM TS7760 is an all new hardware refresh and features Encryption Capable, high-capacity cache that uses 4 TB serial-attached Small Computer System Interface (SCSI) HDDs in arrays that use dynamic disk pool configuration.

- ▶ IBM TS7760T

This option is similar to IBM TS7760, and also has a physical tape library attached to store additional copies or add capacity.

Each FICON channel in the TS7700 can support up to 512 logical paths, providing up to 4,096 logical paths with eight FICON channels. You can also have up to 496 virtual drives defined.

Tape volume cache

The IBM TS7700 appears to the host processor as a single automated tape library with up to 496 virtual tape drives and up to 4,000,000 virtual volumes. The configuration of this system has up to 1.3 PB of Tape Volume Cache native, and up to 16 IBM 3592 tape drives.

Through tape volume cache management policies, the TS7700 management software moves host-created volumes from the tape volume cache to a cartridge managed by the TS7700 subsystem. When a virtual volume is moved from the tape volume cache to tape, it becomes a logical volume.

VTS functions

VTS provides the following functions:

- ▶ Thirty-two 3490E virtual devices.
- ▶ Tape volume cache (implemented in a RAID-5 disk) that contains virtual volumes.

The tape volume cache consists of a high-performance array of DASD and storage management software. Virtual volumes are held in the tape volume cache when they are being used by the host system. Outboard storage management software manages which virtual volumes are in the tape volume cache and the movement of data between the tape volume cache and physical devices.

The size of the DASD is made large enough so that more virtual volumes can be retained in it than just the ones that are currently associated with the virtual devices.

After an application modifies and closes a virtual volume, the storage management software in the system makes a copy of it onto a physical tape. The virtual volume remains available on the DASD until the space it occupies reaches a predetermined threshold. Leaving the virtual volume in the DASD allows for fast access to it during subsequent requests.

The DASD and the management of the space used to keep closed volumes available is called *tape volume cache*. Performance for mounting a volume that is in tape volume cache is quicker than if a real physical volume is mounted.

- ▶ Up to sixteen 3592 tape drives. The real 3592 volume contains logical volumes.
- ▶ Stacked 3592 tape volumes managed by the TS4500. It fills the tape cartridge up to 100%. Putting multiple virtual volumes into a stacked volume, TS7700 uses all of the available space on the cartridge. TS7700 uses IBM 3592 cartridges when stacking volumes.

8.10 Introduction to TS7700 Grid configuration

A grid configuration is specifically designed to enhance data availability. It accomplishes this by providing volume copy, remote functionality, and automatic recovery and switchover capabilities. With a design that reduces single points of failure (including the physical media where logical volumes are stored), the grids improve system reliability and availability, as well as data access. A grid configuration can have from two to six clusters that are connected using a grid link. Each grid link is connected to other grid links on a grid network.

Logical volumes that are created within a grid can be selectively replicated to one or more peer clusters by using a selection of different replication policies. Each replication policy or Copy Consistency Point provides different benefits, and can be intermixed. The grid architecture also enables any volume that is located within any cluster to be accessed remotely, which enables ease of access to content anywhere in the grid.

In general, any data that is initially created or replicated between clusters is accessible through any available cluster in a grid configuration. This concept ensures that data can still be accessed even if a cluster becomes unavailable. In addition, it can reduce the need to have copies in all clusters because the adjacent or remote cluster's content is equally accessible.

8.10.1 Copy consistency

There are currently five available consistency point settings:

Sync	As data is written to the volume, it is compressed and then simultaneously written or duplexed to two TS7700 locations. The mount point cluster is not required to be one of the two locations. Memory buffering is used to improve the performance of writing to two locations. Any pending data that is buffered in memory is hardened to persistent storage at both locations only when an implicit or explicit sync operation occurs. This setting provides a zero RPO at tape sync point granularity.
	Tape workloads in IBM Z environments already assume sync point hardening through explicit sync requests or during close processing, enabling this mode of replication to be performance-friendly in a tape workload environment. When sync is used, two clusters must be defined as sync points. All other clusters can be any of the remaining consistency point options, enabling more copies to be made.
RUN	The copy occurs as part of the Rewind Unload (RUN) operation, and completes before the RUN operation at the host finishes. This mode is comparable to the immediate copy mode of the PtP VTS.
Deferred	The copy occurs after the rewind unload operation at the host. This mode is comparable to the Deferred copy mode of the PtP VTS. This is also called asynchronous replication.
Time Delayed	The copy occurs only after a specified time (1 hour - 379 days). If the data expires before the Time Delayed setting is reached, no copy is produced at all. For Time Delayed, you can specify after creation or after access in the MC.
No Copy	No copy is made.

8.10.2 VTS advanced functions

As with a stand-alone IBM TS7700, the grid configuration has the option to install additional features and enhancements to existing features, including the following:

- ▶ Outboard policy management: Enables you to better manage your IBM TS7700 stacked and logical volumes. With this support, the SMS construct names that are associated with a volume (storage class, storage group, management class, and data class) are sent to the library. At the library, you can define outboard policy actions for each construct name, enabling you and the TS7700 to better manage your volumes.

For example, through the storage group policy and physical volume pooling, you now can group logical volumes with common characteristics on a set of physical stacked volumes.

- ▶ Physical volume pooling: With outboard policy management enabled, you can assign logical volumes to selected storage groups. Storage groups point to primary storage pools. These pool assignments are stored in the library manager database. When a logical volume is copied to tape, it is written to a stacked volume that is assigned to a storage pool as defined by the storage group constructs at the library manager.
- ▶ Tape volume dual copy: With advanced policy management, storage administrators have the facility to selectively create dual copies of logical volumes within a TS7700. This function is also available in the grid environment. At the site or location where the second distributed library is located, logical volumes can also be duplexed, in which case you can have two or four copies of your data.
- ▶ Tape volume cache management: Before the introduction of these features, there was no way to influence cache residency. As a result, all data written to the tape volume cache (TVC) was pre-migrated using a first-in, first-out (FIFO) method. With the introduction of this function, you can now influence the time that virtual volumes reside in the TVC.

8.11 Storage area network

The Storage Network Industry Association (SNIA) defines a SAN as a network whose primary purpose is the transfer of data between computer systems and storage elements. A SAN consists of a communication infrastructure, which provides physical connections, and a management layer, which organizes the connections, storage elements, and computer systems so that data transfer is secure and robust.

The term SAN is usually (but not necessarily) identified with block I/O services rather than file access services. It can also be a storage system that consists of storage elements, storage devices, computer systems, and appliances, plus all control software, communicating over a network.

SANs today are usually built by using Fibre Channel technology, but the concept of a SAN is independent of the underlying type of network.

The following are the major potential benefits of a SAN:

- ▶ Access
Benefits include longer distances between processors and storage, higher availability, and improved performance (because I/O traffic is offloaded from a LAN to a dedicated network, and because Fibre Channel is generally faster than most LAN media). Also, a larger number of processors can be connected to the same storage device, compared to typical built-in device attachment facilities.
- ▶ Consolidation
Another benefit is replacement of multiple independent storage devices by fewer devices that support capacity sharing, which is also called *disk and tape pooling*. SANs provide the ultimate in scalability because software can allow multiple SAN devices to appear as a single pool of storage accessible to all processors on the SAN. Storage on a SAN can be managed from a single point of control. Controls over which hosts can see which storage (called *zoning* and *LUN masking*) can be implemented.
- ▶ Protection
LAN-free backups occur over the SAN rather than the (slower) LAN, and server-free backups can let disk storage “write itself” directly to tape without processor overhead.

There are various SAN topologies on the base of Fibre Channel networks:

- ▶ Point-to-Point

With a SAN, a simple link is used to provide high-speed interconnection between two nodes.

- ▶ Arbitrated loop

The Fibre Channel arbitrated loop offers relatively high bandwidth and connectivity at a low cost. For a node to transfer data, it must first arbitrate to win control of the loop. After the node has control, it is free to establish a virtual point-to-point connection with another node on the loop. After this point-to-point (virtual) connection is established, the two nodes consume all of the loop's bandwidth until the data transfer operation is complete. After the transfer is complete, any node on the loop can then arbitrate to win control of the loop.

- ▶ Switched

Fibre Channel switches function in a manner similar to traditional network switches to provide increased bandwidth, scalable performance, an increased number of devices, and, in certain cases, increased redundancy.

Multiple switches can be connected to form a switch fabric capable of supporting many host servers and storage subsystems. When switches are connected, each switch's configuration information must be copied into all the other participating switches. This process is called *cascading*.

Figure 8-4 shows a sample SAN configuration.

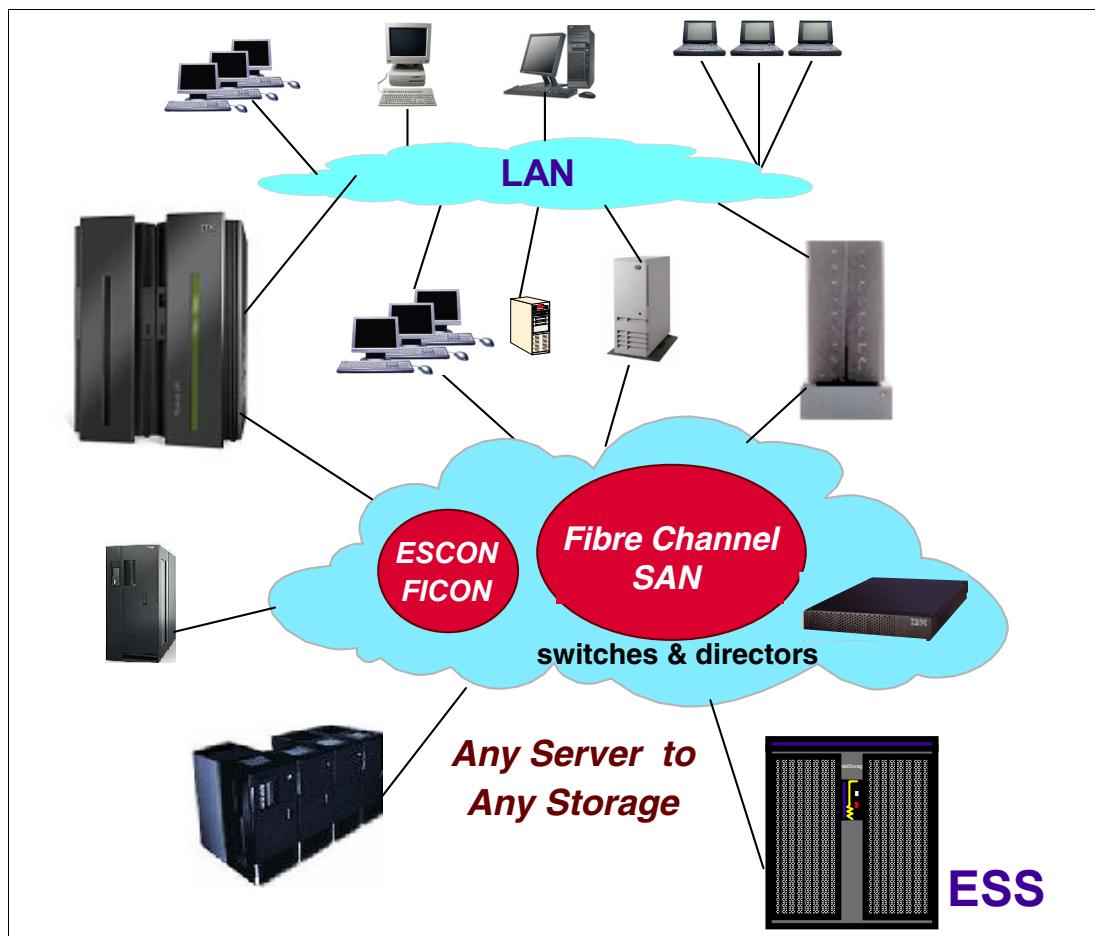


Figure 8-4 Sample SAN configuration

8.11.1 FICON and SAN

From an IBM Z perspective, FICON is the protocol that is used in a SAN environment. A FICON infrastructure can be point-to-point or switched, using FICON directors to provide connections between channels and control units. FICON uses Fibre Channel transport protocols, and so uses the same physical fiber. Today, IBM Z has 16 Gbps link data rate support.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 160. Note that some of the documents referenced here might be available in softcopy only.

- ▶ *DFSMStvs Overview and Planning Guide*, SG24-6971
- ▶ *DFSMStvs Presentation Guide*, SG24-6973
- ▶ *IBM z/OS V2.1 DFSMS Technical Update*, SG24-8190
- ▶ *VSAM Demystified*, SG24-6105

Other publications

These publications are also relevant as further information sources:

- ▶ *Device Support Facilities (ICKDSF) User's Guide and Reference*, GC35-0033-35
- ▶ *Device Support Facilities User's Guide and Reference Release 17*, GC35-0033
- ▶ *DFSMS Optimizer User's Guide and Reference*, SC26-7047
- ▶ *DFSORT Getting Started with DFSORT R14*, SC26-4109
- ▶ *DFSORT Installation and Customization Release 14*, SC33-4034
- ▶ *Tivoli Decision Support for OS/390 System Performance Feature Reference Volume I*, SH19-6819
- ▶ *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394
- ▶ *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407
- ▶ *z/OS DFSMS: Managing Catalogs*, SC26-7409
- ▶ *z/OS DFSMS Object Access Method Application Programmer's Reference*, SC35-0425
- ▶ *z/OS DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Object Support*, SC35-0426
- ▶ *z/OS DFSMS: Using Data Sets*, SC26-7410
- ▶ *z/OS DFSMS: Using the Interactive Storage Management Facility*, SC26-7411
- ▶ *z/OS DFSMS: Using Magnetic Tapes*, SC26-7412
- ▶ *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402
- ▶ *z/OS DFSMSdfp Utilities*, SC26-7414
- ▶ *z/OS DFSMSdss Storage Administration Guide*, SC35-0423
- ▶ *z/OS DFSMSdss Storage Administration Reference*, SC35-0424
- ▶ *z/OS DFSMShsm Storage Administration Guide*, SC35-0421

- ▶ *z/OS DFSMShsm Storage Administration Reference*, SC35-0422
- ▶ *z/OS DFSMSrmm Guide and Reference*, SC26-7404
- ▶ *z/OS DFSMSrmm Implementation and Customization Guide*, SC26-7405
- ▶ *z/OS DFSMStvs Administration Guide*, GC26-7483
- ▶ *z/OS DFSMStvs Planning and Operating Guide*, SC26-7348
- ▶ *z/OS DFSORT Application Programming Guide*, SC26-7523
- ▶ *z/OS MVS JCL Reference*, SA22-7597
- ▶ *z/OS MVS Programming: Assembler Services Guide*, SA22-7605
- ▶ *z/OS MVS System Commands*, SA22-7627
- ▶ *z/OS MVS System Messages, Volume 1 (ABA-AOM)*, SA22-7631
- ▶ *z/OS Network File System Guide and Reference*, SC26-7417

Online resources

These websites and URLs are also relevant as further information sources:

- ▶ For articles, online books, news, tips, techniques, examples, and more, visit the IBM Data storage home page:
<https://www.ibm.com/storage>

How to get IBM Redbooks publications

You can search for, view, or download books, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy books or CD-ROMs, at this website:

[ibm.com/redbooks](https://www.ibm.com/redbooks)

Help from IBM

IBM Support and downloads

[ibm.com/support](https://www.ibm.com/support)

IBM Global Services

[ibm.com/services](https://www.ibm.com/services)



ABCs of IBM z/OS System Programming Volume 3

(0.2"spine)
0.17" <-> 0.473"
90<->249 pages



SG24-6983-04

ISBN 0738442801

Printed in U.S.A.

Get connected

