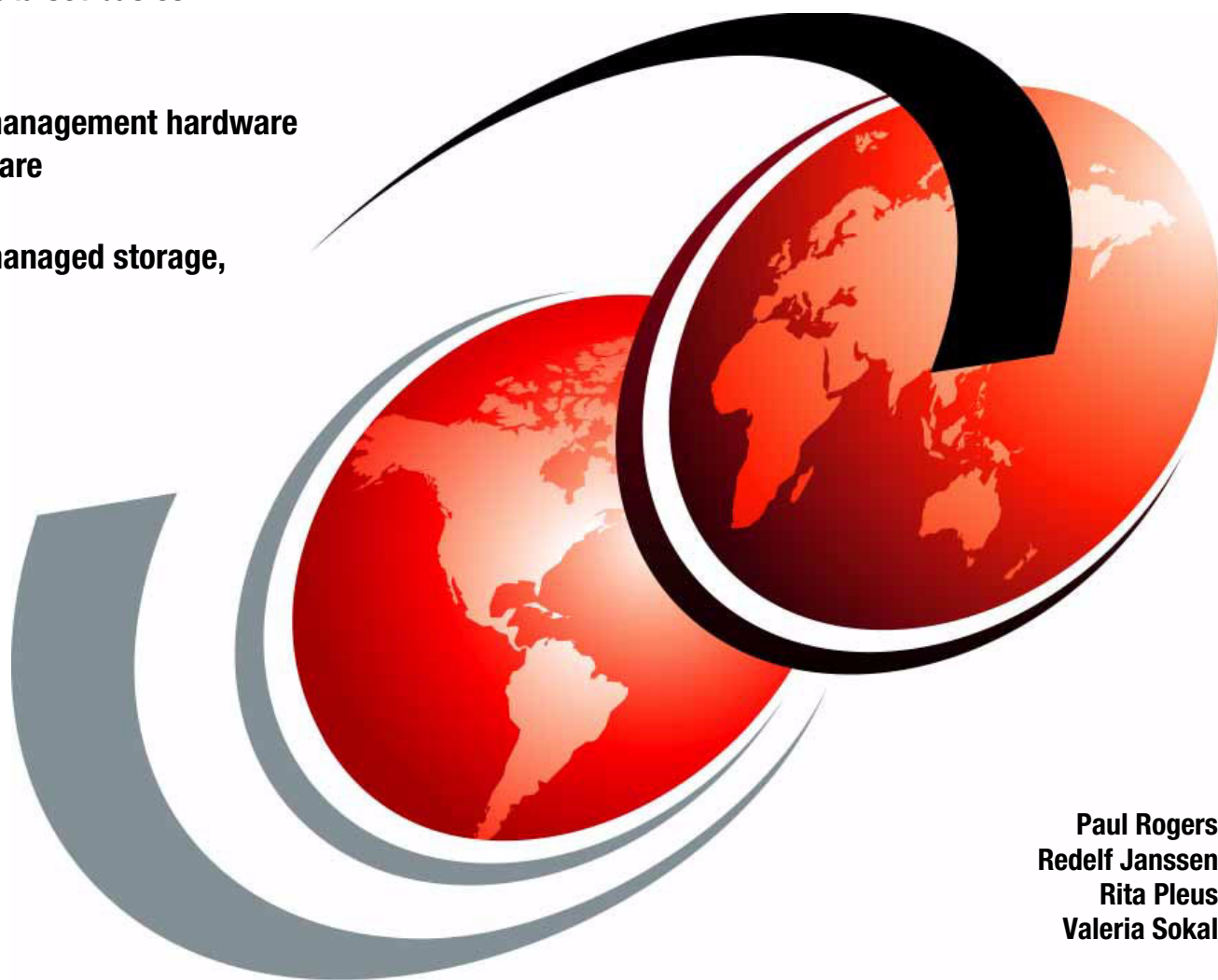


# ABCs of z/OS System Programming Volume 3

DFSMS, data set basics

Storage management hardware  
and software

System-managed storage,  
ISMF



Paul Rogers  
Redelf Janssen  
Rita Pleus  
Valeria Sokal

**Redbooks**





International Technical Support Organization

**ABCs of z/OS System Programming Volume 3**

November 2004

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

**First Edition (November 2004)**

This edition applies to Version 1, Release 4, of z/OS™ (5694-A01), to Version 1, Release 4, of z/OS.e™ (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

**© Copyright International Business Machines Corporation 2004. All rights reserved.**

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



# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
The team that wrote this redbook .....	ix
Become a published author .....	x
Comments welcome .....	x
<b>Chapter 1. DFSMS introduction</b> .....	1
1.1 Introduction to DFSMS .....	3
1.2 Data Facility Storage Management Subsystem .....	4
1.3 DFSMSdfp component .....	5
1.4 DFSMSdss component .....	7
1.5 DFSMSrmm component .....	9
1.6 DFSMSHsm component .....	11
1.7 DFSMStvs component .....	13
<b>Chapter 2. Data set basics</b> .....	15
2.1 Data sets .....	17
2.2 Data set name rules .....	18
2.3 DFSMSdfp data set organizations .....	19
2.4 Types of VSAM data sets .....	21
2.5 Non-VSAM data sets .....	22
2.6 Extended-format data sets and objects .....	23
2.7 z/OS UNIX files .....	25
2.8 Data set organization (DSORG) .....	27
2.9 Allocate a data set with ISPF 3.2 .....	29
2.10 Logical record length (LRECL) .....	30
2.11 Locating a data set .....	32
2.12 Uncataloged and cataloged data sets .....	34
2.13 Volume table of contents (VTOC) .....	35
2.14 Data set control block (DSCB) .....	36
2.15 VTOC index structure .....	38
2.16 Initializing a volume (ICKDSF) .....	40
2.17 Problem determination .....	43
<b>Chapter 3. Storage management hardware</b> .....	45
3.1 Overview of DASD types .....	46
3.2 Traditional DASD capacity .....	48
3.3 Large Volume Support .....	49
3.4 Redundant array of independent disks (RAID) .....	51
3.5 Seascape architecture .....	53
3.6 Enterprise Storage Server (ESS) .....	56
3.7 ESS universal access .....	58
3.8 Operating systems supporting ESS .....	59
3.9 ESS major components .....	60
3.10 ESS host adapters .....	61
3.11 FICON host adapters .....	62
3.12 ESS disks .....	64

3.13	Device adapters	66
3.14	SSA loops	68
3.15	RAID-10	70
3.16	Storage balancing with RAID-10	72
3.17	ESS performance features	73
3.18	WLM controlling PAVs	75
3.19	ESS copy services	77
3.20	TotalStorage Expert product highlights	80
3.21	Introduction to tape processing	82
3.22	SL and NL format	84
3.23	Tape capacity - tape mount management	86
3.24	TotalStorage Enterprise Tape Drive 3592 Model J1A	88
3.25	IBM TotalStorage Enterprise Automated Tape Library 3494	90
3.26	Introduction to Virtual Tape Server (VTS)	92
3.27	IBM TotalStorage Peer-to-Peer VTS	94
3.28	Storage area network (SAN)	96
<b>Chapter 4. Storage management software</b>		<b>99</b>
4.1	Overview of DFSMSdfp utilities	100
4.2	IEBCOMPR (compare data set) program	102
4.3	Comparing data sets	104
4.4	IEBCOPY utility	105
4.5	IEBCOPY copy operation	107
4.6	IEBCOPY compress operation	109
4.7	IEBGENER	110
4.8	Adding members to a PDS using IEBGENER	111
4.9	Copying data to tape	112
4.10	IEHLIST	113
4.11	IEHLIST LISTVTOC output	114
4.12	IEHINITT	115
4.13	IEFBR14	117
4.14	Access method services	118
4.15	AMS functional commands	121
4.16	AMS modal commands	123
4.17	Data Collection Facility (DCOLLECT)	124
4.18	Generation data groups (GDG)	125
4.19	Defining a generation data group	127
4.20	Absolute generation and version numbers	129
4.21	Relative generation numbers	131
4.22	Access method	133
4.23	Major DFSMS access methods	134
4.24	Basic Partitioned Access Method (BPAM)	135
4.25	PDS data organization	136
4.26	Partitioned data set extended (PDSE)	138
4.27	Sequential access methods	140
4.28	Virtual Storage Access Method (VSAM)	142
4.29	VSAM resource pool and buffering techniques	143
4.30	System-managed buffering (SMB)	145
4.31	VSAM terminology and concepts	147
4.32	Control interval (CI)	148
4.33	VSAM data set components	150
4.34	Key sequenced data set (KSDS)	152
4.35	Processing a KSDS data set	153

4.36	Relative record data set (RRDS)	155
4.37	Typical RRDS processing	156
4.38	Linear data set (LDS)	157
4.39	Data-in-virtual	158
4.40	Data-in-virtual objects	159
4.41	Mapping a linear data set	160
4.42	Entry sequenced data set (ESDS)	161
4.43	Typical ESDS processing	162
4.44	DFSORT	163
4.45	DFSMS Network File System	165
4.46	DFSMS Optimizer	167
4.47	DFSMSdss	168
4.48	DFSMSdss: physical and logical processing	169
4.49	DFSMSdss: logical processing	170
4.50	DFSMSdss: physical processing	172
4.51	DFSMSdss stand-alone services	174
4.52	DFSMSshm	175
4.53	Availability management	176
4.54	Space management	177
4.55	Storage device hierarchy	179
4.56	HSM volume types	180
4.57	Automatic space management	182
4.58	Recall	183
4.59	Removable media manager (DFSMSrmm)	184
4.60	Libraries and locations	185
4.61	What DFSMSrmm can manage	186
4.62	Managing libraries and storage locations	188
<b>Chapter 5. System-managed storage</b>		<b>189</b>
5.1	Storage management	190
5.2	DFSMS and DFSMS environment	191
5.3	Benefits of system-managed storage	192
5.4	Establishing service level objectives	195
5.5	Implementing SMS policies	197
5.6	Monitoring SMS policies	199
5.7	Assigning data to be system-managed	200
5.8	Using data classes	201
5.9	Using storage classes	203
5.10	Using management classes	205
5.11	Management class functions	207
5.12	Using storage groups	208
5.13	Using aggregate backup and recovery support (ABARS)	210
5.14	Automatic Class Selection (ACS) routines	212
5.15	SMS configuration	214
5.16	Implementing DFSMS	216
5.17	Steps to activate a minimal SMS configuration	217
5.18	Allocating SMS control data sets	218
5.19	Defining the SMS base configuration	220
5.20	Creating ACS routines	223
5.21	DFSMS setup for z/OS	225
5.22	Starting SMS	227
5.23	Control SMS processing with operator commands	229
5.24	Displaying the SMS configuration	231

5.25	Managing data with minimal SMS configuration . . . . .	232
5.26	Device-independence space allocation. . . . .	234
5.27	Developing naming conventions . . . . .	236
5.28	Setting the low-level qualifier (LLQ) standards . . . . .	238
5.29	Establishing installation standards . . . . .	240
5.30	Planning and defining data classes. . . . .	241
5.31	Data class attributes . . . . .	242
5.32	Use data class ACS routine to enforce standards . . . . .	243
5.33	Simplifying JCL use. . . . .	244
5.34	Allocating a data set . . . . .	245
5.35	Creating a VSAM cluster. . . . .	247
5.36	Space allocation for a VSAM KSDS cluster . . . . .	248
5.37	Retention period and expiration date . . . . .	249
5.38	SMS PDSE support. . . . .	250
5.39	PDSE conversion . . . . .	252
5.40	DFSMS and program objects . . . . .	254
5.41	Selecting data sets to allocate as PDSEs. . . . .	257
5.42	Allocating new PDSEs . . . . .	258
5.43	Identifying PDSEs . . . . .	259
5.44	System-managed data types . . . . .	260
5.45	Data types that cannot be system-managed. . . . .	262
5.46	Introduction to ISMF . . . . .	264
5.47	ISMF product relationships . . . . .	265
5.48	What you can do with ISMF . . . . .	267
5.49	Accessing ISMF . . . . .	269
5.50	ISMF Profile option . . . . .	270
5.51	Navigating through ISMF . . . . .	271
5.52	Obtaining information about a panel field . . . . .	272
5.53	Data Set option . . . . .	273
5.54	Obtaining a data set list. . . . .	274
5.55	Volume Option . . . . .	275
5.56	Obtaining a volume list . . . . .	276
5.57	Management Class option . . . . .	277
5.58	Management Class List panel. . . . .	278
5.59	Data Class option . . . . .	279
5.60	Obtaining a data class listing . . . . .	280
5.61	Displaying data class information . . . . .	281
5.62	Storage Class option. . . . .	282
5.63	Storage Class List panel. . . . .	283
5.64	List option . . . . .	284
5.65	Removable Media Manager option . . . . .	285
	<b>Related publications . . . . .</b>	<b>287</b>
	IBM Redbooks . . . . .	287
	Other publications . . . . .	287
	How to get IBM Redbooks . . . . .	288
	Help from IBM . . . . .	288

# Notices

This information was developed for products and services offered in the U.S.A.

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 give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.*

*The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:* 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 states 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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.


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 the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates 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. 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 IBM's application programming interfaces.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®	FICON®	RAMAC®
@server®	FlashCopy®	Redbooks™
AIX®	IBM®	Redbooks (logo)™
AS/400®	IMS™	Redbooks (logo)  ™
Common User Access®	Infoprint®	RMF™
CICS®	Infospeed®	RS/6000®
CICS/ESA®	iSeries™	S/390®
CUA®	Language Environment®	Seascape®
DB2®	Magstar®	Sequent®
DFS™	MVS™	Tivoli®
DFSMSdfp™	MVS/DFP™	TotalStorage®
DFSMSdss™	OS/2®	VTAM®
DFSMSHsm™	OS/390®	z/Architecture™
DFSMSrmm™	OS/400®	z/OS®
DFSORT™	Parallel Sysplex®	zSeries®
Enterprise Storage Server®	pSeries®	
ESCON®	RACF®	

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

# Preface

The ABCs of z/OS® System Programming is an eleven-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. If you would like to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection will serve as a powerful technical tool.

Volume 3 provides an introduction to DFSMS, storage management hardware and software, system-managed storage, and ISMF.

The contents of the other volumes are:

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, JES2 and JES3, LPA, LNKLST, authorized libraries, and catalogs

Volume 4: Communication Server, TCP/IP and VTAM®

Volume 5: Base and Parallel Sysplex®, system logger, global resource serialization, z/OS system operations, and automatic restart management

Volume 6: RACF®, PKI, LDAP, cryptography, Kerberos and firewall technologies

Volume 7: Infoprint® Server, Language Environment®, and SMP/E

Volume 8: z/OS problem diagnosis

Volume 9: z/OS UNIX® System Services (USS)

Volume 10: Introduction to z/Architecture™, zSeries® processor design, zSeries connectivity, LPAR concepts, and HCD.

Volume 11: WLM, RMF™, and performance management

## The team that wrote this redbook

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

**Paul Rogers** is a Consulting IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on various aspects of z/OS JES3, and z/OS UNIX. Before joining the ITSO 16 years ago, Paul worked in the IBM Installation Support Center (ISC) in Greenford, England, providing OS/390® and JES support for IBM EMEA and the Washington Systems Center in Gaithersburg, Maryland.

**Redelf Janßen** is an IT Architect in IBM Global Services ITS in IBM Germany. He holds a degree in Computer Science from University of Bremen and joined IBM Germany in 1988. His

areas of expertise include IBM zSeries, z/OS and availability management. He has written Redbooks™ on OS/390 Releases 3, 4, and 10.

**Rita Pleus** is an IT Architect in IBM Global Services ITS in IBM Germany. She has 18 years of IT experience in a variety of areas, including systems programming and operations management. Before joining IBM in 2001, she worked for a German S/390® customer. Rita holds a degree in Computer Science from the University of Applied Sciences in Dortmund. Her areas of expertise include z/OS, its subsystems, and systems management.

**Valeria Sokal** is an MVS™ system programmer at an IBM customer. She has 15 years of experience as a mainframe systems programmer.

## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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

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

## Comments welcome

Your comments are important to us!

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

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

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

- ▶ Send your comments in an email to:

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

- ▶ Mail your comments to:

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





# 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 comprises a suite of related data and storage management products for the z/OS system. DFSMS is now an integral element of the z/OS operating system.

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.

DFSMS is an exclusive element of the z/OS operating system and is a software suite that automatically manages data from creation to expiration. The following elements comprise DFSMS:

- ▶ DFSMSdfp™, which 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 DFSMSstvs—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 following optional features enabled, you can take full advantage of all the functions that DFSMS offers:

- ▶ DFSMSdss, an optional feature of z/OS
- ▶ DFSMShsm, an optional feature of z/OS
- ▶ DFSMSrmm, an optional feature of z/OS
- ▶ DFSMSstvs, an optional feature of z/OS

This book is organized into several chapters, where we discuss the following topics:

- ▶ Chapter 2, “Data set basics” on page 15
- ▶ Chapter 3, “Storage management hardware” on page 45
- ▶ Chapter 4, “Storage management software” on page 99
- ▶ Chapter 5, “System-managed storage” on page 189

# 1.1 Introduction to DFSMS

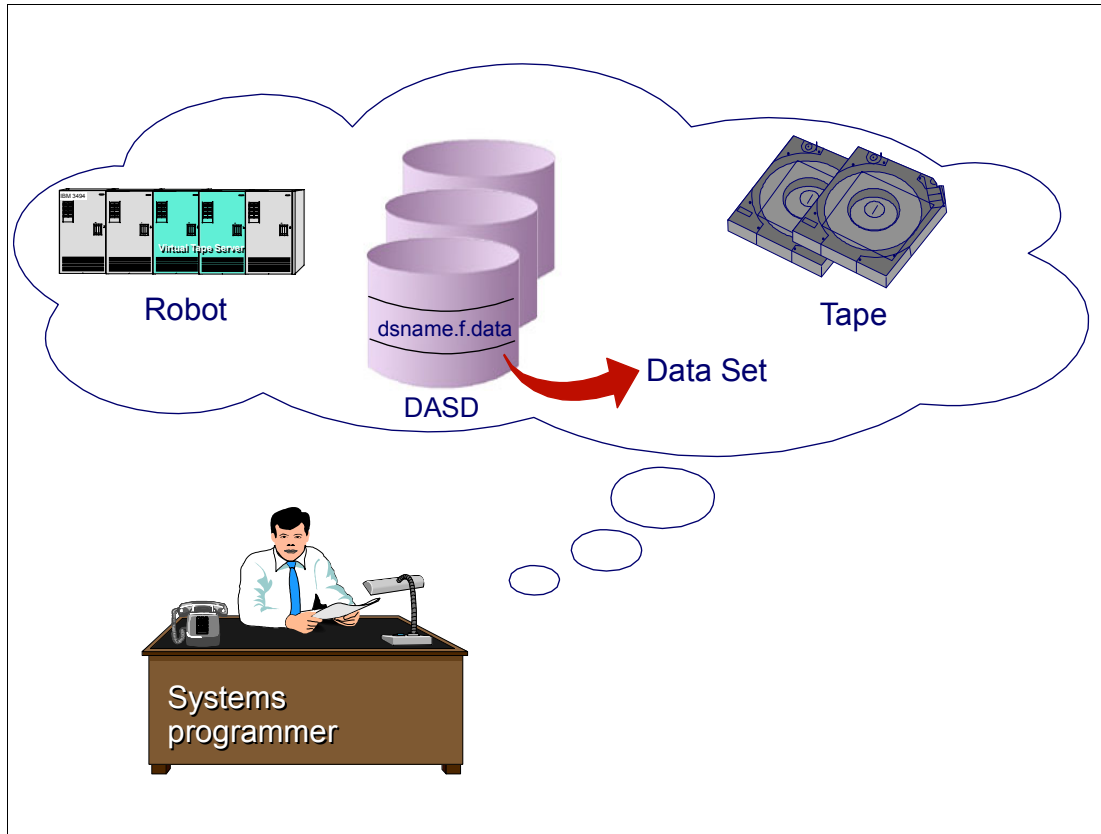


Figure 1-1 Introduction to data management

## Understanding 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.

DFSMSdfp helps you store and catalog information on 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.

## Systems programmer

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

- ▶ 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

## 1.2 Data Facility Storage Management Subsystem

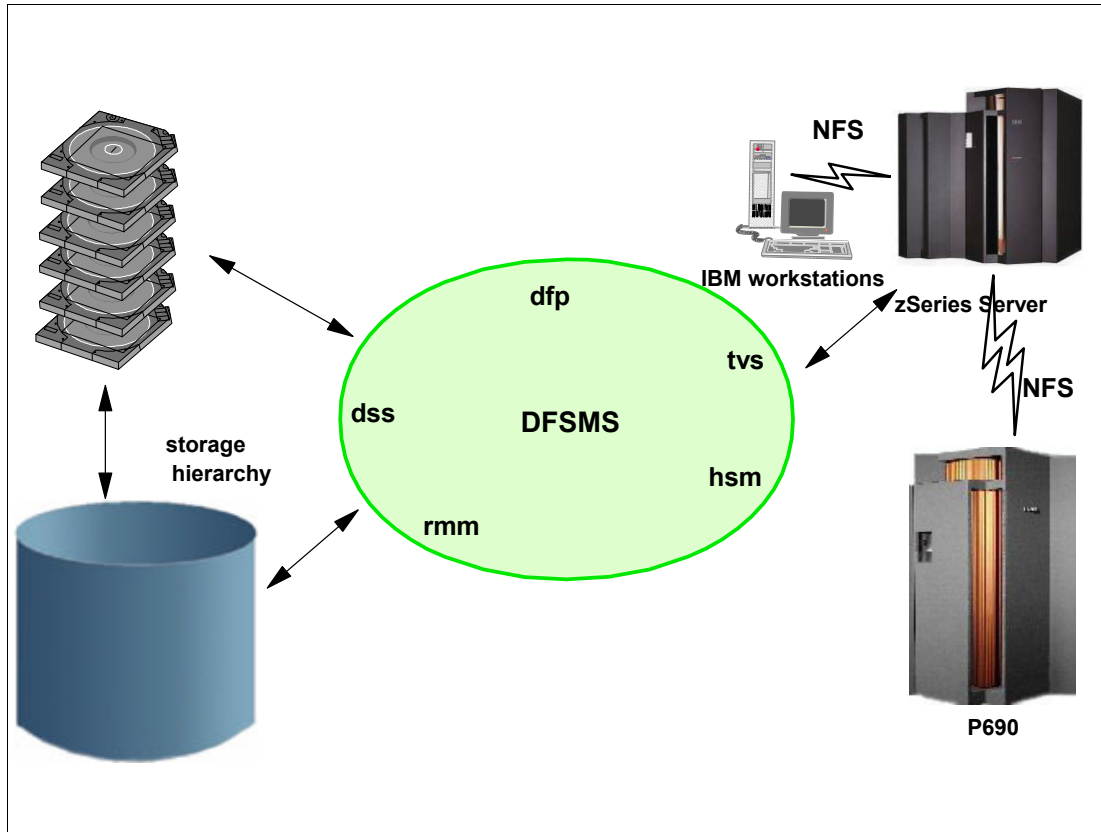


Figure 1-2 Data Facility Storage Management Subsystem

### DFSMS components

DFSMS is a set of products associated with z/OS that is responsible for data management. DFSMS has five MVS data management functional components as an integrated single software package:

- DFSMSdfp** Provides storage, data, program, and device management. It is comprised of several components such as access methods, OPEN/CLOSE/EOV routines, catalog management, DADSM (DASD space control), utilities, IDCAMS, SMS, NFS, ISMF, and other functions.
- DFSMSdss** Provides data movement, copy, backup, and space management functions.
- DFSMShsm** Provides backup, recovery, migration, and space management functions. It invokes DFSMSdss for certain of its functions.
- DFSMSrmm** Provides management functions for removable media such as tape cartridges and optical media.
- DFSMSstvs** Enables batch jobs and 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 located on remote computers as if they were local. NFS is independent of machine types, operating systems, and network architectures.

## 1.3 DFSMSdfp component

- ❑ DFSMSdfp provides the following functions:
  - Managing storage
  - Managing data
  - Using access methods, commands, and utilities
  - Managing devices
  - Tape mount management
  - Distributed data access
  - Advanced copy services
  - Object access method (OAM)

### DFSMSdfp component

DFSMSdfp provides storage, data, program, and device management. It is comprised of several components such as access methods, OPEN/CLOSE/EOV routines, catalog management, DADSM (DASD space control), utilities, IDCAMS, SMS, NFS, ISMF, and other functions.

### Managing storage

The storage management subsystem (SMS) is a DFSMSdfp facility designed for automating and centralizing storage management. SMS automatically assigns attributes to new data when that data is created. SMS automatically controls system storage and assigns data to the appropriate storage device. ISMF panels allow you to specify these data attributes.

For more information on ISMF, see “Introduction to ISMF” on page 264.

### 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

For more information, see “Establishing service level objectives” on page 195.

## **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 or 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 different 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. Once on DASD, you can migrate these data sets to a single tape or small set of tapes, thereby reducing the overhead associated with multiple tape mounts.

## **Distributed data access with DFSMSdfp**

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

Distributed FileManager/MVS is a DFSMSdfp client/server product that enables remote clients in a network to access data on z/OS or OS/390 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.

## **Advanced copy services**

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

## 1.4 DFSMSDss component

- ❑ DFSMSDss provides the following functions:
  - Data movement and replication
  - Space management
  - Data backup and recovery
  - Data set and volume conversion
  - Distributed data management
    - FlashCopy feature with Enterprise Storage Server (ESS)
    - SnapShot feature with RAMAC Virtual Array (RVA)
  - Concurrent copy

### **DFSMSDss component**

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 lets you move or copy data between volumes of like and unlike device types. If you create a backup in DFSMSDss, you can copy a backup copy of data. DFSMSDss also can produce multiple backup copies during a dump operation.

### **Space management**

DFSMSDss can reduce or eliminate DASD free-space fragmentation.

### **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 FlashCopy® feature with Enterprise Storage Server® (ESS) and the SnapShot feature with RAMAC® Virtual Array (RVA). 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 lets you 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 if no updates have occurred.



## 1.5 DFSMSrmm component

□ DFSMSdmm provides the following functions:

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

### **DFSMSrmm component**

DFSMSrmm manages your removable media resources, including tape cartridges and reels. It provides the following functions.

### **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 TotalStorage®
    - Enterprise Automated Tape Library (3494)
    - IBM TotalStorage
    - Virtual Tape Servers (VTS)
- ▶ 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 keeps track of the volumes residing 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 life cycle.

**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.

## 1.6 DFSMSHsm component

□ DFSMSHsm provides the following functions:

- Storage management
- Space management
- Tape mount management

### **DFSMSHsm component**

DFSMSHsm complements DFSMSdss to provide the following functions.

#### **Storage management**

DFSMSHsm provides automatic DASD storage management, thus relieving users from manual storage management tasks.

#### **Space management**

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.

#### **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 interval migration. 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.

## **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 should occur. 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.

## 1.7 DFSMStvs component

- ❑ Provide transactional recovery within VSAM
- ❑ RLS allows batch sharing of recoverable data sets for read
  - RLS provides locking and buffer coherency
  - CICS provides logging and two-phase commit protocols
- ❑ Transactional VSAM allows batch sharing of recoverable data sets for update
  - Logging provided using the System Logger
  - Two-phase commit and backout using Recoverable Resource Management Services (RRMS)

### DFSMStvs component

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.

### VSAM record-level sharing (RLS)

With VSAM RLS, multiple CICS systems can directly access a shared VSAM data set, eliminating the need to ship 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 record-level sharing (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 need only 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.

### **Recoverable resource management services (RRMS)**

RRMS is part of the operating system and comprises registration services, context services, and recoverable resource 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, and it can be a source program, a library of macros, or a file of data records used by a processing program. Data records are the basic unit of information used by a processing program. By placing your data into volumes of organized data sets, you can save and process the data. You can also print the contents of a data set, or display the contents on a terminal.

You can store data on secondary storage devices, such as:

- ▶ A direct access storage device (DASD)

The term DASD applies to disks or to a mass storage medium on which a computer stores data. A *volume* is a standard unit of secondary storage. You can store all types of data sets on DASD.

Each block of data on a DASD volume has a distinct location and a unique address, thus 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 different data sets, and reallocate or reuse space on the volume.

- ▶ A magnetic tape volume

Only sequential data sets can be stored on magnetic tape. Mountable tape volumes can reside in an automated tape library. For information about magnetic tape volumes, see *z/OS DFSMS: Using Magnetic Tapes*, SC26-7412. You can also direct a sequential data set to or from spool, a UNIX file, a TSO/E terminal, a unit record device, virtual I/O (VIO), or a dummy data set.

The Storage Management Subsystem (SMS) is an operating environment that automates the management of storage. Storage management uses the values provided at allocation time to determine, for example, on which volume to place your data set, and how many tracks to allocate for it. Storage management also manages tape data sets on mountable volumes that reside in an automated tape library. With SMS, users can allocate data sets more easily.

The data sets allocated through SMS are called *system-managed data sets* or *SMS-managed data sets*.

An access method defines the technique that is used to store and retrieve data. Access methods have their own data set structures to organize data, macros to define and process data sets, and utility programs to process data sets.

Access methods are identified primarily by the data set organization. For example, use the basic sequential access method (BSAM) or queued sequential access method (QSAM) with sequential data sets. However, there are times when an access method identified with one organization can be used to process a data set organized in a different manner. For example, a sequential data set (not extended-format data set) created using BSAM can be processed by the basic direct access method (BDAM), and vice versa. Another example is UNIX files, which you can process using BSAM, QSAM, basic partitioned access method (BPAM), or virtual storage access method (VSAM).

This chapter describes some basics relating to data sets:

- ▶ Data set name rules
- ▶ Data set characteristics
- ▶ Locating a data set
- ▶ Volume table of contents (VTOC)
- ▶ Initializing a volume



## 2.1 Data sets

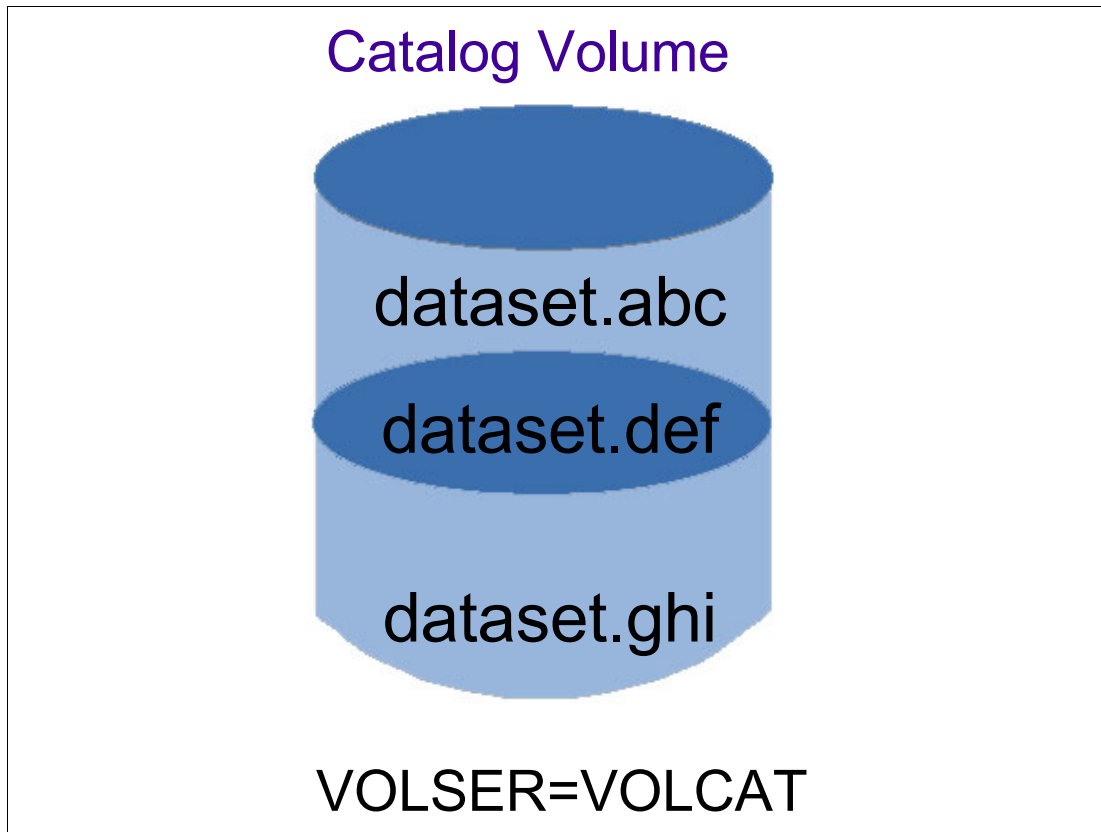


Figure 2-1 Data sets

### MVS data sets

An *MVS data set* is a collection of logically related data records stored on one volume 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 terminal. 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 Hierarchical File System (HFS) data sets, where the collection is of bytes and there is not the concept of logically related data records.

Data can be stored on a direct access storage device (DASD), magnetic tape volume, or optical media. As mentioned, 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. We discuss the types of data sets later.

The next visuals discuss the logical attributes of a data set which are specified at data set allocation time in:

- ▶ DCB/ACB control blocks in the application program
- ▶ DD cards (explicitly, or through the Data Class (DC) option with DFSMS)
- ▶ In an ACS Data Class (DC) routine (overridden by a DD card)

After the allocation, such attributes are kept in catalogs and VTOCs.

## 2.2 Data set name rules

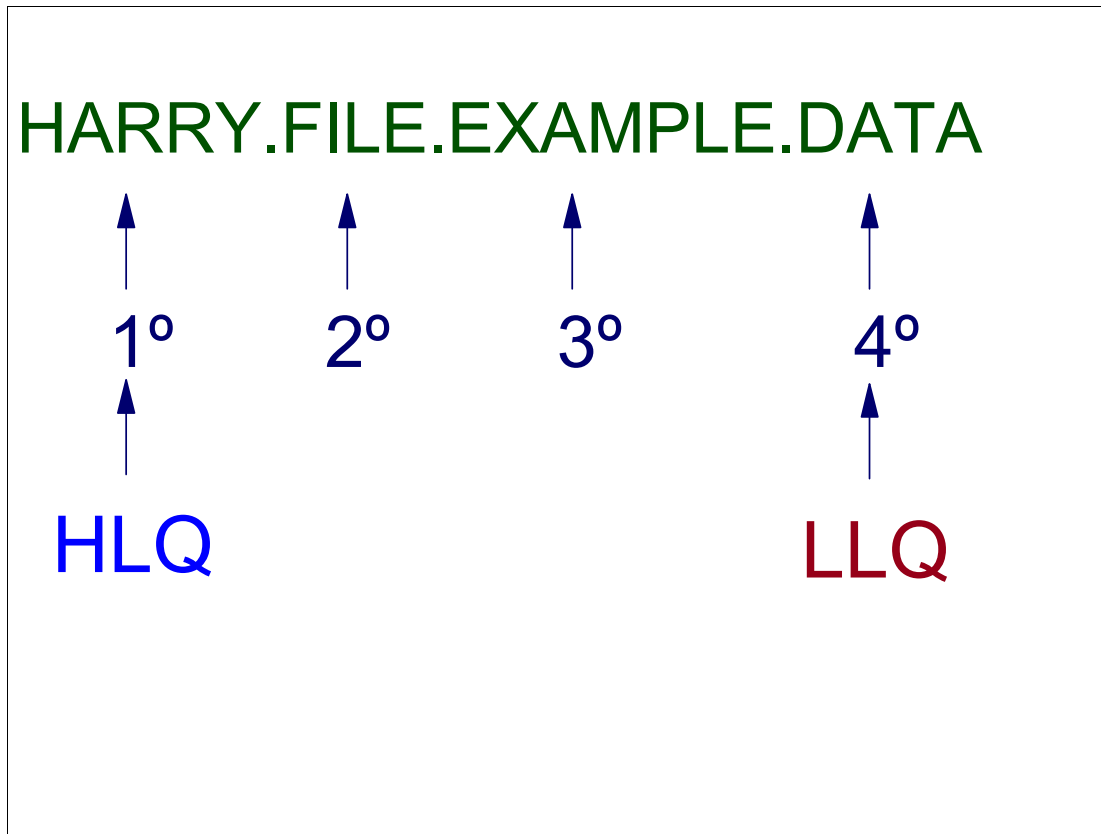


Figure 2-2 Data set name rules

### Data set naming conventions

Whenever you allocate a new data set, you (or MVS) must give the data set a unique name. Usually, the data set name is given as the DSNAME keyword in 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 VERA.LUZ.DATA is composed of three name segments. The first name on the left is called the high-level qualifier (HLQ), the last name on the right is the lowest-level qualifier (LLQ).

Each name segment (qualifier) is one to eight characters in length, the first of which must be alphabetic (A to Z) or national (# @ \$). The remaining seven characters are either alphabetic, numeric (0-9), national, or a hyphen (-). The period (.) separates name segments from each other.

**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 DFSMSdfp data set organizations

- Data set types supported
  - VSAM data sets
  - Non-VSAM data sets
  - Extended-format data sets
  - Objects
  - z/OS UNIX files

Figure 2-3 DFSMSdfp data set types supported

### DFSMSdfp data set types

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

#### VSAM data sets

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.

#### Non-VSAM data sets

Non-VSAM data sets are collections of fixed-length or variable-length records, grouped into blocks. To access non-VSAM data sets, use BSAM, QSAM, or BPAM.

#### Extended-format data sets

You can allocate both sequential and VSAM data sets in extended format on a system-managed DASD. The DASD is attached to a controller that supports Extended Platform.

## **Objects**

*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 a DB2® database, or on an optical drive, or on an optical or tape storage volume.

## **z/OS UNIX files**

z/OS UNIX System Services (z/OS UNIX) enables 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.

## 2.4 Types of VSAM data sets

- Types of VSAM data sets
  - Key-sequenced data set (KSDS)
  - Entry-sequenced data set (ESDS)
  - Relative-record data set (RRDS)
  - Linear data set (LDS)

Figure 2-4 VSAM data set types

### **VSAM data sets**

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.

### **Key-sequenced data set (KSDS)**

A KSDS VSAM data set contains records in order by a key field and can be accessed by the key or by a relative byte address. The key contains a unique value, such as an employee number or part number.

### **Entry-sequenced data set (ESDS)**

A ESDS VSAM data set contains records in the order in which they were entered and can only be accessed by relative byte address. An ESDS is similar to a sequential data set.

### **Relative-record data set (RRDS)**

A RRDS VSAM data set contains records in order by relative-record number and can only be accessed by this number. Relative records can be fixed-length or variable-length.

### **Linear data set (LDS)**

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

## 2.5 Non-VSAM data sets

- Types of non-VSAM data sets
  - Sequential data set (PS)
  - Partitioned data set (PDS)
  - Partitioned data set extended (PDSE)

Figure 2-5 Types of non-VSAM data sets

### Non-VSAM data sets

Non-VSAM data sets are collections of fixed-length or variable-length records, grouped into blocks, and these types of data sets are called *MVS data sets*. To access non-VSAM data sets, use BSAM, QSAM, or BPAM. There are several types of non-VSAM data sets, as explained here.

#### Sequential data set (PS)

Sequential data sets contain 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.

#### Partitioned data set (PDS)

Partitioned data sets contain a directory of sequentially organized members, each of which can contain a program or data. After opening the data set, you can retrieve any individual member without searching the entire data set.

#### Partitioned data set extended (PDSE)

Partitioned data sets extended contain an indexed, expandable directory of sequentially organized members, each of which can contain a program or data. You can use a PDSE instead of a PDS. The main advantage of using a PDSE over a PDS is that a PDSE automatically reuses space within the data set.

## 2.6 Extended-format data sets and objects

- ❑ An extended-format data set supports the following formats:
  - Compression
  - Data striping
  - Extended-addressability
- ❑ Objects
  - Use object access method (OAM)
  - Storage administrator assigns objects

Figure 2-6 Types of extended-format data sets

### Extended-format data sets

You can allocate both sequential and VSAM data sets in extended format on a system-managed DASD. The DASD is attached to a controller that supports Extended Platform. Extended-format VSAM data sets allow you to release partial space and to use system-managed buffering for VSAM batch programs. You can select whether to use the primary or secondary space amount when extending VSAM data sets to multiple volumes.

An extended-format data set supports the following formats:

- ▶ Compression, which reduces the space for storing data and improves I/O, caching, and buffering performance.
- ▶ Data striping, which distributes data for one data set across multiple SMS-managed DASD volumes, which improves I/O performance and reduces the batch window. For example, a data set with 28 stripes is distributed across 28 volumes.  
  
Large data sets with high I/O activity are the best candidates for striped data sets. Data sets defined as extended-format sequential must be accessed using BSAM or QSAM, and *not* EXCP or BDAM.
- ▶ Extended-addressability, which enables you to create a VSAM data set that is larger than 4 GB.

## **Objects**

*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. 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.



## 2.7 z/OS UNIX files

- ❑ Following are the types of z/OS UNIX files:
  - Hierarchical file system (HFS)
  - Network File System (NFS)
  - zSeries File System (zFS)
  - Temporary file system (TFS)

Figure 2-7 z/OS UNIX files

### **z/OS UNIX**

z/OS UNIX System Services (z/OS UNIX) enables z/OS to access UNIX files. UNIX applications also can access z/OS data sets. z/OS UNIX files are byte-oriented, similar to objects. Following are the types of z/OS UNIX files.

#### **Hierarchical file system (HFS)**

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, which consists of directories and all its related files. HFS data sets must reside on DASD volumes.

#### **z/OS Network File System (z/OS NFS)**

z/OS NFS is a distributed file system that enables users to access UNIX files and directories that are located on remote computers as if they were local. z/OS NFS is independent of machine types, operating systems, and network architectures.

#### **zSeries File System (zFS)**

A zFS is a UNIX file system that supports one file system or multiple file systems in a linear VSAM (LDS) data set.

### **Temporary file system (TFS)**

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

## 2.8 Data set organization (DSORG)

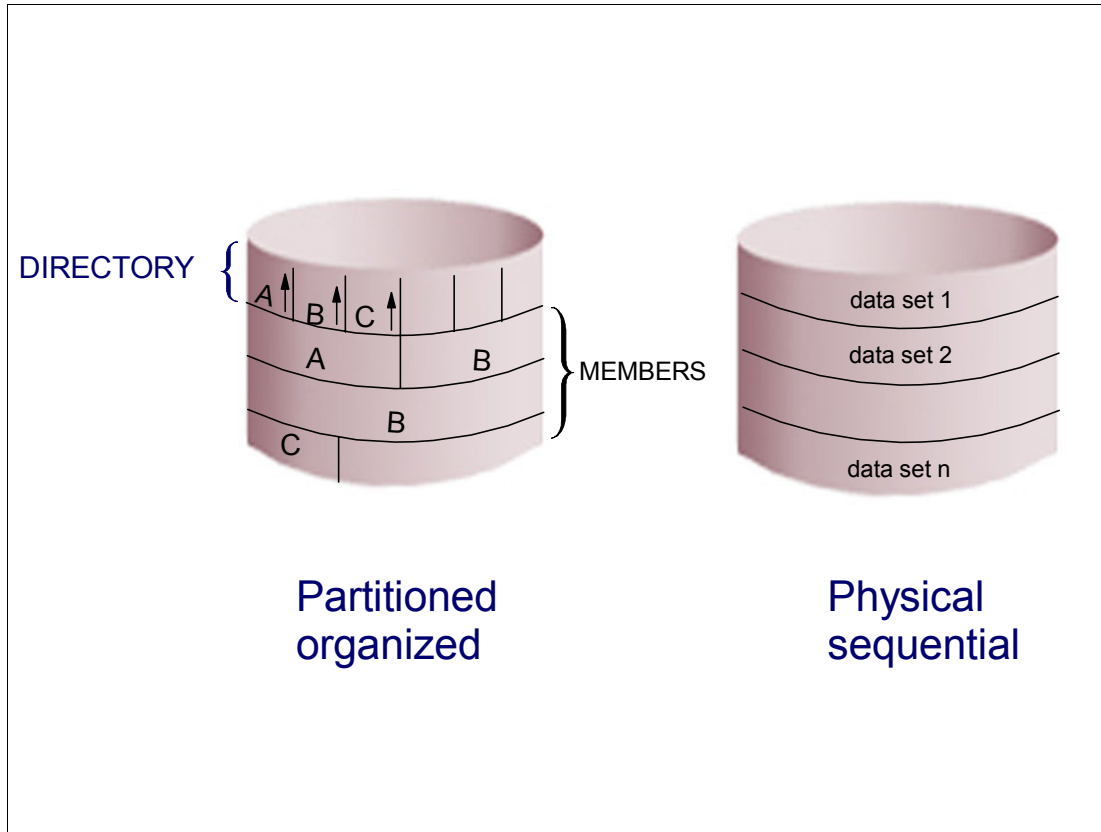


Figure 2-8 Data set organization (DSORG)

### Data Set Organization (DSORG)

There are different types of data set organization used in z/OS. Each organization provides specific benefits to the user. The two shown in Figure 2-8 are:

- ▶ 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.

### Physical sequential (PS)

With this data set organization, the records can only be read or written in “physical sequential” order. If we compare this with a PC file, this is a file in the main directory (C:\).

Sequential data sets can exist on DASD, tape, and optical devices.

### Partitioned Organized (PO)

Partitioned data sets are similar in organization to a library and are often referred to this way. A library contains normally a great number of “books”, and sorted directory entries are used to locate them.

In PDS, or partitioned organized data set, the “books” are called *members* and to locate them, they are pointed to by entries in a *directory*, as shown in Figure 2-8.

The members are individual sequential data sets and can be read or written sequentially, once they have been located via directory. It is almost the same idea as the directory and file organization in a PC.

Partitioned data sets can only exist on DASD. Each member has a unique name, one to eight characters in length, stored in a directory that is part of the data set. The records of a given member are written or retrieved sequentially. See *z/OS DFSMS Macro Instructions for Data Sets*, SC26-7408, for the macros used with partitioned data sets.

The main advantage of using a partitioned data set is that, without searching the entire data set, you can retrieve any individual member after the data set is opened. For example, in a program library (always a partitioned data set) each member is a separate program or subroutine. The individual members can be added or deleted as required. When a member is deleted, the member name is removed from the directory, but the space used by the member cannot be reused until the data set is reorganized; that is, compressed using the IEBCOPY utility (generally requested through an ISPF panel). See “IEBCOPY utility” on page 105 for information about this topic.

## **Directory**

The directory, a series of 256-byte records at the beginning of the data set, contains an entry for each member. Each directory entry contains the member name and the starting location of the member within the data set, as shown. Also, you can specify as many as 62 bytes of information in the entry. The directory entries are arranged by name in alphanumeric collating sequence. Each directory block contains a two-byte count field that specifies the number of active bytes in a block (including the count field). Each block is preceded by a hardware-defined key field containing the name of the last member entry in the block (that is, the member name with the highest binary value).

Partitioned data set member entries vary in length, and are blocked into the member area.

## **BLKSIZE**

If you do not specify a block size (BLKSIZE), the Open routine determines an optimum block size for you. Therefore, you no longer need to perform calculations based on track length. When you allocate space for your data set, you can specify the average record length in kilobytes or megabytes by using the SPACE and AVGREC parameters, and have the system use the block size it calculated for your data set.

## **PDSE**

Another type of PO data set is the PDSE, which must be SMS-managed. See “Partitioned data set extended (PDSE)” on page 138 for more information about this topic.

Refer to “Access method” on page 133 for more types and more information about data set organization.

## 2.9 Allocate a data set with ISPF 3.2

```
Menu RefList Utilities Help
-----
Allocate New Data Set
Command ==> _____
Data Set Name . . . . : ROGERS.JCL.TEST
Management class . . . . _____ (Blank for default management class)
Storage class . . . . _____ (Blank for default storage class)
Volume serial . . . . 037CAT (Blank for system default volume) **
Device type . . . . _____ (Generic unit or device address) **
Data class . . . . _____ (Blank for default data class)
Space units . . . . CYLINDER (BLKS, TRKS, CYLS, KB, MB, BYTES
or RECORDS)
Average record unit _____ (M, K, or U)
Primary quantity . . 5 (In above units)
Secondary quantity . 1 (In above units)
Directory blocks . . 50 (Zero for sequential data set) *
Record format . . . . FB
Record length . . . . 80
Block size . . . . 27920
Data set name type : PDS (LIBRARY, HFS, PDS, or blank) *
Expiration date . . . _____ (YY/MM/DD, YYYY/MM/DD
Enter "/" to select option DDDD for retention period in days
or blank)
_ Allocate Multiple Volumes

( * Specifying LIBRARY may override zero directory block)

( ** Only one of these fields may be specified)
```

Figure 2-9 Allocating a data set with ISPF option 3.2

### Allocating data sets

To process a data set, first allocate it (establish a link to it), then access the data using macros for the access method that you have chosen. The allocation of a data set means either or both of two things:

- ▶ To set aside (create) space for a new data set on a disk or tape
- ▶ To establish a logical link between a job step and any data set using JCL

Figure 2-9 shows the allocation of a data set using ISPF panel 3.2. Other ways to allocate a data set are by using any of the following methods:

- ▶ Access method services

You can define data sets and establish catalogs by using a multifunction services program called *access method services*. Use the ALLOCATE command to create the data set.

- ▶ ALLOCATE command

You can also issue the ALLOCATE command through TSO/E to define VSAM and non-VSAM data sets.

- ▶ Using JCL

Any data set can be defined directly through JCL.

## 2.10 Logical record length (LRECL)

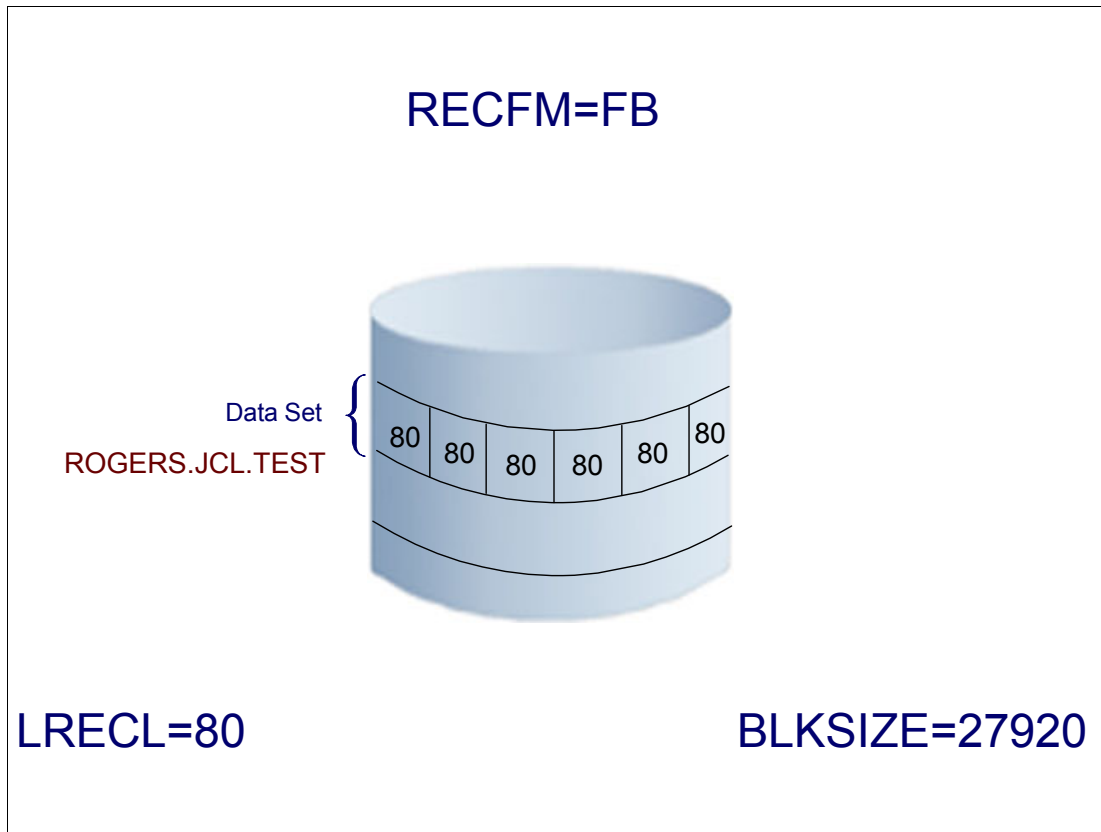


Figure 2-10 Logical record length

### Accessing a data set

After the data set has been allocated, it can be accessed (used). In Figure 2-9, the allocation requested a LRECL=80, BLKSIZE=27920, and RECFM=FB.

### Logical records and block sizes

A *logical record* (LRECL) is a unit of information about a unit of processing (for example, a customer, an account, a payroll employee, and so on). It is the smallest amount of data to be processed, and it is comprised of fields which contain information recognized by the processing application.

Logical records, when located in DASD, tape, or optical devices, are grouped in physical records named blocks (BLKSIZE). Each block of data on a DASD volume has a distinct location and a unique address, thus making it possible to find any block without extensive searching. Logical records can be stored and retrieved either directly or sequentially.

DASD volumes are used for storing data and executable programs (including the operating system itself), and for temporary working storage. One DASD volume can be used for many different data sets, and space on it can be reallocated and reused. The maximum length of a logical record (LRECL) is limited by the physical size of the used media.

## **Record formats**

Use the RECFM parameter to specify the format and characteristics of the logical records in a new data set. RECFM specifies the characteristics of the records in the data set as fixed-length (F), variable-length (V), ASCII variable-length (D), or undefined-length (U).

Blocked records are specified as FB, VB, or DB. Spanned records are specified as VS, VBS, DS, or DBS. You can also specify the records as fixed-length standard by using FS or FBS. You can request track overflow for records other than standard format by adding a T to the RECFM parameter (for example, by coding FBT). Track overflow is ignored for PDSEs.

## 2.11 Locating a data set

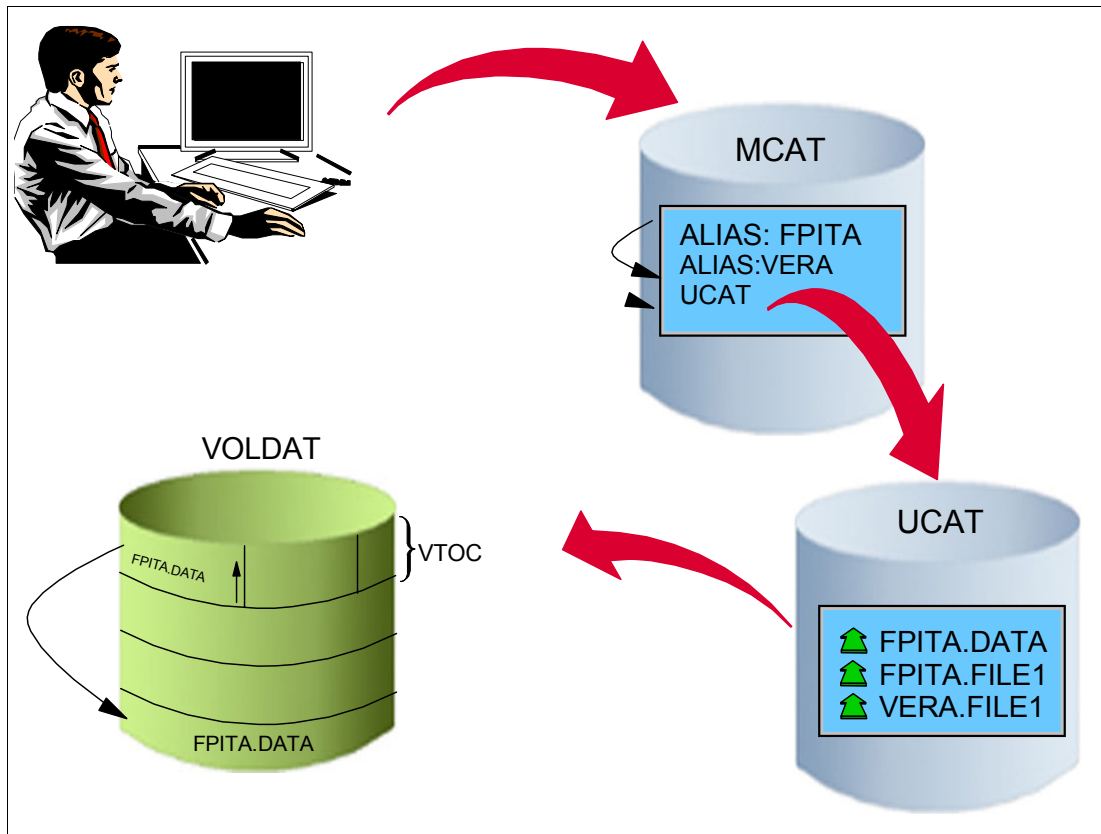


Figure 2-11 Locating a data set

### Locating a data set

Before we describe the procedure used to find a data set, let's introduce terms used in this document (they are explained in more detail later).

#### VTOC

A sequential data set located in a DASD volume that describes the contents of this volume.

#### User Catalogs (UCAT)

A catalog of data sets used to locate in which DASD volume the requested data set is stored; user data sets are cataloged in this type of catalog.

#### Master Catalog (MCAT)

This has the same structure as a user catalog, but points to system data sets. It also contains information about the user catalog location and any alias pointer.

#### Alias

A special entry in the master catalog pointing to a user catalog that coincides with the HLQ of a data set. It means that the data set with this HLQ is cataloged in that user catalog. The alias is used to find in which user catalog that data set location information exists.

### Locating a data set sequence

When the system tries to locate a data set for a request for an existing data set, the following sequence takes place:

- ▶ The MCAT is searched; if found, verify if it is:



- A data set name, then pick up the volume specification and if the indicated device is online, then check VTOC to locate the data set in the specified volume.
  - An alias, that is, the HLQ of the data set name is equal to an alias entry pointing to an UCAT. In this case, go to the referred UCAT.
- ▶ The UCAT is searched (if there is a match in the alias). If the data set name is found, proceed as in an MCAT hit.

Finally, the requesting program can access the data set.

**Note:** It is not recommended that you use private catalogs. One reason for this recommendation is because for SMS-managed data sets, SMS only accesses SMS-managed data sets that are cataloged in a system catalog.

## 2.12 Uncataloged and cataloged data sets

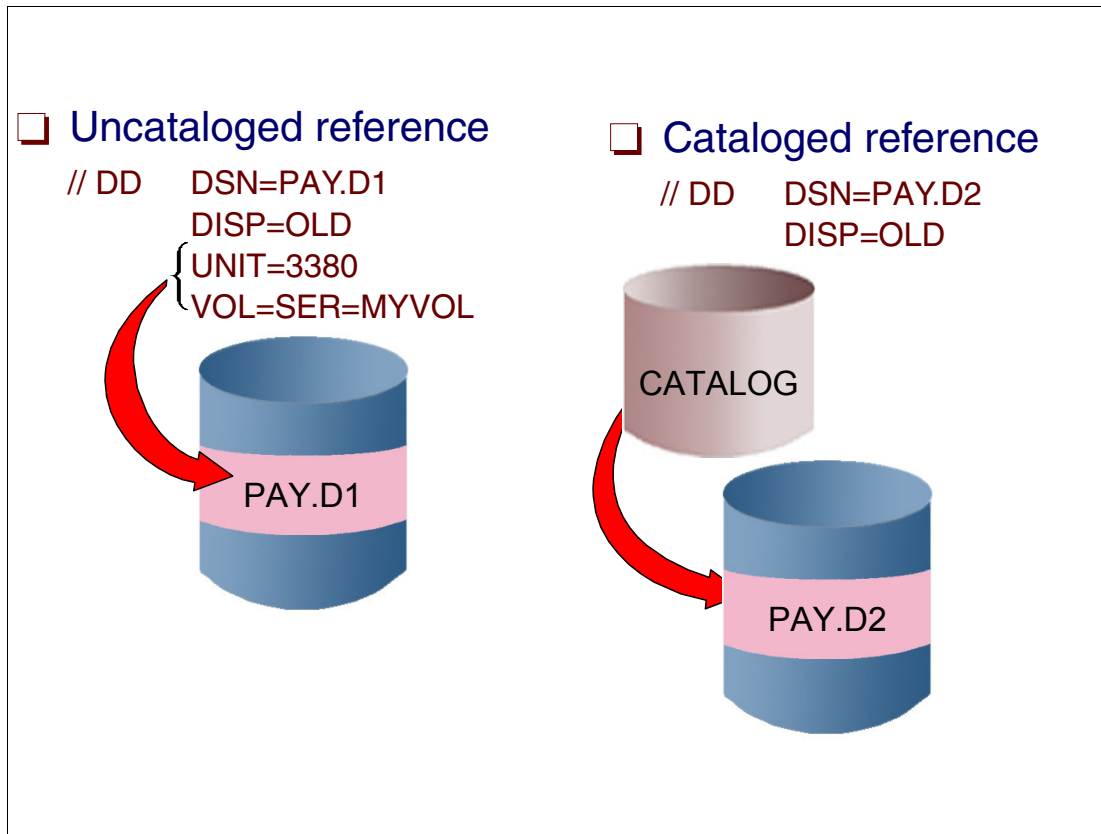


Figure 2-12 Cataloged and uncataloged data sets

### Cataloged data sets

When the data set is cataloged, the system obtains unit and volume information from the catalog. However, if the DD statement for a catalog data set contains `VOLUME=SER=serial-number`, the system does not look in the catalog; in this case, you must code the `UNIT` parameter.

### Uncataloged data sets

When your data set is not cataloged you *must* know in advance its volume location and specify it in your JCL. This can be done through the `UNIT` and `VOL=SER`, as shown in Figure 2-12.

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

**Note:** We strongly recommend that you do not have uncataloged data sets in your installation because uncataloged data sets can cause problems with duplicate data and possible incorrect data set processing.

## 2.13 Volume table of contents (VTOC)

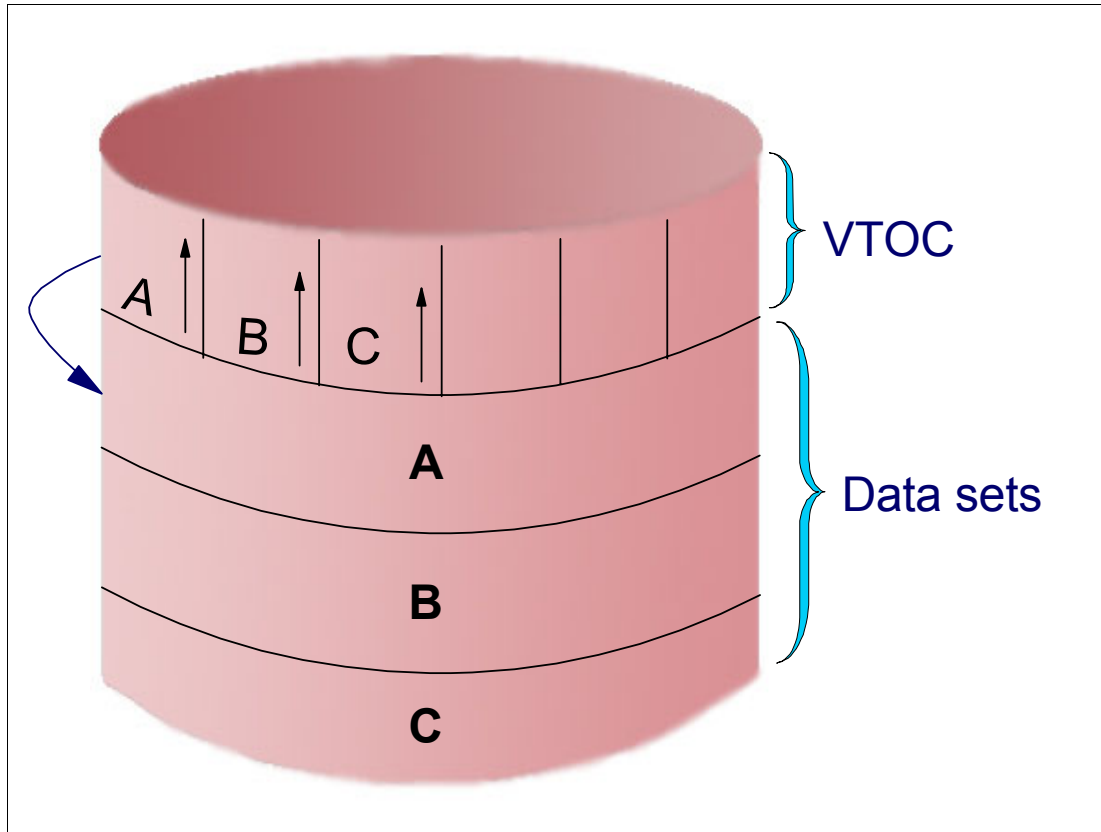


Figure 2-13 Volume table of contents (VTOC)

### VTOC

The VTOC lists the data sets that reside on its volume, along with information about the location and size of each data set, and other data set attributes.

The VTOC is a contiguous data set; that is, it resides in a single extent on the volume. It is pointed at by the record in the first track of the volume, and starts after cylinder 0, track 0 and before track 65,535.

A VTOC's address is located in the VOLVTOC field of the standard volume label. Data is organized in physical blocks preceded by the highest record key in the block (that is, a count-key-data format).

The VTOC has six types of control blocks; they are called data set control blocks (DSCB) and they describe data set characteristics, free space, and other functions that we will see in the next visuals.

There is a set of macros called the Common VTOC Access Facility (CVAF) that allows a program to access VTOC information data.

## 2.14 Data set control block (DSCB)

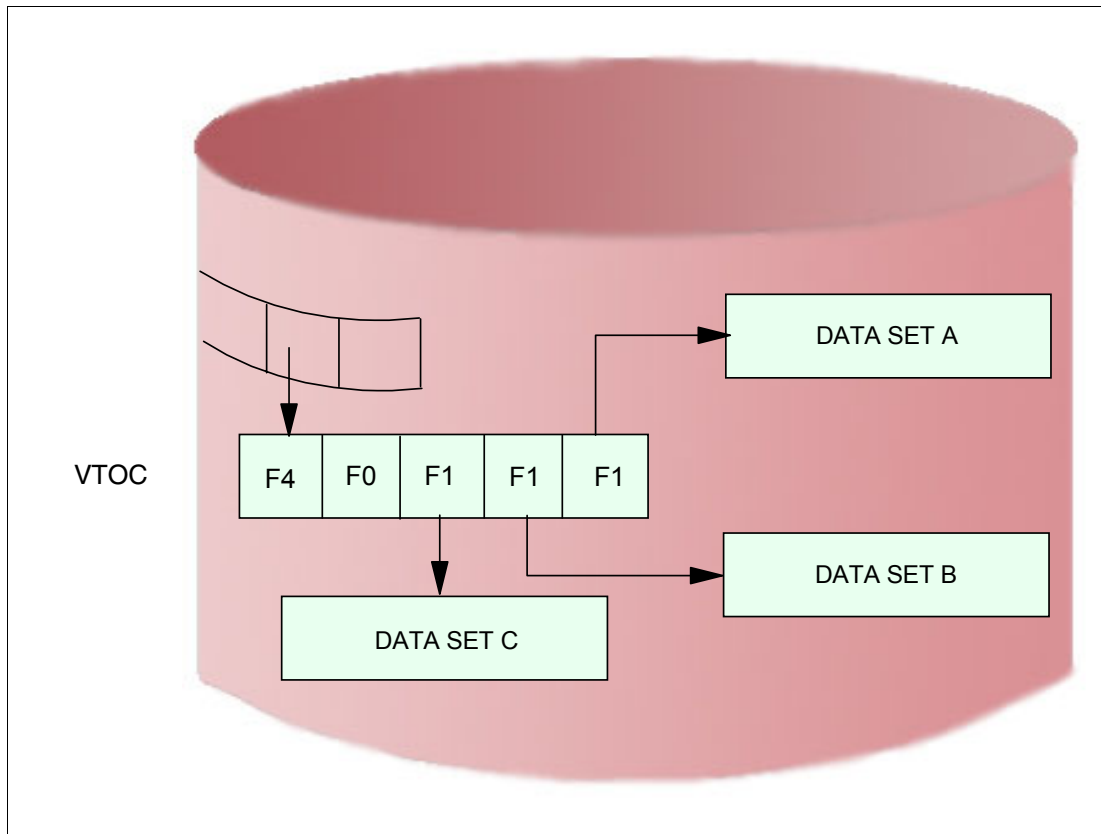


Figure 2-14 Data set control block (DSCB)

### DSCBs

The VTOC is composed of 140-byte (1) data set control blocks (DSCBs) that correspond either to a data set or virtual storage access method (VSAM) data space currently residing on the volume, or to contiguous, unassigned tracks on the volume.

DSCB is the name of the logical record within the VTOC. DSCBs describe data sets allocated in that volume, and also describe the VTOC itself. The system automatically constructs a DSCB when space is requested for a data set on a direct access volume. Each data set on a DASD volume has one or more DSCBs to describe its characteristics. The DSCB appears in the VTOC and, in addition to space allocation and other control information, contains operating system data, device-dependent information, and data set characteristics. There are seven kinds of DSCBs, each with 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 the data set resides on, the volume attributes, and the size and contents of the VTOC data set. The next DSCB in the VTOC data set is a free-space DSCB (format-5), even if the free space is described by format-7 DSCBs. The third and subsequent DSCBs in the VTOC can occur in any order.

Table 2-1 on page 37 describes the different types of DSCBs.

Table 2-1 DSCBs that can be found in the VTOC

Type	Name	Function	How many
0	Free VTOC DSCB	Describes an unused record in the VTOC (contains 140 bytes of binary zeros). To delete a DSCB from the VTOC, a format-0 DSCB is written over it.	One for every unused 140-byte record on the VTOC. The DS4DSREC field of the format-4 DSCB is a count of the number of format-0 DSCBs on the VTOC. This field is not maintained for an indexed VTOC.
1	Identifier	Describes the first three extents of a data set or VSAM data space.	One for every data set or data space on the volume, except the VTOC.
2	Index	Describes the indexes of an ISAM data set.	One for each ISAM data set (for a multivolume ISAM data set, a format-2 DSCB exists only on the first volume).
3	Extension	Describes extents after the third extent of a non-VSAM data set or a VSAM data space.	One for each data set or VSAM data space on the volume that has more than three extents. There can be as many as 10 for a PDSE, HFS, extended format data set, or a VSAM data set cataloged in an integrated catalog facility catalog. PDSEs, HFS, and extended format data sets can have up to 123 extents. Each component of a VSAM data set cataloged in an integrated catalog facility catalog can have up to 123 extents per volume. All other data sets are restricted to 16 extents per volume.
4	VTOC	Describes the extent and contents of the VTOC, and provides volume and device characteristics. This DSCB contains a flag indicating whether the volume is SMS-managed.	One on each volume.
5	Free space	On a nonindexed VTOC, describes the space on a volume that has not been allocated to a data set (available space). For an indexed VTOC, a single empty format-5 DSCB resides in the VTOC; free space is described in the index and DS4IVTOC is normally on.	One for every 26 noncontiguous extents of available space on the volume for a nonindexed VTOC; for an indexed VTOC, there is only one.
7	Free space for certain device	Only one field in the format-7 DSCB is an intended interface. This field indicates whether the DSCB is a format-7 DSCB. You can reference that field as DS1FMTID or DS5FMTID. A character 7 indicates that the DSCB is a format-7 DSCB, and your program should not modify it.	

## 2.15 VTOC index structure

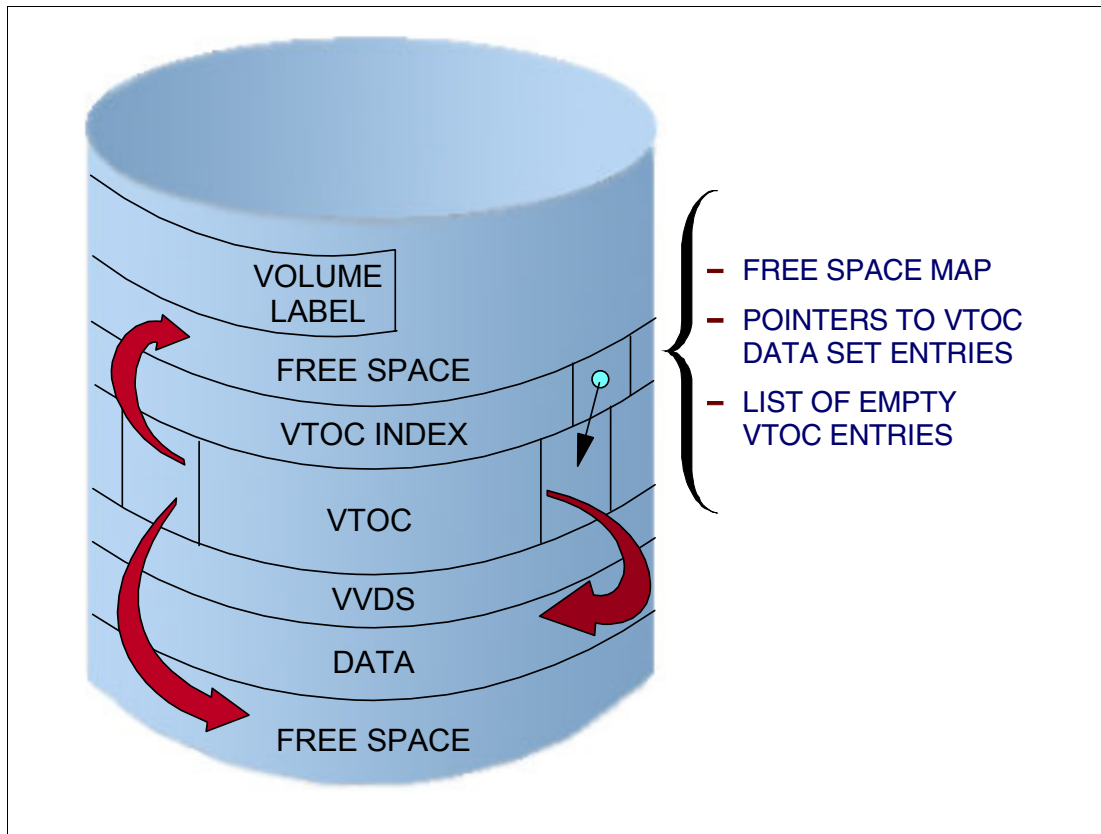


Figure 2-15 VTOC index structure

### VTOC index

A VTOC index decreases search time for DSCBs. 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. If a VTOC index becomes disabled, 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.

### Device Support Facilities (ICKDSF)

Device Support Facilities (ICKDSF) initializes a VTOC index into 2048-byte physical blocks named VTOC index records (VIRs). VIRs are used in several ways. A VTOC index contains the following kinds of VIRs:

- ▶ VTOC index entry record (VIER) identifies the location of format-1 DSCBs and the format-4 DSCB.
- ▶ VTOC pack space map (VPSM) identifies the free and allocated space on a volume.
- ▶ VTOC index map (VIXM) identifies the VIRs that have been allocated in the VTOC index.
- ▶ VTOC map of DSCBs (VMDS) identifies the DSCBs that have been allocated in the VTOC.

## **VTOC format-1 DSCB**

A format-1 DSCB in the VTOC contains the name and extent information of the VTOC index. The name of the index must be 'SYS1.VTOCIX.xxxxxxxx', where 'xxxxxxx' conforms to standard data set naming conventions and is usually the serial number of the volume containing the VTOC and its index. The name must be unique within the system to avoid ENQ contention.

## **Creating the VTOC and VTOC index**

To initialize a volume (preparing for I/O activity), use the Device Support Facilities (ICKDSF) utility to initially build the VTOC. You can create a VTOC index at that time by using the ICKDSF INIT command and specifying the INDEX keyword.

You may use ICKDSF to convert a non-indexed VTOC to an indexed VTOC by using the **BUILDIX** command and specifying the IXVTOC keyword. The reverse operation can be performed by using the **BUILDIX** command and specifying the OSVTOC keyword. For details see *Device Support Facilities User's Guide and Reference Release 17*, GC35-0033, and refer to *z/OS DFSMSdfp Advanced Services*, SC26-7400, for more information on that topic.

## 2.16 Initializing a volume (ICKDSF)

```
//EXAMPLE    JOB
//EXEC      PGM=ICKDSF
//SYSPRINT  DD      SYSOUT=A
//SYSIN     DD      *
INIT UNITADDRESS(0353) NOVERIFY -
VOLID (VOL123)
/*
```

Figure 2-16 Initializing a volume

### ICKDSF program

ICKDSF is a program you can use to perform 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.

On modern DASD devices, there is no reason to run error detection and media maintenance, as these functions are supported internally by the controller. On the other hand, the concept of an MVS volume is not mapped into a unique physical DASD Redundant Access of Independent Disks (RAID) volume. Due to RAID, the MVS volume may be spread in several small disks, as in the case of RVA virtualization or today's ESS implementation. In the following examples, do not take in consideration the commands **ANALYZE** and **INSPECT** if you have a DASD more modern than a real 3390.

### Initializing a DASD volume

After you have completed the installation of a device, you must initialize and format the volume so that it can be used by MVS. If the volume is SMS-managed, the *STORAGEGROUP* option must be declared.

### VTOC and VTOC index

The **INIT** and **BUILDIX** commands will build the VTOC index. The **INIT** command creates space for the index during volume initialization in both operating system and stand-alone versions of ICKDSF. The **BUILDIX** command, which requires that the host operating system contains indexed VTOC programming support, builds VTOC indexes on volumes current in



use on the system. Both commands prepare the VTOC on the target volume to indexed VTOC (IXVTOC) format.

## INIT examples

Following are some examples of initializing volumes.

### *Initializing a volume for the first time in offline mode*

In this example, a volume is initialized at the minimal level because neither the **CHECK** nor **VALIDATE** parameter is specified. Because the volume is being initialized for the first time, it must be mounted offline, and the volume serial number must be specified. Because the **VTOC** parameter is not specified, the default volume table of contents size is the number of tracks in a cylinder minus one. For a 3390, the default is cylinder 0, track 1 for 14 tracks.

```
//EXAMPLE JOB
//          EXEC PGM=ICKDSF
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
INIT UNITADDRESS(0353) NOVERIFY VOLID(VOL123) -
      OWNERID(PAYROLL)
/*
```

### *Initializing a volume to be managed in a DFSMS environment*

In the following example, a volume that is to be system-managed is initialized. The volume is initialized in offline mode at the minimal level. The VTOC is placed at cylinder 2, track 1 and occupies ten tracks. The VTOC is followed by the VTOC index. The **STORAGEGROUP** parameter indicates the volume is to be managed in a DFSMS environment.

```
INIT UNIT(0353) NOVERIFY STORAGEGROUP -
      OWNERID(PAYROLL) VTOC(2,1,10) INDEX(2,11,5)
```

The following example performs an online minimal initialization, and as a result of the command, an index to the VTOC is created:

```
//          JOB
//          EXEC PGM=ICKDSF
//XYZ987   DD  UNIT=3390,DISP=OLD,VOL=SER=PAY456
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
INIT DDNAME(XYZ987) NOVERIFY INDEX('A','B','2')
/*
```

## ICKDSF stand-alone version

You can run the stand-alone version of ICKDSF under any IBM @server zSeries or IBM S/390 machine. To run the stand-alone version of ICKDSF, you IPL ICKDSF with a stand-alone IPL tape that you create under z/OS.

### **Creating an ICKDSF stand-alone IPL tape using z/OS**

For z/OS, the stand-alone code is in SYS1.SAMPLIB as ICKSADSF. You can load the ICKDSF program from a file on tape. The following example can be used to copy the stand-alone program to an unlabeled tape:

***Copy the stand-alone program to an unlabeled tape***

```
//JOBNAME JOB JOB CARD PARAMETERS
//STEPNAME EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY,DCB=BLKSIZE=80
//SYSUT1 DD DSNAME=SYS1.SAMPLIB(ICKSADSF),UNIT=SYSDA,
// DISP=SHR,VOLUME=SER=XXXXXX
//SYSUT2 DD DSNAME=ICKDSF,UNIT=3480,LABEL=(,NL),
// DISP=(,KEEP),VOLUME=SER=YYYYYY,
// DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
```

For details on how to IPL the stand-alone version and to see examples of the commands, refer to *Device Support Facilities User's Guide and Reference Release 17*, GC35-0033.

## 2.17 Problem determination

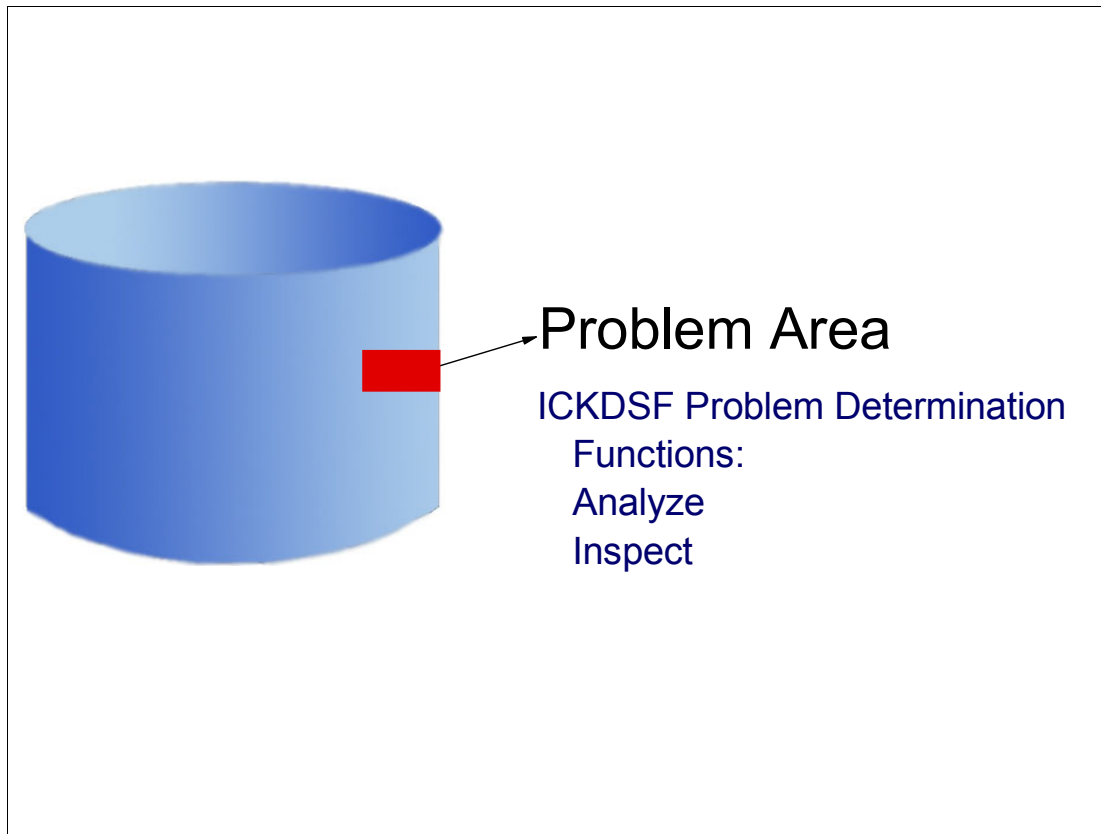


Figure 2-17 Problem determination using ICKDSF

### Using ICKDSF

You can use ICKDSF to help determine if the origin of a problem is hardware or recording media.

### Data check

A data check is an error detected in the bit pattern read from the disk. When it is a media problem, it is most likely caused by an imperfection on the disk surface.

### Analyze

The **ANALYZE** command helps to detect and differentiate recording surface and drive-related problems on a volume. It can also scan data to help detect possible media problems.

You can use the **ANALYZE** command to examine a device and the data on a volume, to help determine the existence and nature of errors.

You use two parameters with the **ANALYZE** command:

- ▶ **DRIVETEST** tests the hardware device.
- ▶ **SCAN** reads data on a volume.

You can use the **DRIVETEST** parameter to ensure that device hardware can perform basic operations, such as seeks, reads, and writes. **DRIVETEST** can impact your system performance, but does not alter data.

You can use **ANALYZE SCAN** to read data that currently exists on a volume. If **ANALYZE SCAN** reads the data successfully the first time, no further rereading of the track takes place.

### **INSPECT command**

The **INSPECT** command inspects a subset of a volume and can do the following:

- ▶ Check the surface of a track to determine if there is a defect
- ▶ Assign a skip to avoid a defect
- ▶ Assign an alternate track
- ▶ Reclaim a track that has been flagged defective
- ▶ Print a map of defective tracks on a volume

**Note:** Before using the **INSPECT** command, you should first make sure there are no hardware problems. It is recommended that you issue **ANALYZE DRIVETEST NOSCAN** before any **INSPECT** operation.

For more information about ICKDSF, refer to *Device Support Facilities User's Guide and Reference Release 17*, GC35-0033.



## Storage management hardware

The use of DFSMS requires storage management hardware that includes both Direct Access Storage Devices (DASD) and tape devices. In this chapter we provide an overview of both storage device categories, as well as a brief introduction to RAID technology.

For many years DASD devices have been the most used storage devices on IBM @server zSeries systems and their predecessors, delivering the fast access to data and high availability that customers have come to expect.

We cover the following types of DASD:

- ▶ Traditional DASD (such as 3380 and 3390)
- ▶ RAMAC Virtual Array (RVA)
- ▶ Enterprise Storage Server (ESS)

The era of tapes began before DASD was introduced. During that time, tapes were used as the primary storage medium. Today customers use tapes for such purposes as backup, archiving, or data transfer between companies.

We cover the following types of tape devices:

- ▶ Traditional tapes like 3480 and 3490
- ▶ IBM Magstar® 3590 and 3592
- ▶ Automated tape library (ATL) 3494
- ▶ Virtual tape server (VTS).

We also briefly explain the storage area network (SAN) concept.

## 3.1 Overview of DASD types

- ❑ Traditional DASD
  - 3380 Models J, E, K
  - 3390 Models 1, 2, 3, 9
- ❑ DASD based on RAID technology
  - RAMAC Array
  - RAMAC Virtual Array (RVA)
  - Enterprise Storage Server (ESS)
    - Seascape architecture

Figure 3-1 Overview of DASD types

### Traditional DASD

In the era of traditional DASD, the hardware consisted of controllers like 3880 and 3990 which contained the necessary functions to operate a storage subsystem. The controllers were connected to S/390 systems via parallel or ESCON® channels. Behind a controller you had several model groups of the 3390 which contained the disk drives. Based on the models, these disk drives had different capacities per device. Within each model group, the different 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.

### 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. We discuss the various RAID implementations in “Redundant array of independent disks (RAID)” on page 51.

The more modern IBM DASD products such as RAMACs, RVA, and Enterprise Storage Server (ESS), including DASD from other vendors, emulate IBM 3380 and 3390 volumes in the geometry, capacity of track, and number of tracks per cylinder. This emulation makes all the other entities think they are dealing with real 3380s or 3390s. Among these entities, we have data processing people not working directly with storage, JCL, MVS commands, open routines, access methods, IOS, channels. One advantage of this emulation is that it allows DASD manufacturers to implement changes in the 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 will always will be 3390s, sometimes with much bigger amount of cylinders, but a 3390.

### **ESS technology**

The IBM TotalStorage Enterprise Storage Server (ESS) is IBM's most powerful disk storage server, developed using IBM Seascape® architecture. The ESS provides unmatched functionality to the family of e-business servers, and also to non-IBM (that is, Intel®-based and UNIX-based) families of servers. Across all of these environments, the ESS features unique capabilities that allow it to meet the most demanding requirements of performance, capacity, and data availability that the computing business may require. See “Enterprise Storage Server (ESS)” on page 56 for more information about this topic.

### **Seascape architecture**

The Seascape architecture is the key to the development of IBM's storage products. Seascape allows IBM to take the best of the technologies developed by the many IBM laboratories and integrate them, producing flexible and upgradeable storage solutions. This Seascape architecture design has allowed the IBM TotalStorage Enterprise Storage Server to evolve from the initial E models to the succeeding F models, and to the later 800 models, each featuring new, more powerful hardware and functional enhancements, and always integrated under the same successful architecture with which the ESS was originally conceived. Refer to “Seascape architecture” on page 53 for more information.

**Note:** In this publication, we use 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.

## 3.2 Traditional DASD capacity

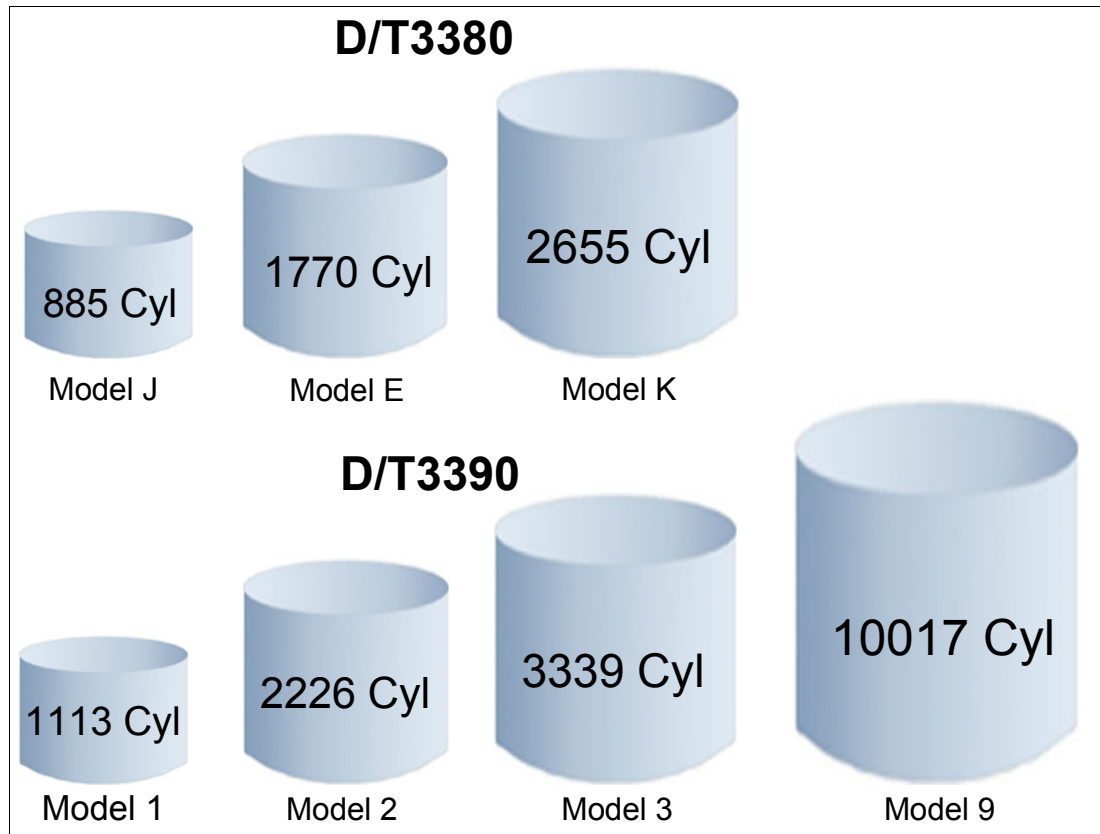


Figure 3-2 Traditional DASD capacity

### DASD capacity

Figure 3-2 shows various DASD device types. 3380 devices were used in the 1980s. Capacity went from 885 to 2,655 cylinders per volume. When storage density increased, new device types were introduced at the end of the 1980s. Those types were called 3390. Capacity per volume ranged from 1,113 to 3,339 cylinders. A special device type model 3390-9 was introduced, to store large amounts of data that needed very fast access. The track geometry within one device category was (and is) always the same; this means that 3380 volumes have 47,476 bytes per track, and 3390 volumes have 56,664 bytes per track.

Table 3-1 lists further information about DASD capacity.

Table 3-1 DASD capacity


Physical characteristics	3380-J	3380-E	3380-K	3390-1	3390-2	3390-3	3390-9
Data Cyl/Device	855	1770	2655	1113	2226	3339	10017
Track/Cyl	15	15	15	15	15	15	15
Bytes/Trk	47476	47476	47476	56664	56664	56664	56664
Bytes/Cylinder	712140	712140	712140	849960	849960	849960	849960
MB/Device	630	1260	1890	946	1892	2838	8514



### 3.3 Large Volume Support

- ❑ A "large volume" is larger than a 3390-9
- ❑ The largest possible volume has 32760 (3390) cylinders
- ❑ That would be a "3390-27" if it had its own device type

➤ Almost 28 GB



32760 Cyl

3390-27

Figure 3-3 Large volume support 3390-27

#### Large volume 3390-27

The IBM TotalStorage Enterprise Storage Server (ESS) initially supported custom volumes of up to 10017 cylinders, the size of the largest standard volume, the 3390 model 9. This was the limit set by the operating system software. The IBM TotalStorage ESS Large Volume Support (LVS) enhancement, announced in November 2001, has now increased the upper limit to 32760 cylinders, approximately 27.8 GB. The enhancement is provided as a combination of IBM TotalStorage ESS licensed internal code (LIC) changes and system software changes, available for z/OS, OS/390, and z/VM.

Large Volume Support (LVS) is available on z/OS and OS/390 operating systems, and the ICKDSF and DFSORT utilities.

Large Volume Support needs to be installed on all systems in a sysplex prior to sharing data sets on large volumes. Shared system/application data sets cannot be placed on large volumes until all system images in a sysplex have Large Volume Support installed.

#### LVS design considerations

Here are some considerations for large volume design:

- ▶ The S/390 I/O processor has an architectural device size limit of 32765 (x17FFD1) cylinders.
- ▶ The current software limit is 10017 cylinders.

- ▶ Records in sequential data sets are located using two-byte relative track addresses (TTR), which imposes a limit of 64 K tracks per data set.
- ▶ Control blocks such as the Data Extent Block (DEB) and channel commands such as SEEK use two bytes for cylinder and head addressing (CCHH) which imposes a limit of 64 K cylinders and tracks.

## 3.4 Redundant array of independent disks (RAID)

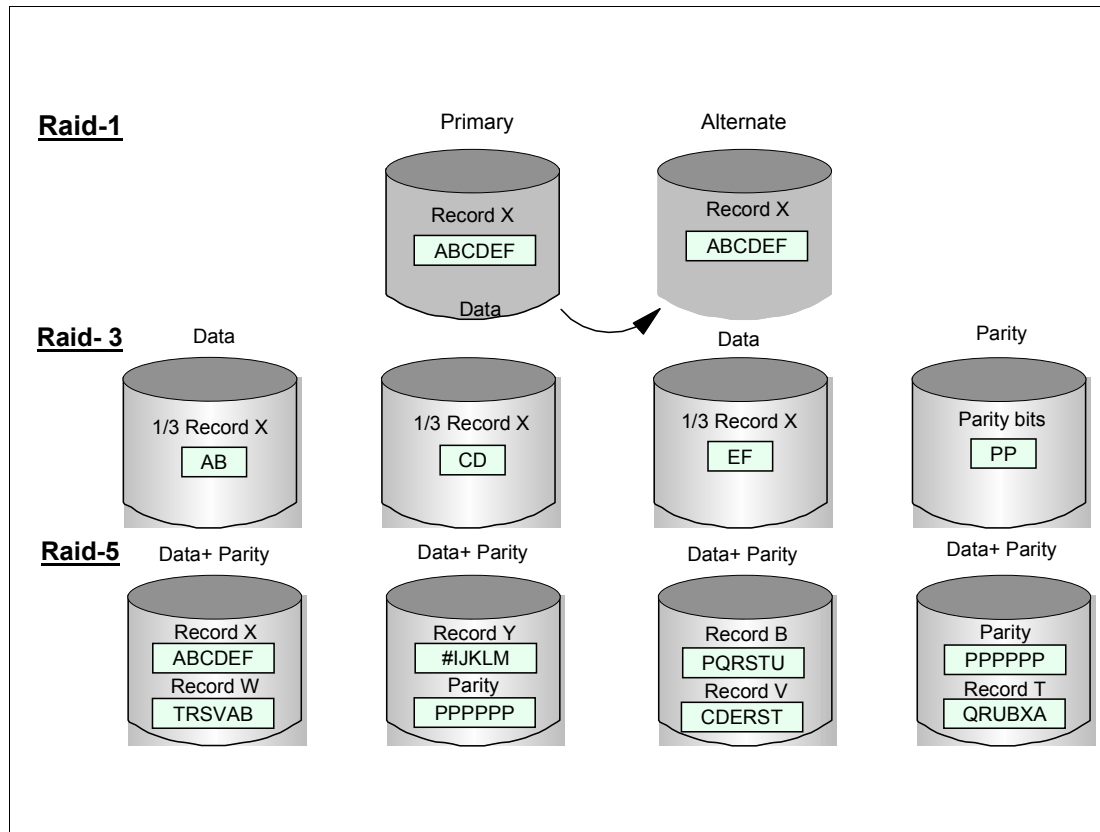


Figure 3-4 Redundant array of independent disks (RAID)

### RAID architecture

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.

The RAID concept involves many small computer system interface (SCSI) disks replacing a big one. The major RAID advantages are:

- ▶ Performance (due to parallelism)
- ▶ Cost (SCSI are commodities)
- ▶ zSeries compatibility
- ▶ Environment (space and energy)

However, RAID increased the chances of malfunction due to media and disk failures and the fact that the logical device is now residing 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.

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.

Various implementations certified by the RAID Architecture Board are:

- RAID-1** This has just disk mirroring, like dual copy.
- RAID-3** This has an array with one dedicated parity disk and just one I/O request at 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 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; that is, each physical block is written in each disk. The access arms move independently. It has strong caching to avoid write penalties; that is, four disk I/Os per write. RAID-5 has a high I/O rate and a medium data rate. RAID-5 is used by the IBM 2105 controller with 8-disk arrays in the majority of configurations.
- RAID-5 does the following:
- ▶ It reads data from an undamaged disk. This is just one, single disk I/O operation.
  - ▶ It reads data from a damaged disk, which implies (n-1) disk I/Os, to recreate 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 in order 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 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 is without write penalty (due to log-structured file, or LFS), and has background free-space reclamation. The access arms all move together for writes. It is used by the RVA controller.
- RAID-10** RAID-10 has a new RAID architecture designed to give performance for striping and has redundancy for mirroring. RAID-10 is optionally implemented in the IBM 2105.

**Note:** Data striping (stripe sequential physical blocks in different disks) is sometimes called RAID-0, but it is not a real RAID because of no redundancy, that is, no parity bits.

## 3.5 Seascope architecture

- ❑ Powerful storage server
- ❑ Snap-in building blocks
- ❑ Universal data access
  - Storage sharing
  - Data copy sharing
    - Network
    - Direct channel
    - Shared storage transfer
  - True data sharing

Figure 3-5 Seascope architecture

### Seascope architecture

Seascope is a storage enterprise architecture that is ideally suited to provide storage server solutions for the networked world. Seascope has three basic concepts:

- ▶ Powerful storage server
- ▶ Snap-in building blocks
- ▶ Universal data access

DFSMS provides device support for the IBM 2105 Enterprise Storage Server (ESS), a high-end storage subsystem. The ESS is the newest storage subsystem succeeding the 3880, 3990, and 9340 subsystem families. Designed for mid-range and high-end environments, the ESS gives you large capacity, high performance, continuous availability, and storage expandability. You can read more about ESS in “Enterprise Storage Server (ESS)” on page 56.

The ESS is also the first of the Seascope architecture storage products to attach directly to IBM @server zSeries and open-system platforms. The Seascope architecture products come with integrated storage controllers. These integrated storage controllers allow the attachment of physical storage devices that emulate 3390 Models 2, 3, and 9, or provide 3380 track-compatibility mode.

## Powerful storage server

It has a storage system which is intelligent and independent, and which can be reached by channels or via the network. This storage system is powered by a set of fast RISC processors.

## Snap-in building blocks

Each Seascope product is comprised of building blocks, such as:

- ▶ Scalable n-way RISC server, PCI-based; this provides the logic of the storage server.
- ▶ Memory cache from RISC processor memory.
- ▶ Channel attachments as FC-AL, SCSI, ESCON, FICON® and SSA.
- ▶ Network attachments, such as Ethernet, FDDI, TR, and ATM.

These attachments may also implement functions—a mix of network interfaces (to be used as a remote and independent storage server) and channel interfaces (to be used as a storage controller interface).

- ▶ Software building blocks, such as an AIX® subset, Java™ applications, and Tivoli® Storage Manager. High level language (HLL) is more flexible than microcode, and is easier to write and maintain.
- ▶ Storage adapters, for mixed storage devices technologies.
- ▶ Storage device building blocks, such as serial disk (7133), 3590 tape (Magstar), optical (3995).
- ▶ Silos and robots (3494).

## Universal data access

Universal data access allows a wide array of connectivity, such as z/OS, UNIX, Linux®, OS/400®, WIN, and OS/2®, to common data. There are three types of universal access:

- ▶ Storage sharing

Physical storage (DASD or tape) is statically divided into fixed partitions available to a given processor. It is not a software function. The subsystem controller knows which processors own which storage partitions. In a sense, only capacity is shared, not data; one server cannot access the data of the other server. It is required that the manual reassignment of storage capacity between partitions be simple and nondisruptive.

The advantages are:

- Purchase higher quantities with greater discounts
- Only one type of storage to manage
- Static shifting of capacity as needed

The drawbacks are:

- Higher price for SCSI data
- Collocation at 20 meters of the SCSI servers
- No priority concept between z/OS and UNIX/NT I/O requests

- ▶ Data copy sharing

Data copy sharing is an interim data replication solution (waiting for a true data sharing) done via data replication from one volume accessed by a platform to another volume accessed by another platform. The replication can be done through software or hardware.

There are three ways to implement data copy sharing:

- Network: Via network data transfer as SNA or TCP/IP.

However, this method has drawbacks such as CPU and network overhead; still slow and expensive for massive data transfer.

- Direct channel: Direct data transfer between the processors involved using channel or bus capabilities, referred to as *bulk data transfer*.

One example of this is IBM Infospeed®, wherein a /390 FTP (Press Data Mover, or PDM) at 28 MB/sec extract utility program writes data at 28 MB/sec to a pipe (Infospeed box) that is concurrently being read by UNIX and NT.

- Shared storage transfer: Writing an intermediate flat file by software into the storage subsystem cache, that is read (and translated) by the receiving processor, so the storage is shared.

- ▶ True data sharing

For data sharing between multiple platforms for read/write of a single copy that addresses the complex issues of mixed data types, file structures, databases, and SCPs, there is no available solution.

## 3.6 Enterprise Storage Server (ESS)

- ❑ Two 6-way RISC processors (668 MHZ)
- ❑ 4.8 GB/sec of aggregate bandwidth
- ❑ Up to 32 ESCON / SCSI / mixed
- ❑ Up to 16 FICON and FCP channels
- ❑ Up to 64 GB of cache
- ❑ 2 GB of NVS cache
- ❑ 18.2 / 36.4 / 72.8 / 145.6 GB capacity disk options
- ❑ Up to 55.9 TB capacity
- ❑ 8 x 160 MB/sec SSA loops
- ❑ 10,000 rpm and 15,000 rpm disk options
- ❑ Connects to SAN
- ❑ RAID-5 or RAID-10



Figure 3-6 Enterprise Storage Server model 800

### Enterprise Storage Server (ESS)

The IBM Enterprise Storage Server (ESS) is a high performance, high-availability capacity storage subsystem. It contains up to two six-way RISC processors (668 MHZ) with up to 64 GB cache and 2 GB of non-volatile storage (NVS) to protect from data loss along power outages. Connectivity to zSeries is through up to 32 ESCON channels and up to 16 FICON channels. For other platforms such as IBM @server iSeries™, UNIX, or NT, the connectivity is through up to 32 SCSI interfaces.

### Cache

Cache is used to store both read and write data to improve ESS performance to the attached host systems. There is the choice of 8, 16, 24, 32 or 64 GB of cache. This cache is divided between the two clusters of the ESS, giving the clusters their own non-shared cache. The ESS cache uses ECC (error checking and correcting) memory technology to enhance reliability and error correction of the cache. ECC technology can detect single- and double-bit errors and correct all single-bit errors. Memory scrubbing, a built-in hardware function, is also performed and is a continuous background read of data from memory to check for correctable errors. Correctable errors are corrected and rewritten to cache. To protect against loss of data on a write operation, the ESS stores two copies of written data, one in cache and the other in NVS.



## **NVS cache**

NVS is used to store a second copy of write data to ensure data integrity, should there be 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, in the event of a cluster failure, the write data for the failed cluster will be in the NVS of the surviving cluster. This write data is then destaged 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 ensures that no data is lost even in the event of a component failure.

## **ESS Model 800**

The ESS Model 800 has a 2 GB NVS. Each cluster has 1 GB of NVS, made up of four cards. Each pair of NVS cards has its own battery-powered charger system that protects data even if power is lost on the entire ESS for up to 72 hours. This model has the following enhancements:

- ▶ Model 800 allows 4.8 GB/sec of aggregate bandwidth.
- ▶ In the disk interface the ESS has eight Serial Storage Architecture (SSA) loops, each one with a rate of 160 MB/sec for accessing the disks. See “SSA loops” on page 68 for more information about this topic.
- ▶ ESS implements RAID-5 or RAID-10 for availability and has eight disks in the majority of the arrays. See “RAID-10” on page 70 for more information about this topic.
- ▶ Four sizes disks of 18.2, 36.4, 72.8, and 145.6 GB, which can be intermixed. The ESS maximum capacity is over 55.9 TB with a second frame attached.

## **SCSI protocol**

Although we do not cover other platforms in this publication, we provide here a brief overview of the SCSI protocol. The SCSI adapter is a card in the host. It connects to a SCSI bus via an SCSI port. There are two different types of SCSI supported by ESS:

- ▶ SCSI Fast Wide with 20 MB/sec
- ▶ Ultra SCSI Wide with 40 MB/sec

Comparing the terminology of ESCON and SCSI, we may say that:

- ▶ An ESCON channel translates into an SCSI adapter (both cards in the host)
- ▶ An ESCON port translates into an SCSI port (both connectors)
- ▶ An ESCON link translates into an SCSI bus (both cables)

## 3.7 ESS universal access

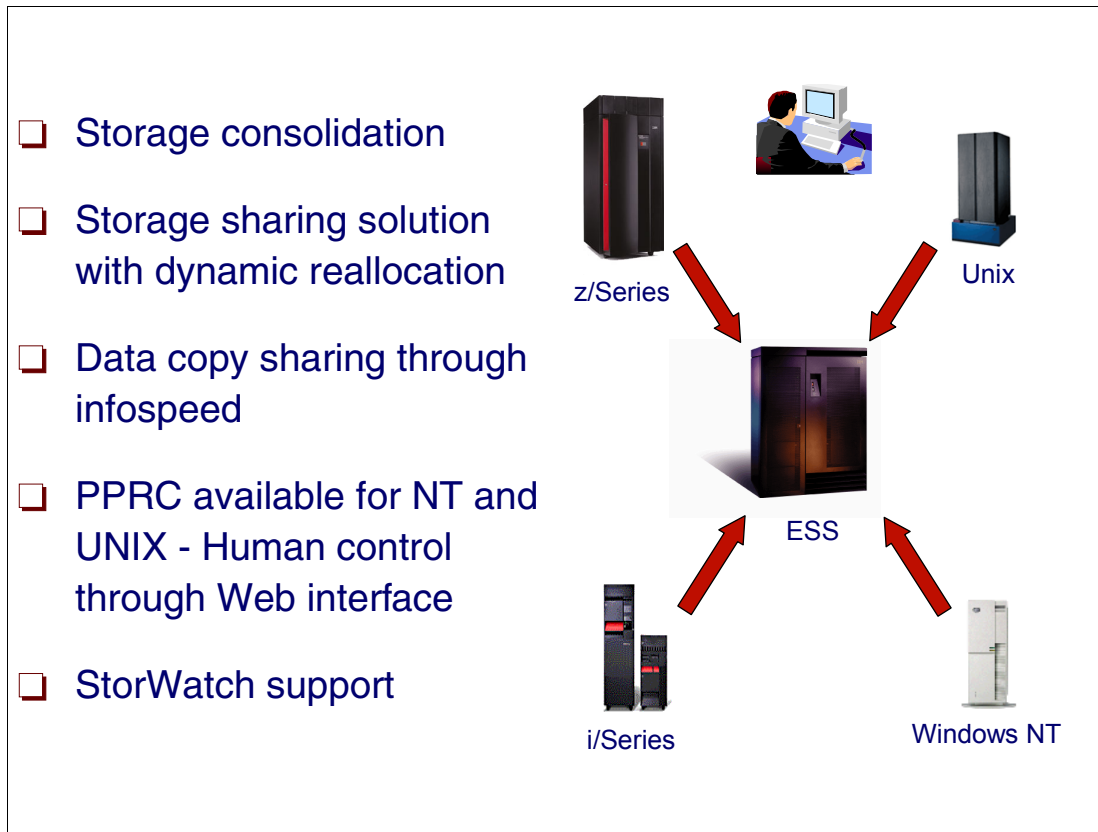


Figure 3-7 ESS universal access

### ESS universal access

ESS is a product designed to implement *storage* consolidation that puts all of your enterprise data under the same cover. This consolidation is the first step in achieving *server* consolidation—that is, to put all of your enterprise applications under the same z/OS cluster.

Thinking in Seascope terms about universal access, the ESS box implements storage sharing with dynamic reallocation and data copy sharing using Infospeed.

Many of the ESS features are now available to non-zSeries platforms such as PPRC for NT and UNIX, where the control is through a Web interface

In the software side, there is StorWatch, a range of products in UNIX/NT that does what DFSMS and automation do for zSeries.

In “Operating systems supporting ESS” on page 59, you will see all the operating systems able to access data in the ESS box in storage sharing mode (physical partition), that is, one device accessed by just one heterogeneous operating system image.

## 3.8 Operating systems supporting ESS

- ❑ AIX 4.2.1 and above
- ❑ OS/400 V3R1 and above
- ❑ HP UNIX 10.20 and above
- ❑ Sun Solaris 2.5.1 and above
- ❑ Windows NT Server 4.0 and above
- ❑ Data General DG/UX 4.2 and above
- ❑ Novell Netware 4.2 and above
- ❑ Various Linux releases on several platforms
- ❑ For an up-to-date list, check:
  - [www.ibm.com/storage](http://www.ibm.com/storage)

*Figure 3-8 Operating systems supporting ESS*

### **Operating systems supporting ESS**

ESS is supported by the open operating systems listed in Figure 3-8. The list contains those systems able to access data in the ESS box at general availability time.

## 3.9 ESS major components

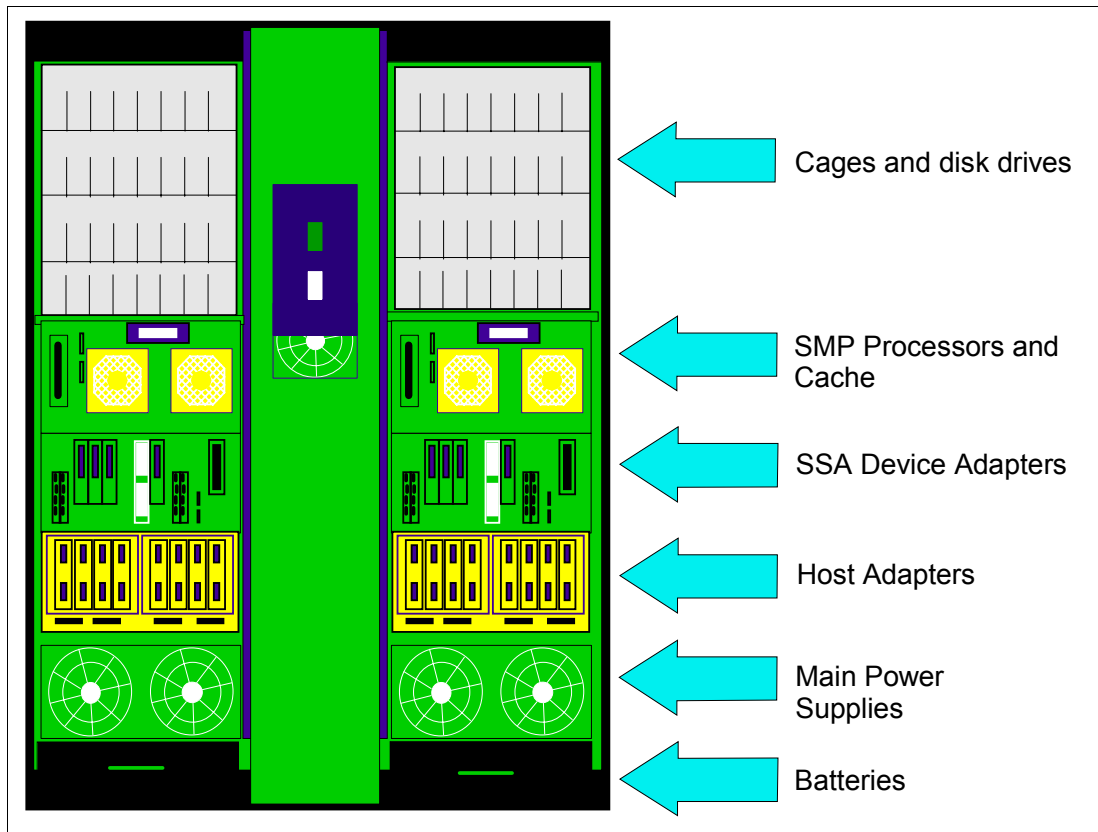


Figure 3-9 ESS major components

### ESS Model 800 major components

Figure 3-9 shows an IBM TotalStorage Enterprise Storage Server Model 800 and its major components. As you can see, the ESS base rack consists of two clusters, each with its own power supplies, batteries, SSA device adapters, processors, cache and NVS, CD drive, hard disk, floppy disk and network connections. Both clusters have access to any host adapter card, even though they are physically spread across the clusters.

At the top of each cluster is an ESS cage. Each cage provides slots for up to 64 disk drives, 32 in front and 32 at the back.

This storage box has two enclosures:

- ▶ A base enclosure, with:
  - Two 3-phase power supplies
  - Up to 128 disk drives in two cages
  - Feature #2110 for Expansion Enclosure attachment
  - Host adapters, cluster processors, cache and NVS, and SSA device adapters
- ▶ An Expansion Enclosure, with:
  - Two 3-phase power supplies
  - Up to 256 disk drives in four cages for additional capacity

## 3.10 ESS host adapters

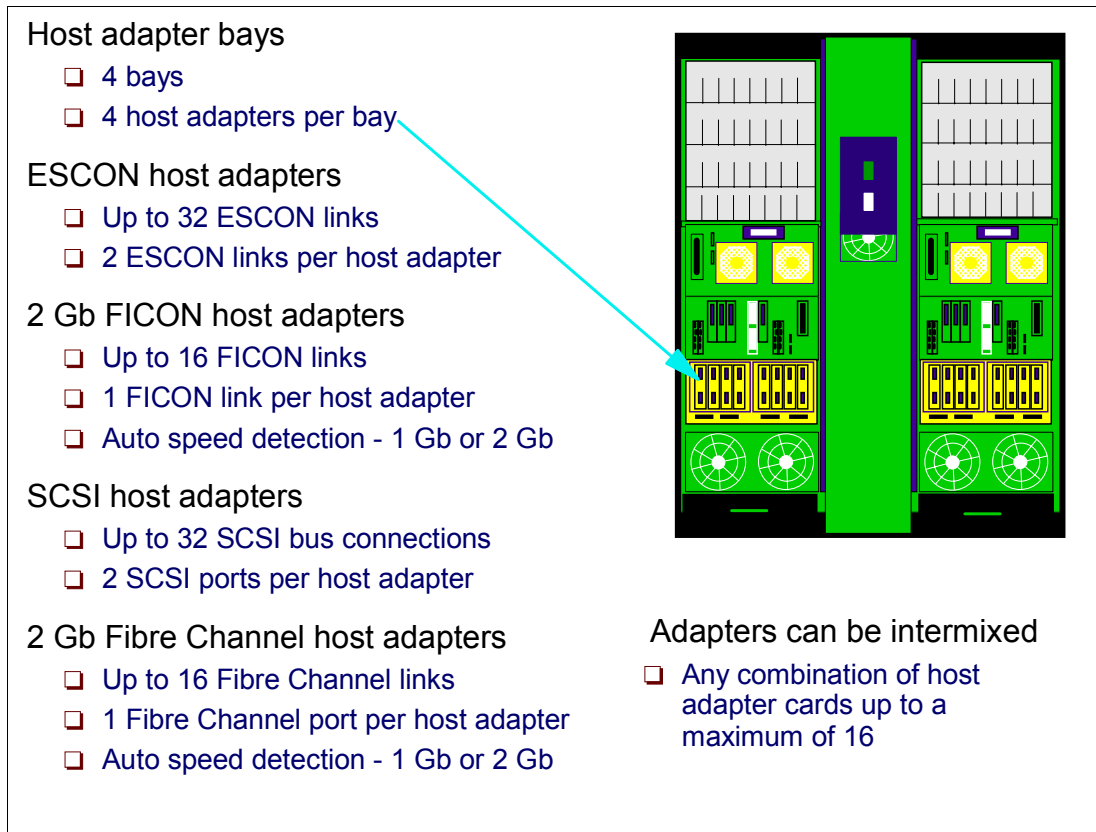


Figure 3-10 Host adapters

### ESS host adapters

The ESS has four host adapter (HA) bays, two in each cluster. Each bay supports up to four host adapter cards. Each of these host adapter cards can be for FICON, ESCON, SCSI, or Fibre Channel server connection. Figure 3-10 lists the main characteristics of the ESS host adapters.

Each host adapter can communicate with either cluster. To install a new host adapter card, the bay must be powered off. For the highest path availability, it is important to spread the host connections across all the adapter bays. For example, if you have four ESCON links to a host, each connected to a different bay, then the loss of a bay for upgrade would only impact one out of four of the connections to the server. The same would be valid for a host with FICON connections to the ESS.

Similar considerations apply for servers connecting to the ESS by means of SCSI or fibre channel links. For open system servers, the Subsystem Device Driver (SDD) program that comes standard with the ESS can be installed on the connecting host servers to provide multiple paths or connections to handle errors (path failover) and balance the I/O load to the ESS.

The ESS connects to a large number of different servers, operating systems, host adapters, and SAN fabrics. A complete and current list is available at the following Web site:

<http://www.storage.ibm.com/hardsoft/products/ess/supserver.htm>

## 3.11 FICON host adapters

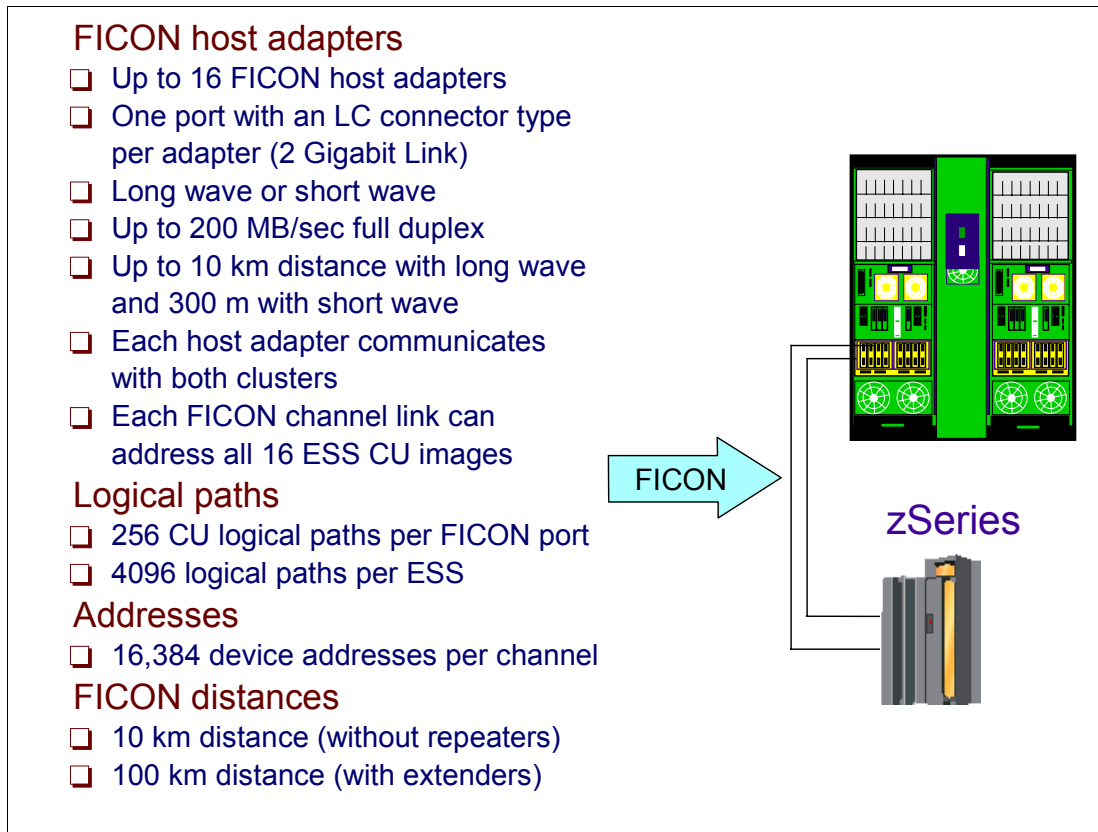


Figure 3-11 FICON host adapter

### FICON host adapters

FICON (Fiber Connection) is based on the standard Fibre Channel architecture, and therefore shares the attributes associated with Fibre Channel. This includes the common FC-0, FC-1, and FC-2 architectural layers, the 100 MBps bidirectional (full-duplex) data transfer rate, and the point-to-point distance capability of 10 kilometers. The ESCON protocols have been mapped to the FC-4 layer, the Upper Level Protocol (ULP) layer, of the Fibre Channel architecture. All this provides a full-compatibility interface with previous S/390 software and puts the zSeries servers in the Fibre Channel industry standard.

### FICON versus ESCON

FICON goes beyond ESCON limits:

- ▶ Addressing limit, from 1024 device addresses per channel to up to 16,384 (maximum of 4096 devices supported within one ESS).
- ▶ Up to 256 control unit logical paths per port.
- ▶ FICON channel to ESS allows multiple concurrent I/O connections (the ESCON channel supports only one I/O connection at one time).
- ▶ Greater channel and link bandwidth: FICON has up to 10 times the link bandwidth of ESCON (1 Gbps full-duplex, compared to 200 MBps half duplex). FICON has up to more than four times the effective channel bandwidth.
- ▶ FICON path consolidation using switched point-to-point topology.

- ▶ Greater unrepeated fiber link distances (from 3 km for ESCON to up to 10 km, or 20 km with an RPQ, for FICON).

These characteristics allow simpler and more powerful configurations. The ESS supports up to 16 host adapters, which allows for a maximum of 16 Fibre Channel/FICON ports per machine, as shown in Figure 3-11 on page 62.

Each Fibre Channel/FICON host adapter provides one port with an LC connector type. The adapter is a 2 Gb card and provides a nominal 200 MBps full-duplex data rate. The adapter will auto-negotiate between 1 Gb and 2 Gb, depending upon the speed of the connection at the other end of the link. For example, from the ESS to a switch/director, the FICON adapter can negotiate to 2 Gb if the switch/director also has 2 Gb support. The switch/director to host link can then negotiate at 1 Gb.

### **Host adapter cards**

There are two types of host adapter cards you can select: long wave (feature 3024), and short wave (feature 3025). With long-wave laser, you can connect nodes at distances of up to 10 km (without repeaters). With short-wave laser, you can connect distances of up to 300 m. These distances can be extended using switches/directors.

## 3.12 ESS disks

- ❑ **Eight-packs**
  - Set of 8 similar capacity/rpm disk drives packed together
  - Installed in the ESS cages
  - Initial minimum configuration is 4 eight-packs
  - Upgrades are available increments of 2 eight-packs
  - Maximum of 48 eight-packs per ESS with expansion
- ❑ **Disk drives**
  - 18.2 GB 15,000 rpm or 10,000 rpm
  - 36.4 GB 15,000 rpm or 10,000 rpm
  - 72.8 GB 10,000 rpm
  - 145.6 GB 10,000 rpm
- ❑ **Eight-pack conversions**
  - Capacity and/or RPMs

Figure 3-12 ESS disks

### ESS disks

With a number of disk drive sizes and speeds available, including intermix support, the ESS provides a great number of capacity configuration options.

The maximum number of disk drives supported within the IBM TotalStorage Enterprise Storage Server Model 800 is 384—with 128 disk drives in the base enclosure and 256 disk drives in the expansion rack. When configured with 145.6 GB disk drives, this gives a total physical disk capacity of approximately 55.9 TB (see Table 3-2 on page 65 for more details).

### Disk drives

The minimum available configuration of the ESS Model 800 is 582 GB. This capacity can be configured with 32 disk drives of 18.2 GB contained in four eight-packs, using one ESS cage. All incremental upgrades are ordered and installed in pairs of eight-packs; thus the minimum capacity increment is a pair of similar eight-packs of either 18.2 GB, 36.4 GB, 72.8 GB, or 145.6 GB capacity.

The ESS is designed to deliver substantial protection against data corruption, not just relying on the RAID implementation alone. The disk drives installed in the ESS are the latest state-of-the-art magneto resistive head technology disk drives that support advanced disk functions such as disk error correction codes (ECC), Metadata checks, disk scrubbing and predictive failure analysis.



## Eight-pack conversions

The ESS eight-pack is the basic unit of capacity within the ESS base and expansion racks. As mentioned before, these eight-packs are ordered and installed in pairs. Each eight-pack can be configured as a RAID 5 rank (6+P+S or 7+P) or as a RAID 10 rank (3+3+2S or 4+4).

The IBM TotalStorage ESS Specialist will configure the eight-packs on a loop with spare DDMs as required. Configurations that include drive size intermixing may result in the creation of additional DDM spares on a loop as compared to non-intermixed configurations. Currently there is the choice of four different new-generation disk drive capacities for use within an eight-pack:

- ▶ 18.2 GB/15,000 rpm disks
- ▶ 36.4 GB/15,000 rpm disks
- ▶ 72.8 GB/10,000 rpm disks
- ▶ 145.6 GB/10,000 rpm disks

Also available is the option to install eight-packs with:

- ▶ 18.2 GB/10,000 rpm disks or
- ▶ 36.4 GB/10,000 rpm disks

The eight disk drives assembled in each eight-pack are all of the same capacity. Each disk drive uses the 40 MBps SSA interface on each of the four connections to the loop.

It is possible to mix eight-packs of different capacity disks and speeds (rpm) within an ESS, as described in the following sections.

Table 3-2 should be used as a guide for determining the capacity of a given eight-pack. This table shows the capacities of the disk eight-packs when configured as RAID ranks. These capacities are the *effective capacities* available for user data.

Table 3-2 Disk eight-pack effective capacity chart (gigabytes)

Disk Size	Physical Capacity (raw capacity)	Effective usable capacity (2)			
		RAID 10		RAID 5 (3)	
		3 + 3 + 2S Array (4)	4 + 4 Array (5)	6+P+S Array (6)	7 + P Array (7)
18.2	145.6	52.50	70.00	105.20	122.74
36.4	291.2	105.12	140.16	210.45	245.53
72.8	582.4	210.39	280.52	420.92	491.08
145.6	1,164.8	420.78	561.04	841.84	982.16

### 3.13 Device adapters

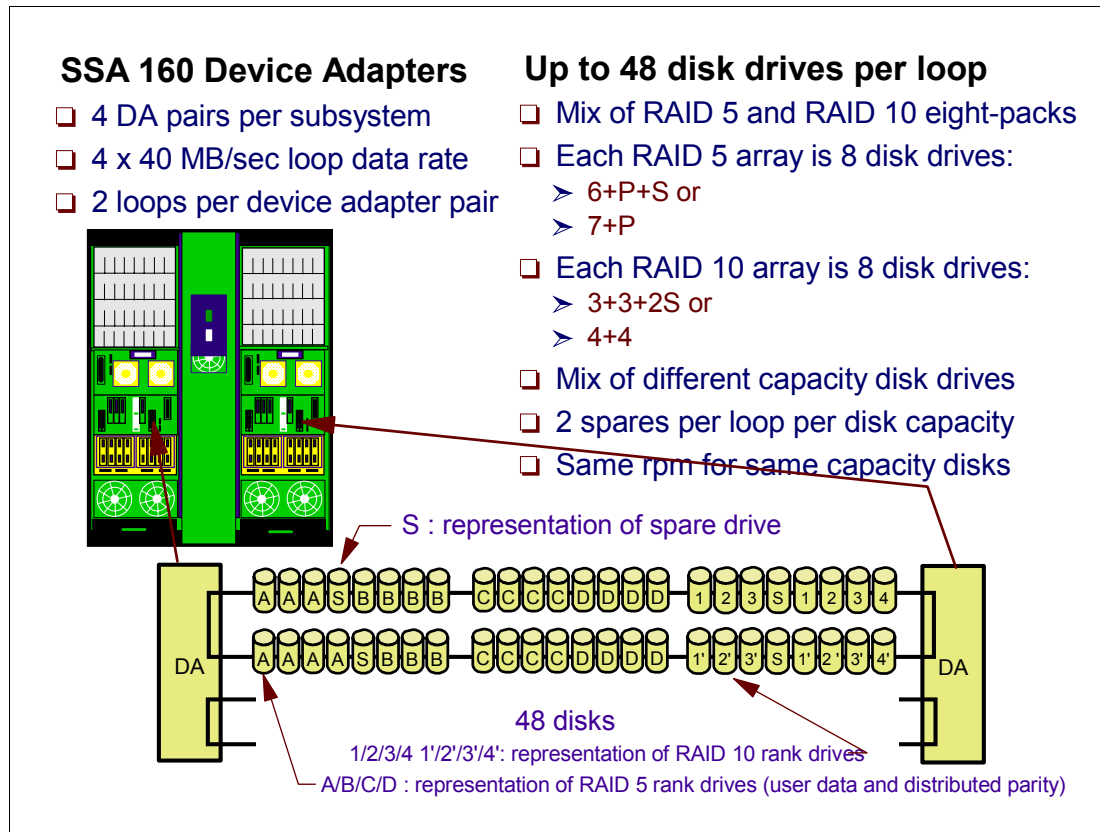


Figure 3-13 Device adapters

#### ESS device adapters

Device adapters (DA) provide the connection between the clusters and the disk drives. The ESS Model 800 implements faster Serial Storage Architecture (SSA) device adapters than its predecessor models.

The ESS Storage Server Model 800 uses the latest SSA160 technology in its device adapters (DA). With SSA 160, each of the four links operates at 40 MBps, giving a total nominal bandwidth of 160 MBps for each of the two connections to the loop. This amounts to a total of 320 MBps across each loop. Also, each device adapter card supports two independent SSA loops, giving a total bandwidth of 320 MBps per adapter card. There are eight adapter cards, giving a total nominal bandwidth capability of 2,560 MBps. Refer to “SSA loops” on page 68 for more information about this topic.

#### SSA loops

One adapter from each pair of adapters is installed in each cluster as shown in Figure 3-13. The SSA loops are between adapter pairs, which means that all the disks can be accessed by both clusters. During the configuration process, each RAID array is configured by the IBM TotalStorage ESS Specialist to be normally accessed by only one of the clusters. Should a cluster failure occur, the remaining cluster can take over all the disk drives on the loop.

#### RAID 5 and RAID 10

RAID 5 and RAID 10 are managed by the SSA device adapters. RAID 10 is explained in detail in “RAID-10” on page 70. Each loop supports up to 48 disk drives, and each adapter

pair supports up to 96 disk drives. There are four adapter pairs supporting up to 384 disk drives in total. Figure 3-13 shows a logical representation of a single loop with 48 disk drives (RAID ranks are actually split across two eight-packs for optimum performance). You can see there are six RAID arrays: four RAID 5 designated A to D, and two RAID 10 (one 3+3+2 spare and one 4+4).

### **Disk drives per loop**

Each loop supports up to 48 disk drives, and each adapter pair supports up to 96 disk drives. There are four adapter pairs supporting up to 384 disk drives in total.

Figure 3-13 on page 66 shows a logical representation of a single loop with 48 disk drives (RAID ranks are actually split across two eight-packs for optimum performance). In the figure you can see there are six RAID arrays: four RAID 5 designated A to D, and two RAID 10 (one 3+3+2 spare and one 4+4).

## 3.14 SSA loops

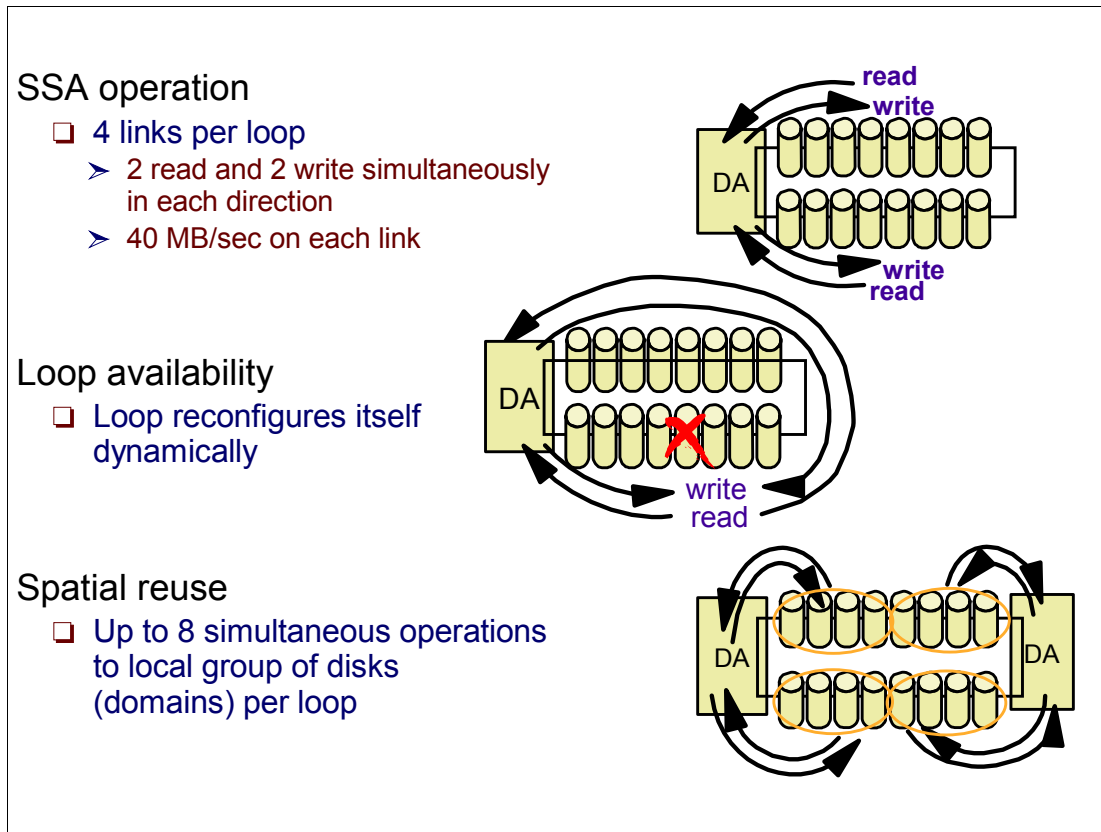


Figure 3-14 SSA loops

### SSA operation

SSA is a high performance, serial connection technology for disk drives. SSA is a full-duplex loop-based architecture, with two physical read paths and two physical write paths to every disk attached to the loop. Data is sent from the adapter card to the first disk on the loop and then passed around the loop by the disks until it arrives at the target disk. Unlike bus-based designs, which reserve the whole bus for data transfer, SSA only uses the part of the loop between adjacent disks for data transfer. This means that many simultaneous data transfers can take place on an SSA loop, and it is one of the main reasons that SSA performs so much better than SCSI. This simultaneous transfer capability is known as “spatial release”.

Each read or write path on the loop operates at 40 MB/s, providing a total loop bandwidth of 160 MB/s.

### Loop availability

The loop is a self-configuring, self-repairing design that allows genuine hot-plugging. If the loop breaks for any reason, then the adapter card will automatically reconfigure the loop into two single loops. In the ESS, the most likely scenario for a broken loop is if the actual disk drive interface electronics should fail. If this should happen, the adapter card will dynamically reconfigure the loop into two single loops, effectively isolating the failed disk. If the disk were part of a RAID array, the adapter card would automatically regenerate the missing disk using the remaining data and parity disks to the spare disk. Once the failed disk has been replaced, the loop will automatically be reconfigured into full duplex operation, and the replaced disk will become a new spare.

## **Spatial reuse**

Spatial reuse allows domains to be set up on the loop. A *domain* means that one or more groups of disks belong to one of the two adapter cards, as is the case during normal operation. The benefit of this is that each adapter card can talk to its domains (or disk groups) using only part of the loop. The use of domains allows each adapter card to operate at maximum capability because it is not limited by I/O operations from the other adapter. Theoretically, each adapter card could drive its domains at 160 MB/s, giving 320 MB/s throughput on a single loop! The benefit of domains may reduce slightly over time, due to disk failures causing the groups to become intermixed, but the main benefits of spatial reuse will still apply.

If a cluster should fail, the remaining cluster device adapter will own all the domains on the loop, thus allowing full data access to continue.

## 3.15 RAID-10

- ❑ **RAID-10 configurations:**
  - First RAID-10 rank configured in the loop will be:  $3 + 3 + 2S$
  - Additional RAID-10 ranks configured in the loop will be  $4 + 4$
  - For a loop with an intermixed capacity, the ESS will assign two spares for each capacity. This means there will be one  $3+3+2S$  array per capacity

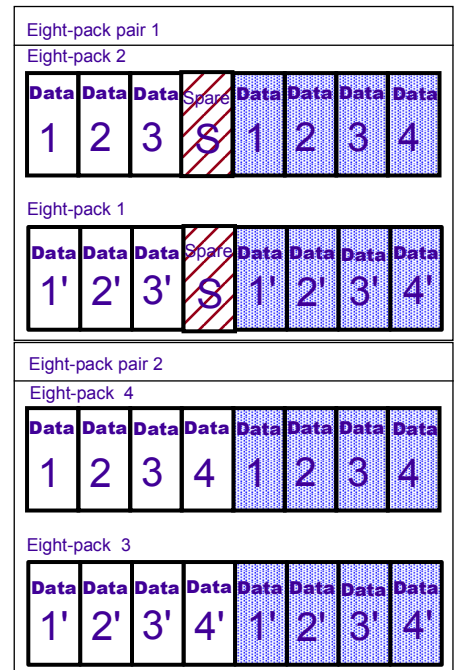


Figure 3-15 RAID-10

### RAID-10

RAID-10 is also known as RAID 0+1, 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 (in the ESS Model 800 implementation, three or four DDMs). RAID 1 is the protection against a disk failure by having a mirrored copy of each disk. By combining the two, RAID 10 provides data protection with I/O performance.

### Array

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 ESS, the arrays are built upon the disks of the disk eight-packs.

### Disk eight-pack

The physical storage capacity of the ESS is materialized by means of the disk eight-packs. These are sets of eight DDMs that are installed in pairs in the ESS. Two disk eight-packs provide for two disk groups—four DDMs from each disk eight-pack. These disk groups can be configured as either RAID-5 or RAID-10 ranks.

### Spare disks

The ESS requires that a loop have a minimum of two spare disks to enable sparing to occur. The sparing function of the ESS is automatically initiated whenever a DDM failure is detected on a loop and enables regeneration of data from the failed DDM onto a hot spare DDM.

A hot DDM *spare pool* consisting of two drives, created with one 3+3+2S array (RAID 10), is created for each drive size on an SSA loop. Therefore, if only one drive size is installed on a loop, only two spares are required. The hot sparing function is managed at the SSA loop level. SSA will spare to a larger capacity DDM on the loop in the very uncommon situation that no spares are available on the loop for a given capacity.

Figure 3-15 on page 70 shows the following:

1. In eight-pack pair 1, the array consists of three data drives mirrored to three copy drives. The remaining two drives are used as spares.
2. In eight-pack pair 2, the array consists of four data drives mirrored to four copy drives.

## 3.16 Storage balancing with RAID-10

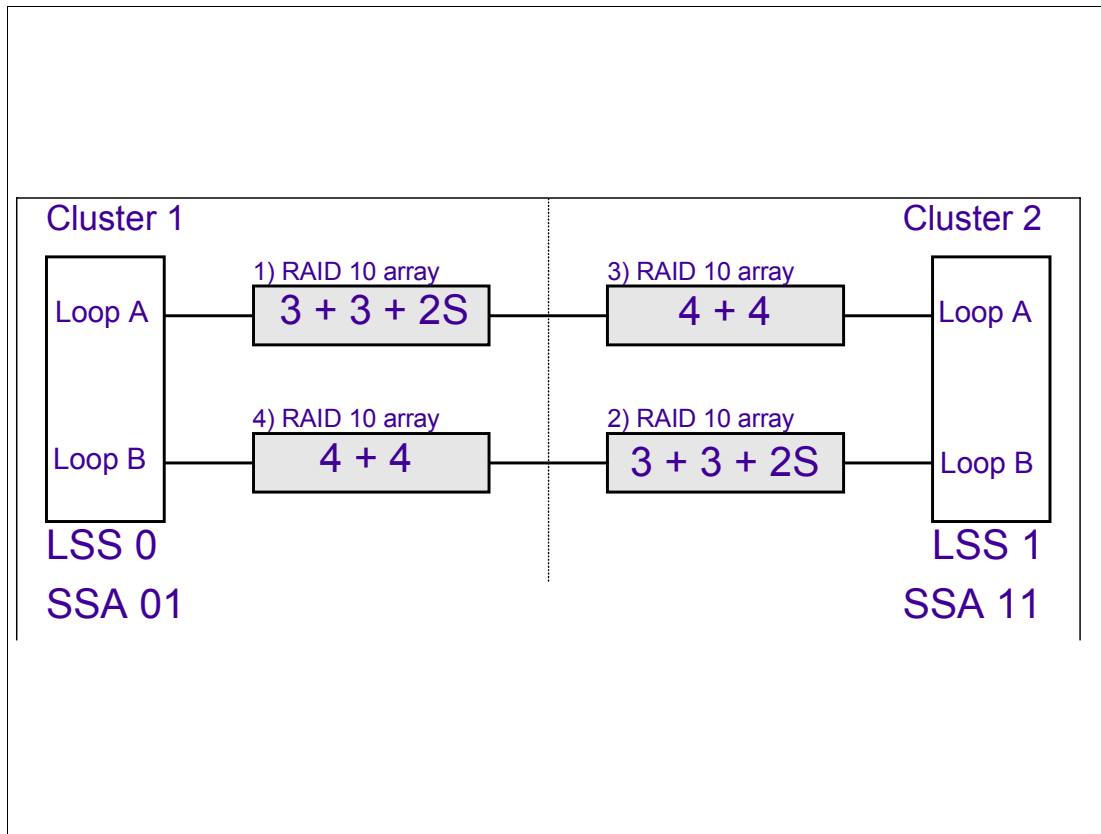


Figure 3-16 Storage balancing with RAID-10

### Logical Storage Subsystem (LSS)

The Logical Storage Subsystem (or Logical Subsystem) is a logical structure that is internal to the ESS. It is a logical construct that groups up to 256 logical volumes (logical volumes are defined during the logical configuration procedure) of the same disk format (CKD or FB), and it is identified by the ESS with a unique ID. Although the LSS relates directly to the logical control unit (LCU) concept of the ESCON and FICON architectures, it does not directly relate to SCSI and FCP addressing.

### ESS storage balancing

For performance reasons you should try to allocate storage on the ESS equally balanced across both clusters and among the SSA loops. One way to accomplish this is to assign two arrays (one from loop A and one from loop B) to each Logical Subsystem. To achieve this you can follow this procedure when configuring RAID-10 ranks:

1. Configure the first array for LSS 0/loop A. This will be a 3 + 3 + 2S array.
2. Configure the first array for LSS 1/loop B. This will also be a 3 + 3 + 2S array.
3. Configure the second array for LSS1/loop A. This will now be a 4 + 4.
4. Configure the second array for LSS 0/loop B. This will also now be a 4 + 4.

Figure 3-16 on page 72 illustrates the results of this configuration procedure.



## 3.17 ESS performance features

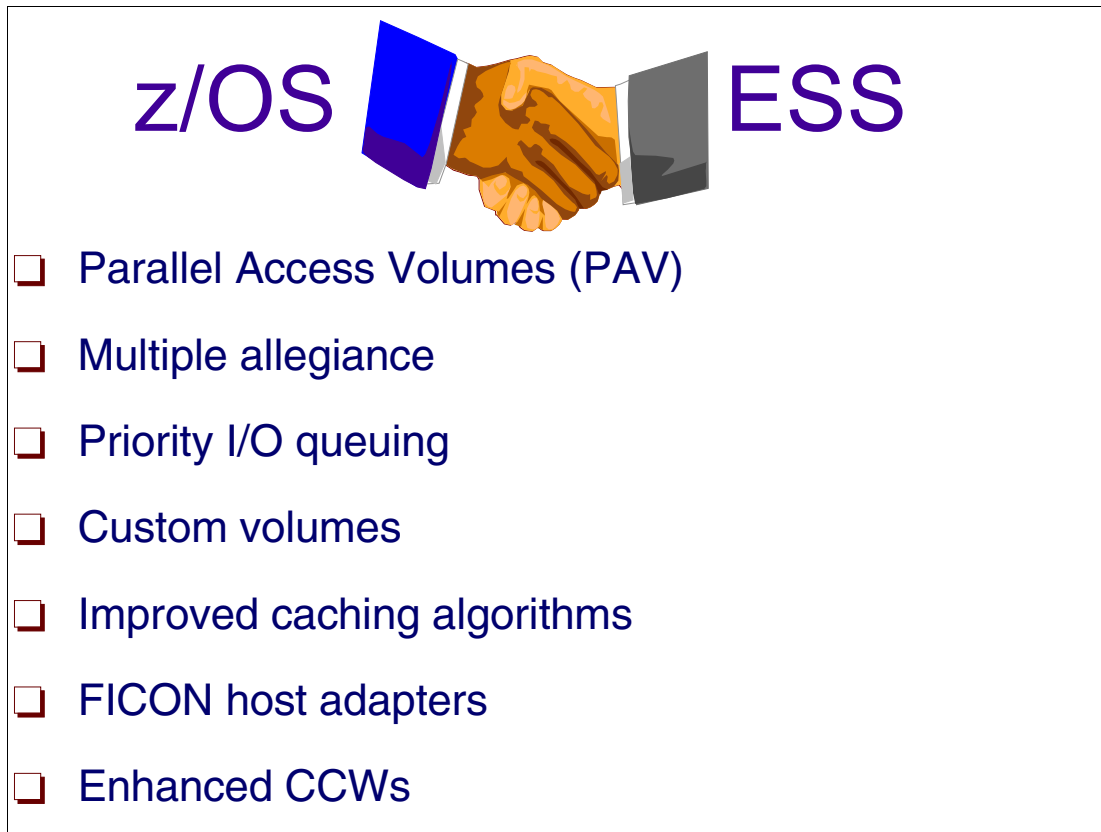


Figure 3-17 ESS performance features

### Parallel access volumes (PAV) and multiple allegiance

Traditional 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 time. However, in modern DASD subsystems like ESS, the device (such as a 3390) is only a logical view. The contents of this logical device are spread in HDA RAID arrays and in caches. Therefore, it is technically possible to have more than one I/O operation towards the same logical device.

Changes are made in z/OS (in IOS code), in channel subsystem (SAP), and in ESS in order to allow more than one I/O operation on the same logical device. This is called parallel I/O, and has two flavors:

- ▶ Parallel Access Volume (PAV), when the concurrent I/Os originate from the same z/OS image
- ▶ Multiple Allegiance, when the concurrent I/Os originate from different z/OS images

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.

To implement PAV, the IOS introduces the concept of *alias addresses*. Instead of one UCB per logical volume, an MVS host can now use several UCBs for the same logical volume. Apart from the conventional Base UCB, alias UCBs can be defined and used by z/OS to issue I/Os in parallel to the same logical volume device.

## **I/O priority queueing**

Prior to ESS, 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 WLM, depending on the transaction owning the I/O request. There was no concept of priority queueing within the internal queues of the I/O control units; instead, the queue regime was FIFO.

With ESS, it is possible to have this queue concept internally; I/O Priority Queueing in ESS has the following properties:

- ▶ I/O can be queued with the ESS 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.

## **Custom volumes**

Custom volumes provides the possibility of defining small size 3390 or 3380 volumes. This causes less contention on a volume. Custom volumes is designed for high activity data sets. Careful size planning is required.

## **Improved caching algorithms**

With its effective caching algorithms, IBM TotalStorage Enterprise Storage Server Model 800 is able to minimize wasted cache space and reduce disk drive utilization, thereby reducing its back-end traffic. The ESS Model 800 has a maximum cache size of 64 GB, and the NVS standard size is 2 GB.

The ESS manages 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 could exhaust cache capacity while filling up with small random reads. Thus the ESS, having smaller cache segments, is able to avoid wasting cache space for situations of small record sizes that are common in interactive applications.

This efficient cache management, together with the ESS Model 800 powerful back-end implementation that integrates new (optional) 15,000 rpm drives, enhanced SSA device adapters, and twice the bandwidth (as compared to previous models) to access the larger NVS (2 GB) and the larger cache option (64 GB), all integrate to give greater throughput while sustaining cache speed response times.

## **FICON host adapters**

FICON extends the IBM TotalStorage Enterprise Storage Server Model 800's ability to deliver bandwidth potential to the volumes needing it, when they need it.

## **Performance enhanced channel command words (CCWs)**

For the z/OS environments, the ESS supports channel command words (CCWs) that reduce the characteristic overhead associated to the previous (3990) CCW chains. Basically, with these CCWs, the ESS can read or write more data with fewer CCWs. CCW chains using the old CCWs are converted to the new CCWs whenever possible. The cooperation of z/OS software and the ESS provides the best benefits for the application's performance. In other words, in ESS there is less overhead associated with CCW chains by combining tasks into fewer CCWs, introducing Read Track Data and Write Track Data CCWs. They allow reading and writing more data with fewer CCWs. It will be used by z/OS to reduce ESCON protocol for multiple record transfer chains. Measurements on 4 KB records using an EXCP channel program showed a 15 percent reduction in channel overhead for the Read Track Data CCW.

### 3.18 WLM controlling PAVs

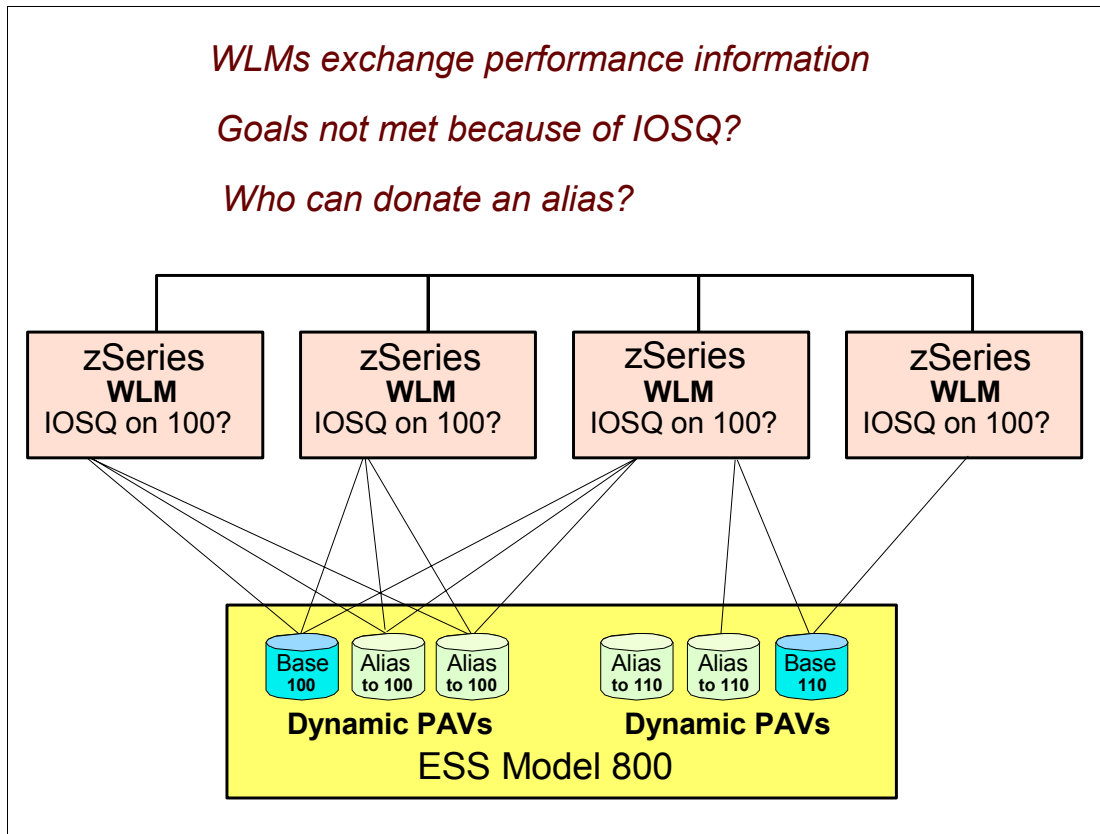


Figure 3-18 WLM controlling PAVs

#### Workload Manager (WLM)

In the zSeries Parallel Sysplex environments, the z/OS Workload Manager (WLM) controls where work is run and optimizes the throughput and performance of the total system. The ESS provides the WLM with more sophisticated ways to control the I/O across the sysplex. These functions include parallel access to both single-system and shared volumes, and the ability to prioritize the I/O based upon WLM goals. The combination of these features significantly improves performance in a wide variety of workload environments.

#### Parallel Access Volume (PAV)

Parallel Access Volume is one of the original features that the IBM TotalStorage Enterprise Storage Server brings specifically for z/OS and OS/390 operating systems, helping the zSeries running applications to concurrently share the same logical volumes.

The ability to do multiple I/O requests to the same volume nearly eliminates IOS queue time (IOSQ), one of the major components in z/OS response time. Traditionally, access to highly active volumes has involved manual tuning, splitting data across multiple volumes, and more. With PAV and the Workload Manager, you can almost forget about manual performance tuning. WLM manages PAVs across all members of a sysplex, too. The ESS, in conjunction with z/OS, has the ability to meet the performance requirements on its own.

#### Alias assignment

It will not always be easy to predict which volumes should have an alias address assigned, and how many. Your software can automatically manage the aliases according to your goals.

z/OS can exploit automatic PAV tuning if you are using WLM in goal mode. z/OS recognizes the aliases that are initially assigned to a base during the Nucleus Initialization Program (NIP) phase. WLM can dynamically tune the assignment of alias addresses. WLM monitors the device performance and is able to dynamically reassign alias addresses from one base to another if predefined goals for a workload are not met. WLM instructs IOS to reassign an alias.

### **WLM goal mode management in a sysplex**

WLM keeps track of the devices utilized by the different workloads, accumulates this information over time, and broadcasts it to the other systems in the same sysplex. If WLM determines that any workload is not meeting its goal due to IOSQ time, WLM attempts to find an alias device that can be reallocated to help this workload achieve its goal

Through WLM, there are two mechanisms to tune the alias assignment:

- ▶ The first mechanism is goal based. This logic attempts to give additional aliases to a PAV-device that is experiencing IOS queue delays and is impacting a service class period that is missing its goal. To give additional aliases to the receiver device, a donor device must be found with a less important service class period. A bitmap is maintained with each PAV-device that indicates the service classes using the device.
- ▶ The second mechanism is to move aliases to high contention PAV-devices from low contention PAV-devices. High contention devices will be identified by having a significant amount of IOSQ. This tuning is based on efficiency rather than directly helping a workload to meet its goal.

## 3.19 ESS copy services

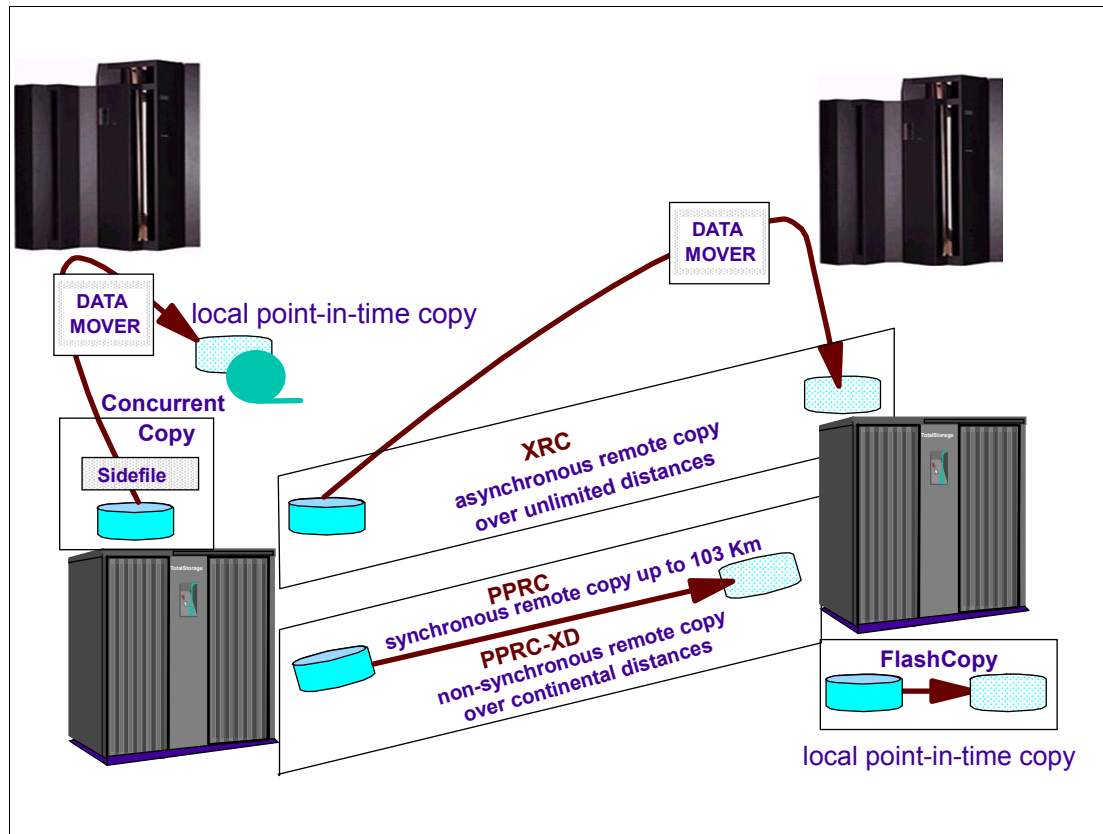


Figure 3-19 ESS copy services

### DFSMS copy services

DFSMS provides Advanced Copy Services that include a hardware and software solution to help you manage and protect your data. These solutions help ensure that your data remains available 24 hours a day, seven days a week. Advanced Copy Services provide solutions to disaster recovery, data migration, and data duplication. Many of these functions run on the IBM TotalStorage Enterprise Storage Server (ESS). With DFSMS, you can perform the following data management functions:

- ▶ Use remote copy to prepare for disaster recovery
- ▶ Move your PPRC data more easily

Remote copy provides two options that enable you to maintain a current copy of your data at a remote site. These two options are used for disaster recovery and workload migration:

- ▶ Extended remote copy (XRC)
- ▶ Peer-to-peer remote copy (PPRC)

There are two types of copy:

- ▶ To copy, an instantaneous copy where all the late updates in the primary are not copied. It is used for fast backups and data replication in general. The examples in ESS are Concurrent Copy and Flash Copy.
- ▶ Mirroring, a never-ending copy where all the updates are mirrored as fast as possible. It is used for disaster recovery and planned outages. The example in ESS are Enhanced PPRC service and XRC.

## Peer-to-peer remote copy (PPRC)

PPRC is a hardware solution which provides rapid and accurate disaster recovery as well as a solution to workload movement and device migration. Updates made on the primary DASD volumes are synchronously shadowed to the secondary DASD volumes. The local storage subsystem and the remote storage subsystem are connected through a communications link called a PPRC path. You can use one of the following protocols to copy data using PPRC:

- ▶ ESCON
- ▶ Fibre Channel Protocol

**Note:** Fibre Channel Protocol is supported only on ESS Model 800 with the appropriate licensed internal code (LIC) level and the PPRC Version 2 feature enabled.

PPRC provides a synchronous volume copy across ESS controllers. The copy is done from one controller (the one having the primary logical device) to the other (having the secondary logical device). It is synchronous because the task doing the I/O receives the CPU back with the guarantee that the copy was executed. There is a performance penalty for distances longer than 10 km. PPRC is used for disaster recovery, device migration, and workload migration; for example, it enables you to switch to a recovery system in the event of a disaster in an application system.

You can issue the **CQUERY** command to query the status of one volume of a PPRC volume pair or to collect information about a volume in the simplex state. The **CQUERY** command is modified and enabled to report on the status of S/390-attached CKD devices.

See *z/OS DFSMS Advanced Copy Services*, SC35-0428, for further information about the PPRC service and the **CQUERY** command.

## Peer-to-peer remote copy extended distance (PPRC-XD)

When you enable the PPRC extended distance feature (PPRC-XD), the primary and recovery storage control sites can be separated by long distances. Updates made to a PPRC primary volume are sent to a secondary volume asynchronously, thus requiring less bandwidth.

If you are trying to decide whether to use synchronous or asynchronous PPRC, consider the differences between the two modes:

- ▶ When you use synchronous PPRC, no data loss occurs between the last update at the primary system and the recovery site, but it increases the impact to applications and uses more resources for copying data.
- ▶ Asynchronous PPRC using the extended distance feature reduces impact to applications that write to primary volumes and uses less resources for copying data, but data might be lost if a disaster occurs. To use PPRC-XD as a disaster recovery solution, customers need to periodically synchronize the recovery volumes with the primary site and make backups to other DASD volumes or tapes.

PPRC Extended Distance (PPRC-XD) is a non-synchronous version of PPRC. This means that host updates to the source volume are not delayed by waiting for the update to be confirmed in the secondary volume. It also means that the sequence of updates on the secondary volume is not guaranteed to be the same as on the primary volume.

PPRC-XD is an excellent solution for:

- ▶ Remote data copy
- ▶ Remote data migration
- ▶ Offsite backup
- ▶ Transmission of inactive database logs

- ▶ Application disaster recovery solutions based on periodic Point-in-Time (PiT) copies of the data, if the application tolerates short interruptions (application quiesce).

PPRC-XD can operate at very long distances (such as continental distances), well beyond the 103 km supported for PPRC synchronous transmissions—and with minimal impact on the application. The distance is limited only by the network and channel extender technology capabilities.

### **Extended remote copy (XRC)**

XRC combines hardware and software to provide continuous data availability in a disaster recovery or workload movement environment. XRC provides an asynchronous remote copy solution for both system-managed and non-system-managed data to a second, remote location.

XRC relies on the IBM TotalStorage Enterprise Storage Server, IBM 3990, RAMAC Storage Subsystems, and DFSMSdftp. The 9393 RAMAC Virtual Array (RVA) does not support XRC for source volume capability.

XRC relies on the system data mover, which is part of DFSMSdftp. The system data mover is a high-speed data movement program that efficiently and reliably moves large amounts of data between storage devices. XRC is a continuous copy operation, and it is capable of operating over long distances (with channel extenders). It runs unattended, without involvement from the application users. If an unrecoverable error occurs at your primary site, the only data that is lost is data that is in transit between the time when the primary system fails and the recovery at the recovery site.

You can implement XRC with one or two systems. Let us suppose that you have two systems: an application system at one location, and a recovery system at another. With these two systems in place, XRC can automatically update your data on the remote disk storage subsystem as you make changes to it on your application system. You can use the XRC suspend/resume service for planned outages. You can still use this standard XRC service on systems attached to the ESS if these systems are installed with the toleration or transparency support.

Coupled Extended Remote Copy (CXRC) allows XRC sessions to be coupled together to guarantee that all volumes are consistent across all coupled XRC sessions. CXRC can manage thousands of volumes. IBM TotalStorage XRC Performance Monitor provides the ability to monitor and evaluate the performance of a running XRC configuration.

### **Concurrent copy**

Concurrent copy is an extended function that enables data center operations staff to generate a copy or a dump of data while applications are updating that data. Concurrent copy delivers a copy of the data, in a consistent form, as it existed before the updates took place.

### **FlashCopy service**

FlashCopy is a point-in-time copy services function that can quickly copy data from a source location to a target location. FlashCopy enables you to make copies of a set of tracks, with the copies immediately available for read or write access. This set of tracks can consist of an entire volume, a data set, or just a selected set of tracks. The primary objective of FlashCopy is to create a copy of a source volume on the target volume. This copy is called a *point-in-time copy*. Access to the point-in-time copy of the data on the source volume is through reading the data from the target volume. The actual point-in-time data that is read from the target volume might or might not be physically stored on the target volume. The ESS FlashCopy service is compatible with the existing provided by DFSMSdss. Therefore, you can invoke the FlashCopy service on the ESS with DFSMSdss.

## 3.20 TotalStorage Expert product highlights

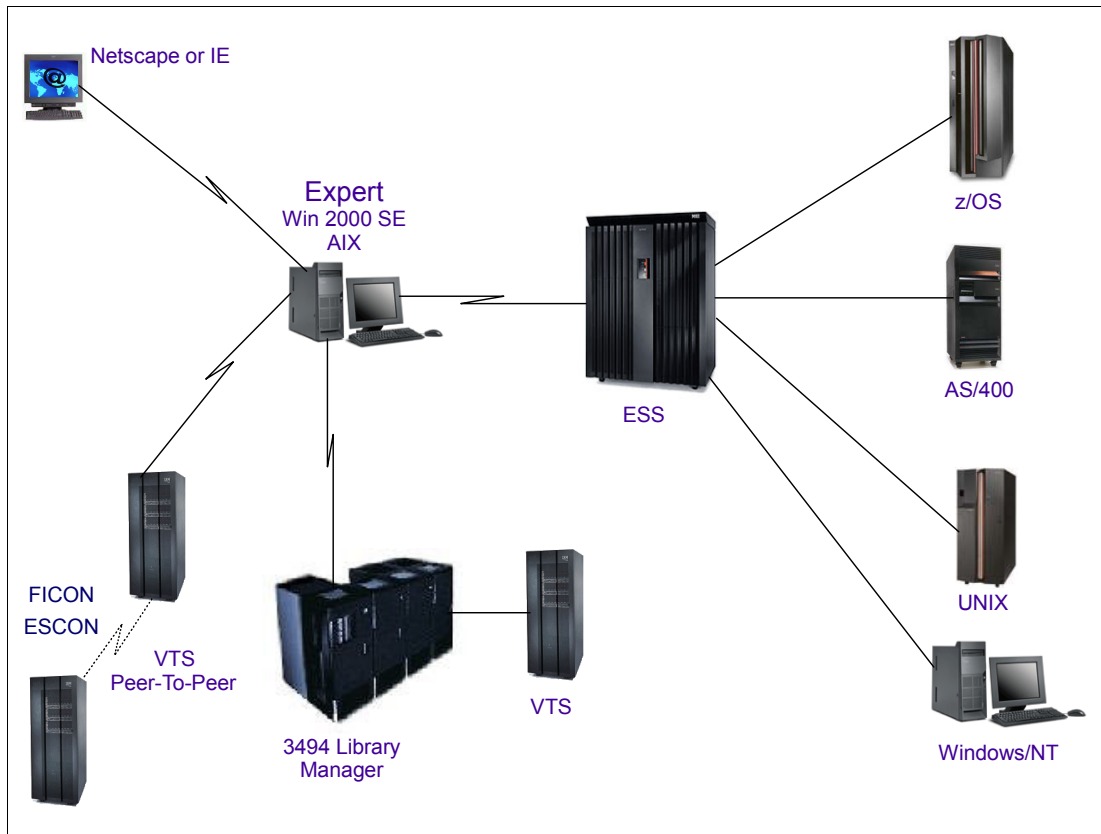


Figure 3-20 TotalStorage Expert

### TotalStorage Expert

TotalStorage Expert is an innovative software tool that gives administrators powerful, yet flexible storage asset, capacity, and performance management capabilities to centrally manage Enterprise Storage Servers located anywhere in the enterprise.

IBM TotalStorage Expert has two available features:

- ▶ The ESS feature, which supports ESS
- ▶ The ETL feature, which supports Enterprise tape library products

The two features are licensed separately. There are also upgrade features for users of StorWatch Expert V1 with either the ESS or the ETL feature, or both, who want to migrate to TotalStorage Expert V2.1.1.

TotalStorage Expert is designed to augment commonly used IBM performance tools such as Resource Management Facility (RMF), DFSMS Optimizer, AIX Performance Toolkit and similar host-based performance monitors. While these tools provide performance statistics from the host system's perspective, TotalStorage Expert provides statistics from the ESS and ETL system perspective.

By complementing other performance tools, TotalStorage Expert provides a more comprehensive view of performance; it gathers and presents information that provides a complete management solution for storage monitoring and administration.



TotalStorage Expert helps storage administrators by increasing the productivity of storage resources.

The ESS is ideal for businesses with multiple heterogeneous servers including zSeries, UNIX, Windows® NT, Windows 2000, Novell NetWare, HP/UX, Sun Solaris, and AS/400® servers.

With Version 2.1.1, the TotalStorage ESS Expert is packaged with the TotalStorage ETL Expert. The ETL Expert provides performance, asset, and capacity management for IBM's three ETL solutions:

- ▶ IBM TotalStorage Enterprise Automated Tape Library, described in "IBM TotalStorage Enterprise Automated Tape Library 3494" on page 90
- ▶ IBM TotalStorage Virtual Tape Server, described in "Introduction to Virtual Tape Server (VTS)" on page 92
- ▶ IBM TotalStorage Peer-to-Peer Virtual Tapeserver, described in "IBM TotalStorage Peer-to-Peer VTS" on page 94

Both tools can run on the same server, share a common database, efficiently monitor storage resources from any location within the enterprise, and provide a similar look and feel through a Web browser user interface. Together they provide a complete solution that helps optimize the potential of IBM disk and tape subsystems.

## 3.21 Introduction to tape processing

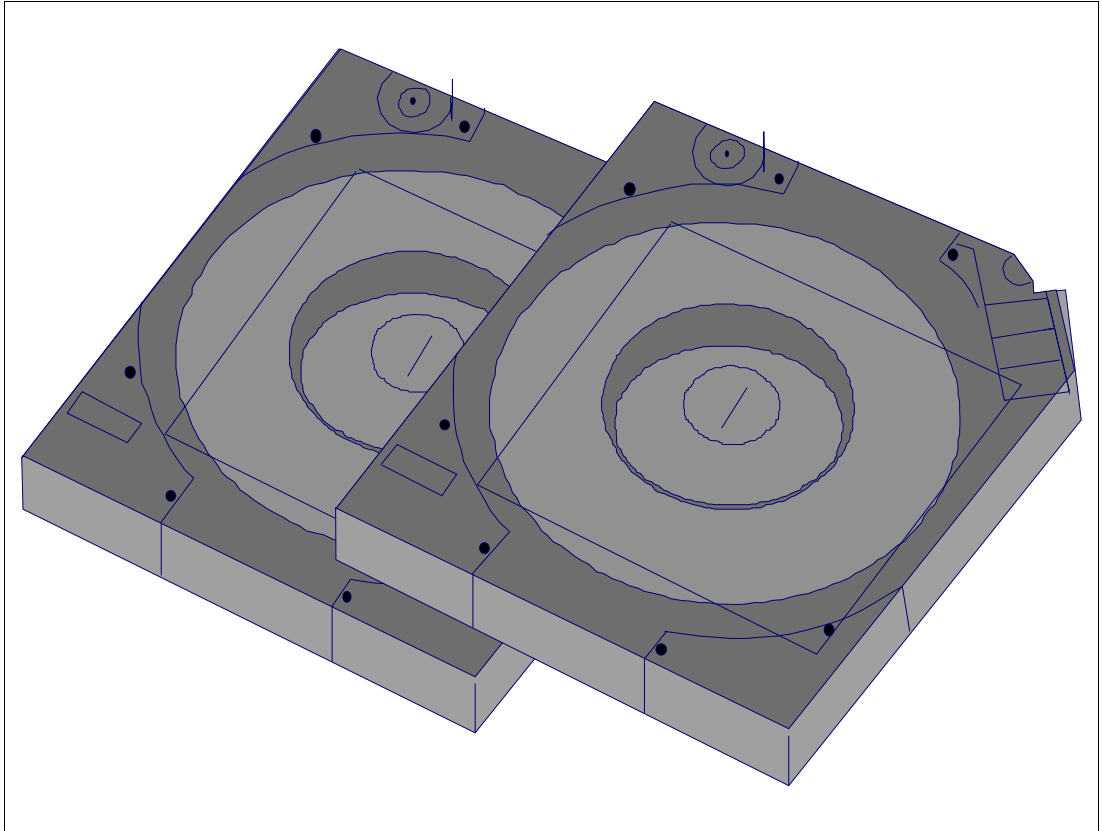


Figure 3-21 Introduction to tape processing

### Tape volumes

*Tape* refer 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:

- ▶ IBM standard labels
- ▶ Labels that follow standards published by:
  - International Organization for Standardization (ISO)
  - American National Standards Institute (ANSI)
  - Federal Information Processing Standard (FIPS)
- ▶ Nonstandard labels
- ▶ No labels

**Note:** Your installation can install a bypass for any type of label processing; however, the use of labels is recommended 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 the respective organizations, are similar to the formats of IBM standard labels.

Nonstandard tape labels can have any format and are processed by routines you provide. Unlabeled tapes contain only data sets and tape marks.

## 3.22 SL and NL format

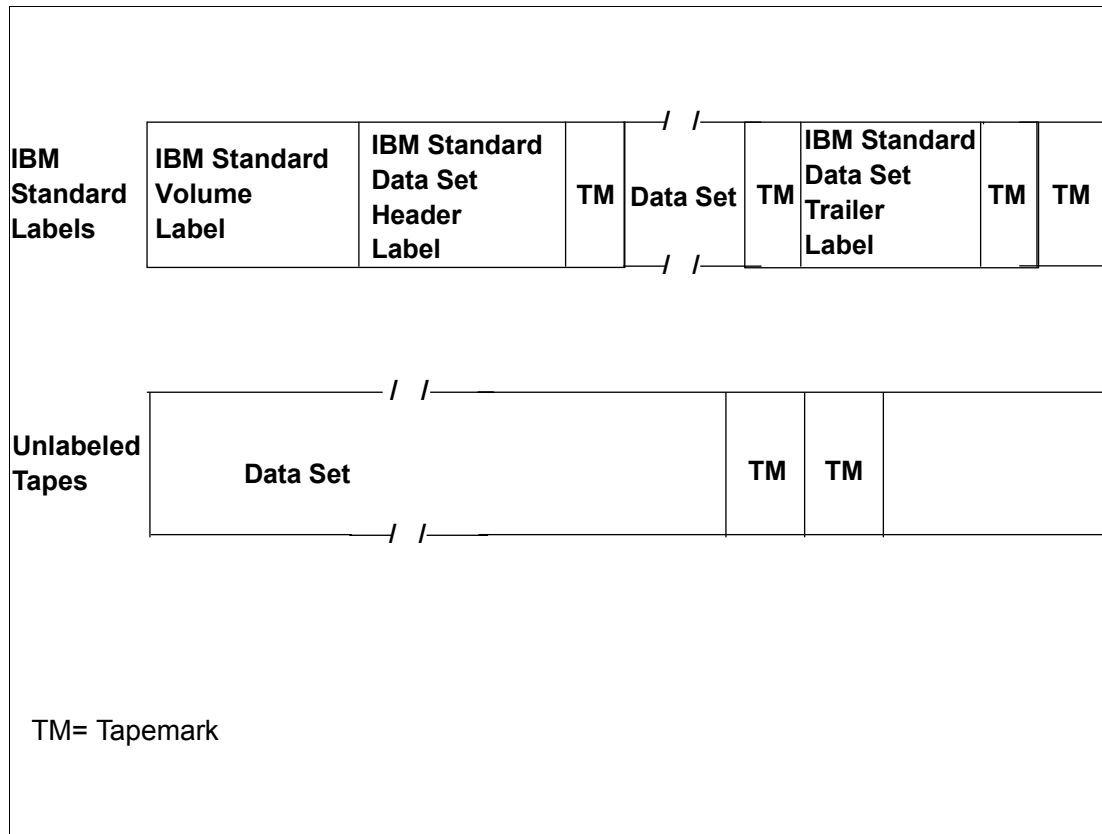


Figure 3-22 SL and NL format

### 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 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 following subparameters of the LABEL parameter as shown in Table 3-3 on page 85.

Table 3-3 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 if 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 if 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.

## 3.23 Tape capacity - tape mount management

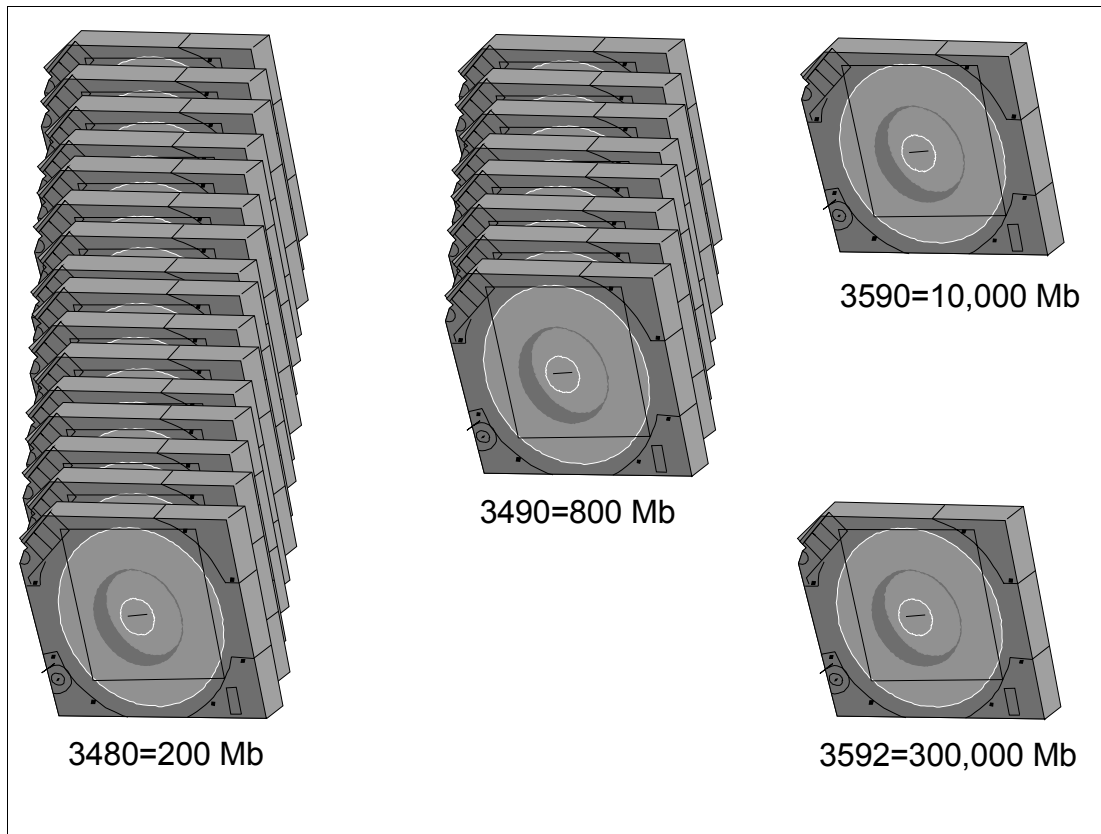


Figure 3-23 Tape capacity

### Tape capacity

The capacity of a tape depends on the device type that is recording it. 3480 and 3490 tapes are physically the same cartridges. The IBM 3590 and 3592 high performance cartridge tape is not compatible with the 3480, 3490, or 3490E drives. 3490 units can read 3480 cartridges, but cannot record as a 3480, and 3480 units cannot read or write as a 3490.

### 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 information you need to effectively implement the tape mount management methodology recommended by IBM.

Tape mount management allows you to efficiently fill a tape cartridge to its capacity and gain full benefit from improved data recording capability (IDRC) compaction, 3490E Enhanced Capability Magnetic Tape Subsystem, 36-track enhanced recording format, and Enhanced Capacity Cartridge System Tape. By filling your tape cartridges, you reduce your tape mounts and even the number of tape volumes you need.

With an effective tape cartridge capacity of 2.4 GB using 3490E and the Enhanced Capacity Cartridge System Tape, DFSMS can intercept all but extremely large data sets and manage them with tape mount management. By implementing tape mount management with DFSMS, you might reduce your tape mounts by 60 to 70% with little or no additional hardware

required. Therefore, the resulting tape environment would be able to fully exploit integrated cartridge loaders (ICL), IDRC, and 3490E.

Tape mount management also improves job throughput because jobs are no longer queued up on tape drives. Approximately 70% of all tape data sets queued up on drives are less than 10 MB. With tape mount management, these data sets reside on DASD while in use. This frees up the tape drives for other allocations.

Tape mount management recommends that you use DFSMSHsm (TM) 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. DFSMSHsm scans this buffer on a regular basis 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.

Table 3-4 lists all IBM tape capacities supported since 1952.

*Table 3-4 Tape capacity of various IBM products*

Year	Product	Capacity (Mb)	Transfer Rate (KB/S)
1952	IBM 726	1.4	7.5
1953	IBM 727	5.8	15
1957	IBM 729	23	90
1965	IBM 2401	46	180
1968	IBM 2420	46	320
1973	IBM 3420	180	1,250
1984	IBM 3480	200	3,000
1989	IBM 3490	200	4,500
1991	IBM 3490E	400	9,000
1992	IBM 3490E	800	9,000
1995	IBM 3590 Magstar	10,000 (uncompacted)	9,000 (uncompacted)
1999	IBM 3590E Magstar	20,000 (uncompacted)	14,000
2000	IBM 3590E Magstar XL Cartridge	20,000/40,000 (dependent on Model B or E)	14,000
2003	IBM 3592 TotalStorage Enterprise Tape Drive	300,000 (for high capacity requirements) or 60,000 (for fast data access requirements)	40,000

For further information about tape processing, see *z/OS DFSMS Using Magnetic Tapes*, SC26-7412.

**Note:** z/OS supports tape devices starting from D/T 3420.

## 3.24 TotalStorage Enterprise Tape Drive 3592 Model J1A

- ❑ The IBM 3592 is an "Enterprise Class" tape drive
  - Data rate 40 MB/s (without compression)
  - Capacity 300 GB per cartridge (without compression)
  - 60 GB format to provide fast access to data
- ❑ Dual ported 2 Gbps Fiber Channel interface
  - Autonegotiates (1 or 2 Gbps, Fabric or loop support)
  - Options may be hard-set at drive
- ❑ Drive designed for automation solutions
  - Small form factor
  - Cartridge similar form factor to 3590 and 3490
- ❑ Improved environmentals



Figure 3-24 TotalStorage Enterprise Tape Drive 3592 Model J1A

### IBM 3592 tape drive

The IBM 3592 tape drive is the fourth generation of high capacity and high performance tape systems. It was announced in September 2003 and connects to IBM @server zSeries systems via the TotalStorage Enterprise Tape Controller 3592 Model J70 using ESCON or FICON 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 standalone solution or as an automated solution within a 3494 tape library.

### Enterprise class tape drive

The native rate for data transfer increases up to 40 MB/sec compared to 14 MB/sec in a 3590 Magstar. The uncompressed amount of data which fits on a single cartridge increases to 300 GB and will be used for scenarios where high capacity is needed. The tape drive has a second option, where you can store a maximum of 60 GB per tape. This option is used whenever fast access to tape data is needed.

### Dual ported 2 Gbps fiber channel interface

This tape drive generation connects to the tape controller 3592 Model J70 via fiber channel. SCSI connection, as it used in 3590 configuration, is no longer supported. However, if you connect a 3590 Magstar tape drive to a 3592 controller, SCSI connection is possible.



### **Drive designed for automation solutions**

The drive has a smaller form factor. Thus, you can integrate more drives into an automated tape library. The cartridges have a similar form factor to the 3590 and 3490 cartridge, so they fit into the same slots in a 3494 automated tape library.

### **Improved environmental**

By using a smaller form factor than 3590 Magstar drives, you can put two 3592 drives in place of one 3590 drive in the 3494. In a standalone solution you can put a maximum of 12 drives into one 19-inch rack, managed by one controller.

## 3.25 IBM TotalStorage Enterprise Automated Tape Library 3494



Figure 3-25 3494 tape library

### IBM 3494 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. Various solutions that provide tape automation are available, including:

- ▶ The Automatic Cartridge Loader on IBM 3480 and 3490E tape subsystems, which provides quick scratch (a volume with no valued data, used for output) mount.
- ▶ The Automated Cartridge Facility on the Magstar 3590 tape subsystem, which, working with application software, can provide a 10-cartridge mini-tape library.
- ▶ The IBM 3494, an automated tape library dataserwer), is a device consisting of robotics components, cartridge storage areas (or shelves), tape subsystems, and controlling hardware and software, together with the set of tape volumes that reside in the library and can be mounted on the library tape drives.
- ▶ The Magstar Virtual Tape Server (VTS), which provides “volume stacking” capability and exploits the capacity and bandwidth of Magstar 3590 technology.

### 3494 models and features

IBM 3494 offers a wide range of models and features:

- ▶ Up to 96 tape drives

- ▶ Support through the Library Control Unit for attachment of up to 15 additional frames including the Magstar VTS, for a total of 16 frames, not including the High Availability unit
- ▶ Cartridge storage capacity of 291 to 6145 tape cartridges
- ▶ Data storage capacity of up to 1.84 PB (Petabytes) of uncompact data and 5.52 PB of compacted data (at a compression rate of 3:1)
- ▶ Support of the High Availability unit that provides a high level of availability for tape automation.
- ▶ Support of the IBM Total Storage Virtual Tape Server
- ▶ Support of the IBM Total Storage Peer-to-Peer VTS
- ▶ Support for the following tape drives:
  - IBM 3490E Model F1A tape drive
  - IBM 3490E Model CxA tape drives
  - IBM Magstar 3590 Model B1A tape drives
  - IBM Magstar 3590 Model E1A tape drives
  - IBM Magstar 3590 Model H1A tape drives
  - IBM TotalStorage Enterprise Tape Drive 3592 Model J1A
- ▶ Attachment to and sharing by multiple host systems, such as IBM @server zSeries, iSeries, pSeries®, S/390, RS/6000®, AS/400, HP, and Sun processors
- ▶ Data paths through FICON, fibre channels, SCSI-2, ESCON, and parallel channels depending on the tape subsystem installed
- ▶ Library management commands through RS-232, a local area network (LAN), and parallel, ESCON and FICON channels

## 3.26 Introduction to Virtual Tape Server (VTS)

- ❑ VTS models:
  - Model B10 VTS
  - Model B20 VTS
  - Peer-to-Peer (PtP) VTS (up to twenty-four 3590 tape drives)
- ❑ VTS design (single VTS)
  - 32, 64, 128 or 256 3490E virtual devices
  - Tape volume cache:
    - Analogous to DASD cache
    - Data access through the cache
    - Dynamic space management
    - Cache hits eliminate tape mounts
  - Up to twelve 3590 tape drives (the real 3590 volume contains up to 250,000 virtual volumes per VTS)
  - Stacked 3590 tape volumes managed by the 3494

Figure 3-26 Introduction to VTS

### VTS introduction

The IBM Magstar Virtual Tape Server (VTS), integrated with the IBM Tape Library Dataservers (3494), delivers an increased level of storage capability to the traditional storage products hierarchy. The host software sees VTS as a 3490 Enhanced Capability (3490E) Tape Subsystem with associated standard (CST) or Enhanced Capacity Cartridge System Tapes (ECCST). This virtualization of both the tape devices and the storage media to the host allows for transparent utilization of the capabilities of the IBM 3590 tape technology.

Along with introducing the IBM Magstar VTS, IBM introduced new views of volumes and devices because of the different knowledge about volumes and devices in the host system and the hardware. Using a VTS subsystem, the host application writes tape data to virtual devices. The volumes created by the hosts are called Virtual Volumes and are physically stored in a tape volume cache that is built from RAID DASD.

### VTS models

These are the IBM 3590 drives you can choose:

- ▶ For the Model B10 VTS, four, five or six 3590-B1A/E1A/H1A can be associated with VTS
- ▶ For the Model B20 VTS, six to twelve 3590-B1A/E1A/H1A can be associated with VTS

Each ESCON channel in the VTS is capable of supporting 64 logical paths, providing up to 1024 logical paths for Model B20 VTS with sixteen ESCON channels, and 256 logical paths for Model B10 VTS with four ESCON channels. Each logical path can address any of the 32, 64, 128, or 256 virtual devices in the Model B20 VTS.

Each FICON channel in the VTS can support up to 128 logical paths, providing up to 1024 logical paths for the Model B20 VTS with eight FICON channels. With a Model B10 VTS, 512 logical paths can be provided with four FICON channels. As with ESCON, each logical path can address any of the 32, 64, 128, or 256 virtual devices in the Model B20 VTS.

**Note:** Intermixing FICON and SCSI interfaces is not supported.

### **Tape volume cache**

The IBM TotalStorage Peer-to-Peer Virtual Tape Server appears to the host processor as a single automated tape library with 64, 128, or 256 virtual tape drives and up to 250,000 virtual volumes. The configuration of this system has up to 3.5 TB of Tape Volume Cache native (10.4 TB with 3:1 compression), up to 24 IBM TotalStorage 3590 tape drives, and up to 16 host ESCON or FICON channels.

Through tape volume cache management policies, the VTS management software moves host-created volumes from the tape volume cache to a Magstar cartridge managed by the VTS subsystem. When a virtual volume is moved from the tape volume cache to tape, it becomes a logical volume.

### **VTS design**

VTS looks like an automatic tape library with thirty-two 3490E drives and 50,000 volumes in 37 square feet. Its major components are:

- ▶ Magstar 3590 (three or six tape drives) with two ESCON channels
- ▶ Magstar 3494 Tape Library
- ▶ Fault-tolerant RAID-1 disks (36 Gb or 72 Gb)
- ▶ RISC Processor

### **VTS functions**

VTS provides the following functions:

- ▶ Thirty-two 3490E virtual devices.
- ▶ Tape volume cache (implemented in a RAID-1 disk) which contains virtual volumes. At close time, the virtual volume is copied to logical volumes in the 3590 tape volumes:
  - Analogous to DASD cache
  - Data access through the cache
  - Dynamic space management
  - Cache hits eliminate tape mounts
- ▶ Up to six 3590 tape drives; the real 3590 volume contains logical volumes. Installation sees up to 50,000 volumes.
- ▶ Stacked 3590 tape volumes managed by the 3494. It fills the tape cartridge up to 100%. Putting multiple virtual volumes into a stacked volume, VTS uses all of the available space on the cartridge. VTS uses IBM 3590 cartridges when stacking volumes

VTS is expected to provide a ratio of 59:1 in volume reduction, with dramatic savings in all tape hardware (drives, controllers, and robots).

## 3.27 IBM TotalStorage Peer-to-Peer VTS

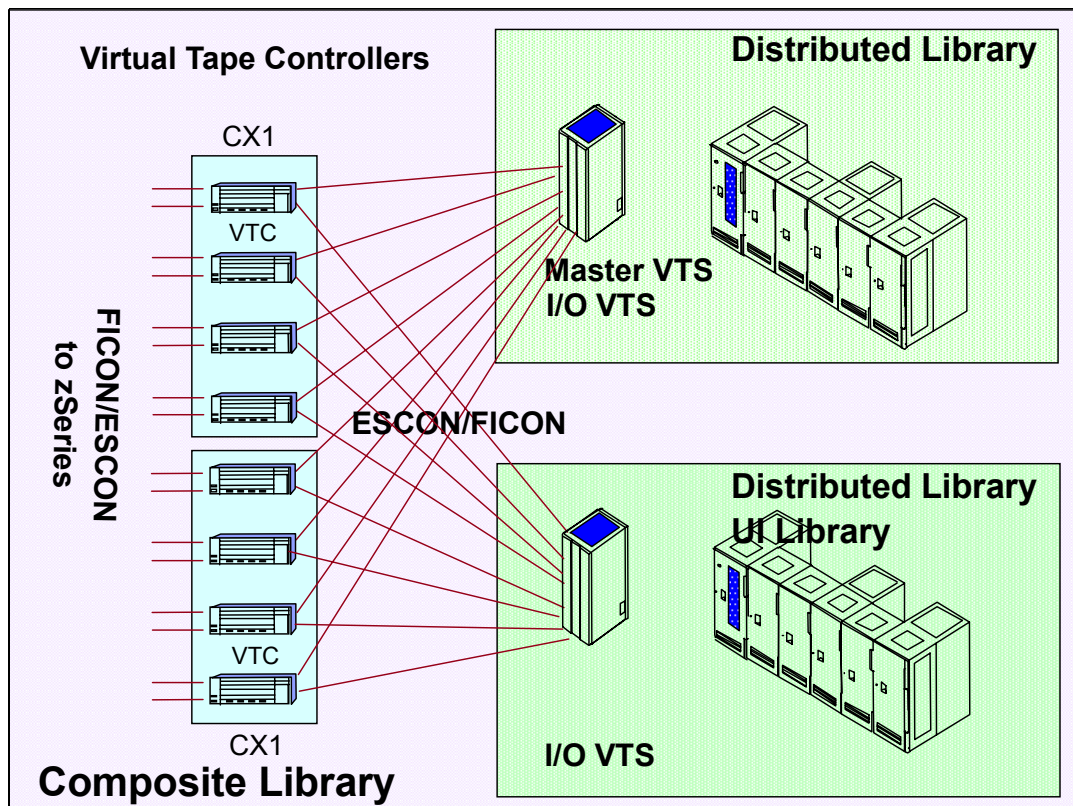


Figure 3-27 IBM TotalStorage Peer-to-Peer VTS

### Peer-to-Peer VTS

IBM TotalStorage Peer-to-Peer Virtual Tape Server, an extension of IBM TotalStorage Virtual Tape Server, is specifically designed to enhance data availability. It accomplishes this by providing dual 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), IBM TotalStorage Peer-to-Peer Virtual Tape Server improves system reliability and availability, as well as data access. To help protect current hardware investments, existing IBM TotalStorage Virtual Tape Servers can be upgraded for use in this new configuration.

IBM TotalStorage Peer-to-Peer Virtual Tape Server consists of new models and features of the 3494 Tape Library that are used to join two separate Virtual Tape Servers into a single, interconnected system. The two virtual tape systems can be located at the same site or at different sites that are geographically remote. This provides a remote copy capability for remote vaulting applications.

IBM TotalStorage Peer-to-Peer Virtual Tape Server appears to the host IBM @server zSeries processor as a single automated tape library with 64, 128, or 256 virtual tape drives and up to 500,000 virtual volumes. The configuration of this system has up to 3.5 TB of Tape Volume Cache native (10.4 TB with 3:1 compression), up to 24 IBM TotalStorage 3590 tape drives, and up to 16 host ESCON or FICON channels.

In addition to the 3494 VTS components B10, B18 and B20, the Peer-to-Peer VTS consists of the following components:

- ▶ The 3494 virtual tape controller model VTC

The VTC in the Virtual Tape Frame 3494 Model CX1 provides interconnection between two VTSs with the Peer-to-Peer Copy features, and provides two host attachments for the PtP VTS. There must be four (for the Model B10 or B18) or eight (for the Model B20 or B18) VTCs in a PtP VTS configuration. Each VTC is an independently operating, distributed node within the PtP VTS, which continues to operate during scheduled or unscheduled service of another VTC.

- ▶ The 3494 auxiliary tape frame model CX1

The Model CX1 provides the housing and power for two or four 3494 virtual tape controllers. Each Model CX1 can be configured with two or four Model VTCs. There are two power control compartments, each with its own power cord, to allow connection to two power sources.

### **Peer-to-Peer copy features**

Special features installed on 3494 Models B10, B18 and B20 in a Peer-to-Peer configuration provide automatic copies of virtual volumes. These features can be installed on existing VTS systems to upgrade them to a Peer-to-Peer VTS.

### **VTS advanced functions**

As with a stand-alone VTS, the Peer-to-Peer VTS has the option to install some additional features and enhancements to existing features. These new features are:

- ▶ Outboard policy management - Outboard policy management enables the storage administrator to manage SMS data classes, storage classes, management classes and storage groups at the library manager or the 3494 specialist.
- ▶ Physical volume pooling - With outboard policy management enabled, you are able to 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 VTS. This function is also available in the Peer-to-Peer environment. At the site or location where the second distributed library is located, logical volumes can also be duplexed, in which case you could have two or four copies of your data.
- ▶ Peer-to-Peer copy control

There are two types of copy operations:

- Immediate - which creates a copy of the logical volume in the companion connected virtual tape server prior to completion of a rewind/unload command. This mode provides the highest level of data protection.
  - Deferred - which creates a copy of the logical volume in the companion connected virtual tape server as activity permits after receiving a rewind/unload command. This mode provides protection that is superior to most currently available backup schemes.
- ▶ Tape volume cache management - prior to the introduction of these features, there was no way to influence cache residency. As a result, all data written to the TVC was premigrated using a first-in, first-out (FIFO) method. With the introduction of this function, you now have the ability to influence the time that virtual volumes reside in the TVC.

## 3.28 Storage area network (SAN)



Figure 3-28 Storage area network (SAN)

### Storage area network

The Storage Network Industry Association (SNIA) defines 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 consisting of storage elements, storage devices, computer systems, and/or appliances, plus all control software, communicating over a network.

SANs today are usually built using fibre channel technology, but the concept of a SAN is independent of the underlying type of network.

The major potential benefits of a SAN can be categorized as:

#### ► 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—this 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 different 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. In order for a node to transfer data, it must first arbitrate to win control of the loop. Once 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. Once 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 some cases, increased redundancy.

Multiple switches can be connected to form a switch fabric capable of supporting a large number of host servers and storage subsystems. When switches are connected, each switch’s configuration information has to be copied into all the other participating switches. This is called *cascading*.

## **FICON and SAN**

From a zSeries perspective, FICON is the protocol which is used in a SAN environment. A FICON infrastructure may be point-to-point or switched, using ESCON directors with FICON bridge cards or FICON directors to provide connections between channels and control units. FICON uses fibre channel transport protocols, and so uses the same physical fiber.

Today zSeries has 2 Gbps link data rate support. The 2 Gbps links are for native FICON, FICON CTC, cascaded directors and fibre channels—FCP channels—on the FICON Express cards on z800, z900, and z990 only.





## Storage management software

DFSMS is an exclusive element of the z/OS operating system and automatically manages data from creation to expiration. In this chapter we present the following:

- ▶ 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 the DFSMS,:
  - DFSMSdftp, 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
  - DFSMSStvs, an optional feature of z/OS
  - z/OS DFSORT™

## 4.1 Overview of DFSMSdfp utilities

- ❑ IEBCOMPR: Compare records in SEQ/PDS(E)
- ❑ IEBCOPY: Copy/Merge/Compr./Manage PDS(E)
- ❑ IEBDG: Create test data set
- ❑ IEBEDIT: Selectively copy Job steps
- ❑ IEBGENER (ICEGENER): Convert SEQ to PDS
- ❑ IEBPTPCH: Print a SEQ/PDS(E)
- ❑ IEBUPDTE: Modify SEQ/PDS(E)
- ❑ IEHLIST: List data sets
- ❑ IEHINITT: Write standard labels on tape volumes
- ❑ IEHMOVE: Move or copy data
- ❑ IEHPROGM: Build, maintain system control data
- ❑ IFHSTATR: Select, format, write info about tape errors

Figure 4-1 DFSMSdfp utilities

### DFSMSdfp utilities

Utilities are programs which 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 JCL, and utility control statements.

The base JCL and some utility control statements necessary to use these utilities are provided in the major discussion of each utility program in this chapter. For more details and to help you find the program that performs the function you need, see “Guide to Utility Program Functions” in topic 1.1 of *z/OS DFSMSdfp Utilities*, SC26-7414.

### 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 performed more efficiently with other programs, such as IDCAMS, ISMF, or DFSMSrmm.

Table 4-1 on page 101 lists and describes system utilities.

**Note:** Programs that provide functions which are better performed by newer applications (such as ISMF or DFSMSrmm or DFSMSdss) are marked with an asterisk (\*) in the table.

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.

### 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 partitioned data sets extended (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 topic cannot be used with VSAM data sets. You use the IDCAMS utility to manipulate VSAM data set; refer to “Invoking the IDCAMS utility program” on page 119.

Table 4-2 lists data set utility programs and their use. Programs that provide functions which are better performed by newer applications such as ISMF or DFSMSrmm or DFSMSdss are marked with an asterisk (\*) in the table.

Table 4-2 Data set utility programs

Data set utility	Use
*IEBCOMPR, SuperC, (PDF 3.12)	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 or AMS REPRO	Modify, print, or link modules for use with the IBM 3800 Printing Subsystem, the IBM 3262 Model 5, or the 4284 printer.
*IEBISAM	Unload, load, copy, or print an ISAM data set.
IEBPTPCH or PDF 3.1 or 3.6	Print or punch records in a sequential or partitioned data set.
IEBUPDTE	Incorporate changes to sequential or partitioned data sets, or PDSEs.

The next figures show examples of using utility programs.

## 4.2 IEBCOMPR (compare data set) program

```
//STEPA EXEC PGM=IEBCOMPR
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSNAME=FPITA.DATA1,
        DISP=(OLD,KEEP)
//SYSUT2 DD DSNAME=FPITA.DATA2,
        DISP=(OLD,KEEP)

//SYSIN DD DUMMY
```

Figure 4-2 IEBCOMPR utility example

### IEBCOMPR utility

IEBCOMPR is a data set utility used to compare two sequential data sets, two partitioned data sets, or two PDSEs at the logical record level, to verify a backup copy. Fixed, variable, or undefined records from blocked or unblocked data sets or members can also be compared. However, you should not use IEBCOMPR to compare *load modules*.

Two sequential data sets are considered equal (that is, are considered to be identical) if:

- ▶ The data sets contain the same number of records, and
- ▶ Corresponding records and keys are identical

Two partitioned data sets or two PDSEs are considered equal if:

- ▶ Corresponding members contain the same number of records
- ▶ Note lists are in the same position within corresponding members
- ▶ Corresponding records and keys are identical
- ▶ Corresponding directory user data fields are identical

If all these conditions are not met for a specific type of data set, those data sets are considered *unequal*. If records are unequal, the record and block numbers, the names of the DD statements that define the data sets, and the unequal records are listed in a message data set. Ten successive unequal comparisons stop the job step, unless you provide a routine for handling error conditions.

**Note:** Load module partitioned data sets that reside on different types of devices should not be compared. Under most circumstances, the data sets will not compare as equal.

A partitioned data set or partitioned data set extended (PDSE) can be compared only if all names in one or both directories have counterpart entries in the other directory. The comparison is made on members identified by these entries and corresponding user data.

**Recommendation:** Use the SuperC utility instead of IEBCOMPR. SuperC is part of ISPF/PDF and the High Level Assembler Toolkit Feature. SuperC can be processed in the foreground as well as in batch, and its report is more useful.

An example of the IEBCOMPR utility is shown in Figure 4-3 on page 104 and Figure 4-2 on page 102.

## 4.3 Comparing data sets

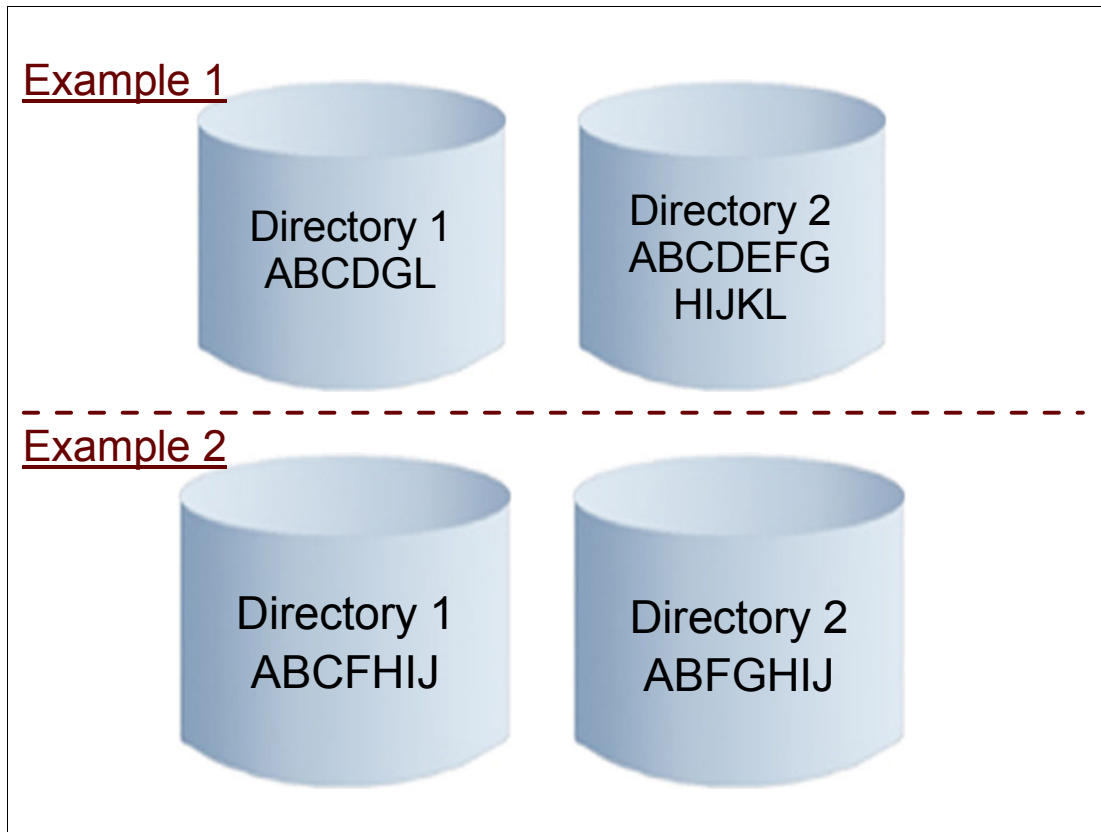


Figure 4-3 Comparing data sets examples

### Examples of comparing data sets

As mentioned, partitioned data sets or PDSEs can be compared only if all the names in one or both of the directories have counterpart entries in the other directory. The comparison is made on members that are identified by these entries and corresponding user data.

#### Example 1

Figure 4-3 shows the directories of two partitioned data sets. Directory 2 contains corresponding entries for all the names in Directory 1; therefore, the data sets can be compared.

#### Example 2

Figure 4-3 shows the directories of two partitioned data sets. Each directory contains a name that has no corresponding entry in the other directory; therefore, the data sets cannot be compared, and the job step will be ended.

**Note:** Load module partitioned data sets that reside on different types of devices should not be compared. Under most circumstances, the data sets will not compare as equal.



## 4.4 IEBCOPY utility

```
//COPY JOB ...
//JOBSTEP EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=A
//OUT1 DD DSN=DATASET1,UNIT=disk,VOL=SER=111112,
//      DISP=(OLD,KEEP)
//IN6  DD DSN=DATASET6,UNIT=disk,VOL=SER=111115,
//      DISP=OLD
//IN5  DD DSN=DATASET5,UNIT=disk,VOL=SER=111116,
//      DISP=(OLD,KEEP)
//SYSUT3 DD UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN DD *
COPYOPER COPY OUTDD=OUT1
         INDD=IN5,IN6
         SELECT MEMBER=((B,,R),A)
```

Figure 4-4 IEBCOPY utility example

### IEBCOPY utility example

IEBCOPY is a data set utility used to copy or merge members between one or more partitioned data sets, or partitioned data sets extended (PDSE), in full or in part. You can also use IEBCOPY to create a backup of a partitioned data set into a sequential data set (called an unload data set or PDSU), and to copy members from the backup into a partitioned data set.

IEBCOPY is used to:

- ▶ Make a copy of a partitioned data set or PDSE
- ▶ Merge partitioned data sets (except when unloading)
- ▶ Create a sequential form of a partitioned data set or PDSE for a backup or transport
- ▶ Reload one or more members from a PDSU into a partitioned data set or PDSE
- ▶ Select specific members of a partitioned data set or PDSE to be copied, loaded, or unloaded
- ▶ Replace members of a partitioned data set or PDSE
- ▶ Rename selected members of a partitioned data set or PDSE
- ▶ Exclude members from a data set to be copied, unloaded, or loaded (except on COPYGRP)
- ▶ Compress a partitioned data set in place

- ▶ Upgrade an OS format load module for faster loading by MVS program fetch
- ▶ Copy and re-block load modules
- ▶ Convert load modules in a partitioned data set to program objects in a PDSE when copying a partitioned data set to a PDSE
- ▶ Convert a partitioned data set to a PDSE, or a PDSE to a partitioned data set
- ▶ Copy—to or from a PDSE data set— a member and its aliases together as a group (COPYGRP)

In addition, IEBCOPY automatically lists the number of unused directory blocks and the number of unused tracks available for member records in the output partitioned data set.

## 4.5 IEBCOPY copy operation

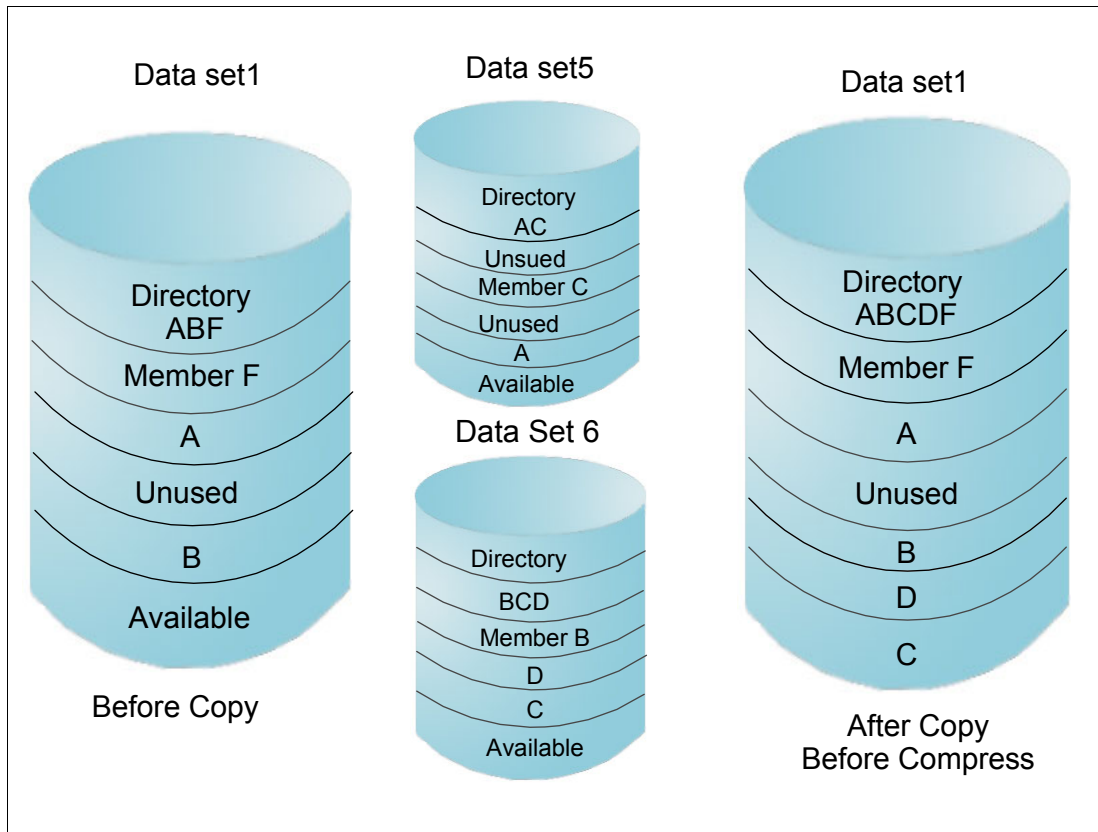


Figure 4-5 IEBCOPY copy operation

### Copy control command example

In Figure 4-5, two input partitioned data sets (data set5 and data set6) are copied to an existing output partitioned data set (data set1). In addition, all members on data set6 are copied; members on the output data set that have the same names as the copied members are replaced. After data set6 is processed, the output data set (data set1) is compressed in place. Figure 4-5 shows the input and output data sets before and after copy processing. The compress process is shown in Figure 4-7 on page 109. Figure 4-6 on page 107 shows the job that is used to copy and compress partitioned data sets.

```
//COPY      JOB      ...
//JOBSTEP   EXEC    PGM=IEBCOPY
//SYSPRINT  DD      SYSOUT=A
//INOUT1    DD      DSNAME=data set1,UNIT=disk,VOL=SER=111112,DISP=(OLD,KEEP)
//IN5       DD      DSNAME=data set5,UNIT=disk,VOL=SER=111114,DISP=OLD
//IN6       DD      DSNAME=data set6,UNIT=disk,VOL=SER=111115,
//           DISP=(OLD,KEEP)
//SYSUT3    DD      UNIT=SYSDA,SPACE=(TRK,(1))
//SYSUT4    DD      UNIT=SYSDA,SPACE=(TRK,(1))
//SYSIN     DD      *
              COPYOPER COPY  OUTDD=INOUT1,INDD=(IN5,(IN6,R),INOUT1)
/*
```

Figure 4-6 IEBCOPY with copy and compress

The control statement is discussed below:

- ▶ INOUT1 DD defines a partitioned data set (data set1), which contains three members (A, B, and F).
- ▶ IN5 DD defines a partitioned data set (data set5), which contains two members (A and C).
- ▶ IN6 DD defines a partitioned data set (data set6), which contains three members (B, C, and D).
- ▶ SYSUT3 and SYSUT4 DD define temporary spill data sets. One track is allocated for each on a disk volume.
- ▶ SYSIN DD defines the control data set, which follows in the input stream. The data set contains a COPY statement.
- ▶ COPY indicates the start of the copy operation. The OUTDD operand specifies data set1 as the output data set.

Processing occurs as follows:

1. Member A is not copied from data set5 into data set1 because it already exists on data set1 and the replace option was not specified for data set5.
2. Member C is copied from data set5 to data set1, occupying the first available space.
3. All members are copied from data set6 to data set1, immediately following the last member. Members B and C are copied even though the output data set already contains members with the same names because the replace option is specified on the data set level.

## 4.6 IEBCOPY compress operation

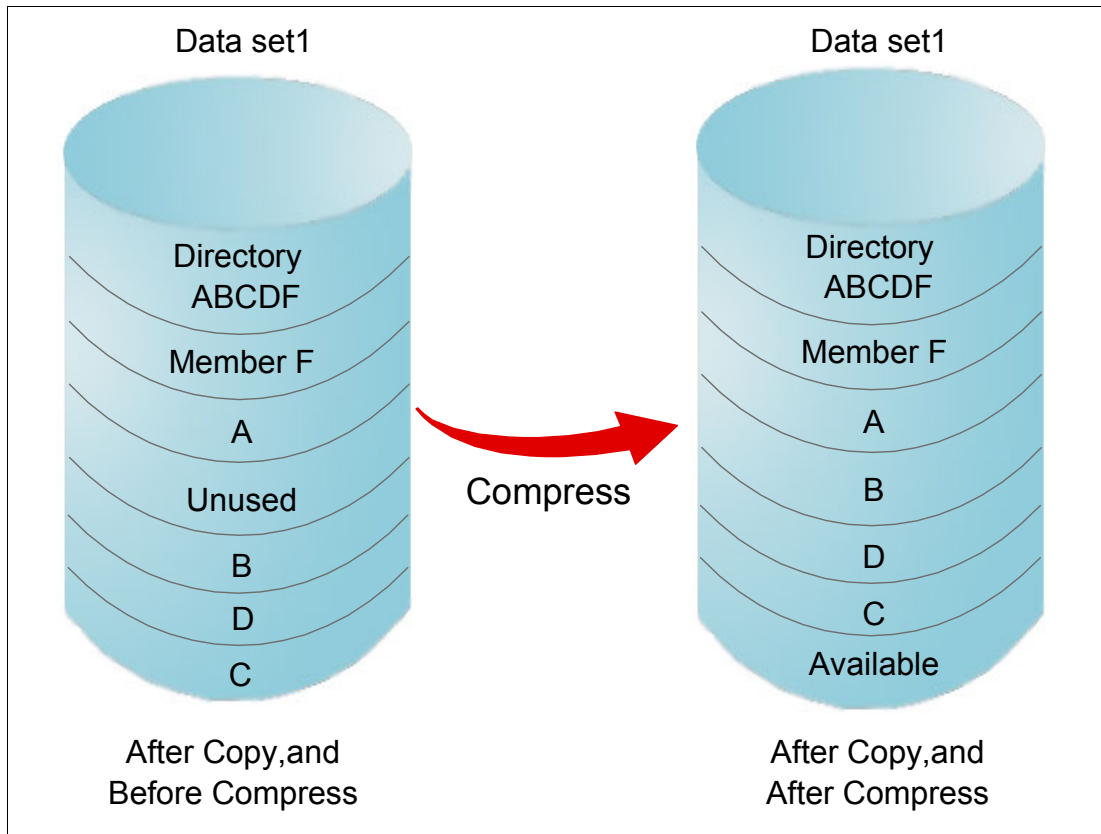


Figure 4-7 IEBCOPY compress operation

### IEBCOPY compress operation

The pointers in the data set1 directory are changed to point to the new members B and C. Thus, the space occupied by the old members B and C is unused. The members currently on data set1 are compressed in place, thereby eliminating embedded unused space.

## 4.7 IEBCGEN

```
//DISKTOTP JOB ...
//STEP1 EXEC PGM=IEBCGEN
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSNAME=INSET,UNIT=disk,DISP=SHR
//SYSUT2 DD DSNAME=OUTPUT,UNIT=CART,DISP=(NEW,KEEP),,
//          VOLUME=SER=APSG90,UNIT=3490,LABEL=(1,SL),
//          DCB=*.SYSUT1
//SYSIN DD *
```

Figure 4-8 IEBCGEN

### Using IEBCGEN

You can use IEBCGEN to:

- ▶ Create a backup copy of a sequential data set, a member of a partitioned data set, or PDSE.
- ▶ Produce a partitioned data set or PDSE, or a member of a partitioned data set or PDSE, from a sequential data set.
- ▶ Expand an existing partitioned data set or PDSE by creating partitioned members and merging them into the existing data set.
- ▶ Produce an edited sequential or partitioned data set or PDSE.
- ▶ Manipulate data sets containing double-byte character set data.
- ▶ Print sequential data sets or members of partitioned data sets or PDSEs.
- ▶ Re-block or change the logical record length of a data set.

**Note:** If you have the DFSORT product installed, you should be using ICEGEN as an alternative to IEBCGEN when making an unedited copy of a data set or member. It may already be installed in your system under the name IEBCGEN. It generally gives better performance.

## 4.8 Adding members to a PDS using IEBGENER

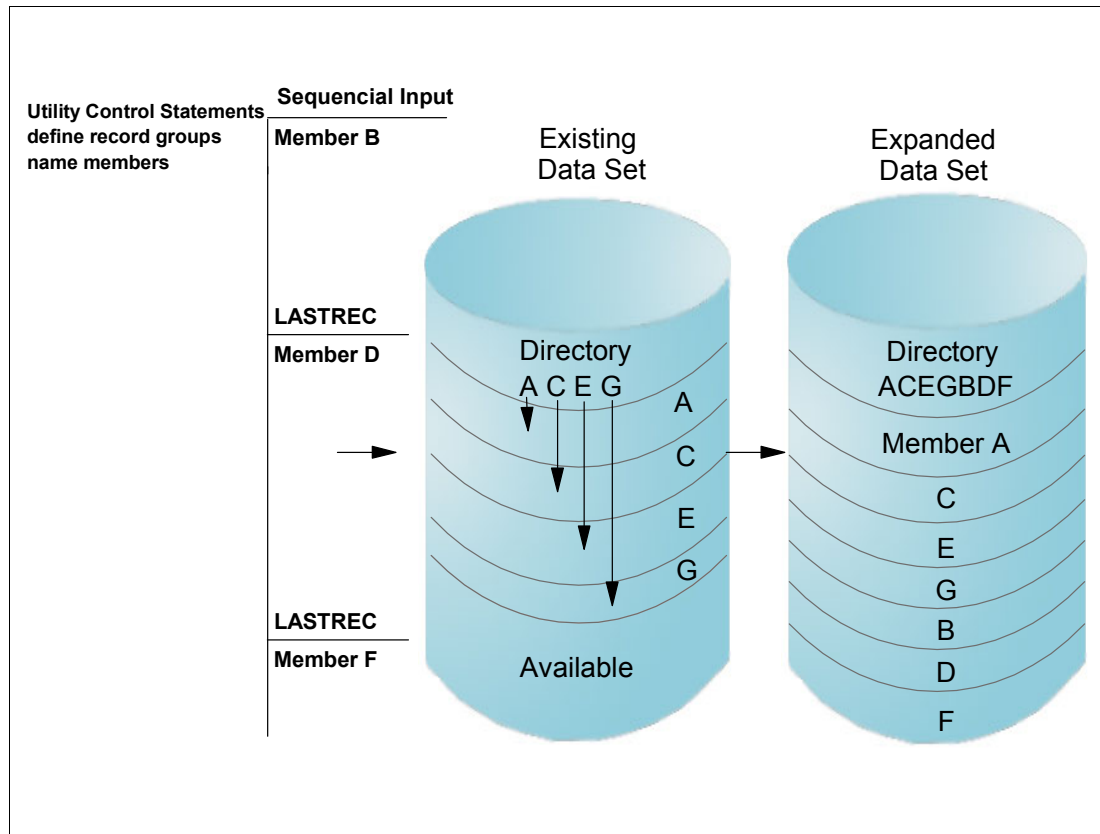


Figure 4-9 Adding members to a PDS using IEBGENER

### Adding members to a PDS

You can use IEBGENER to add members to a partitioned data set or PDSE. IEBGENER creates the members from sequential input and adds them to the data set. The merge operation—the ordering of the partitioned directory—is automatically performed by the program.

Figure 4-9 shows how sequential input is converted into members that are merged into an existing partitioned data set or PDSE. The list on the left side of the figure shows the sequential input that is to be merged with the partitioned data set or PDSE shown in the middle of the figure. Utility control statements are used to divide the sequential data set into record groups and to provide a member name for each record group. The right side of the figure shows the expanded partitioned data set or PDSE.

Note that members B, D, and F from the sequential data set were placed in available space, and that they are sequentially ordered in the partitioned directory.

## 4.9 Copying data to tape

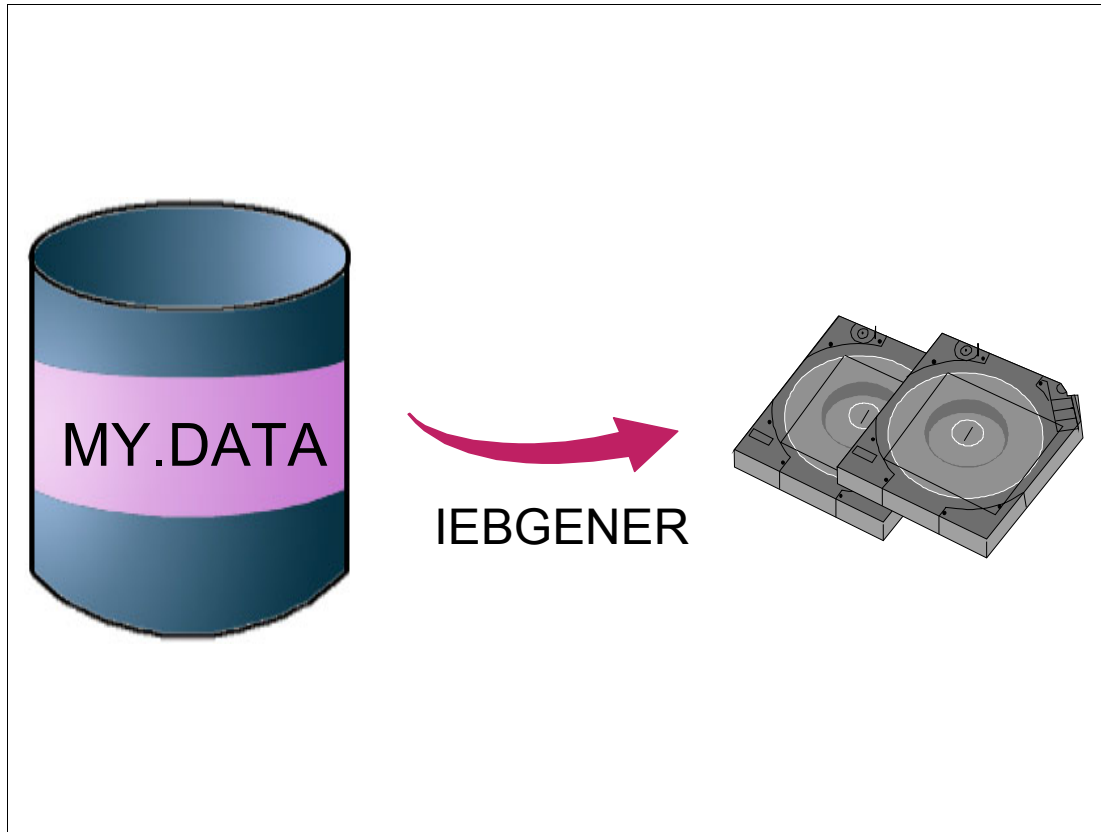


Figure 4-10 Copying data to tape

### Copying data to tape example

You can use IEBGENER to copy data to tape. Figure 4-11 copies the data set MY.DATA to an SL cartridge. The data set name on tape is MY.DATA.OUTPUT.

```
//DISKTOTP JOB ...
//STEP1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSNAME=MY.DATA,DISP=SHR
//SYSUT2 DD DSNAME=MY.DATA.OUTPUT,UNIT=3490,DISP=(,KEEP),
//          VOLUME=SER=IBM001,LABEL=(1,SL)
//SYSIN DD *
```

Figure 4-11 Copying data to tape with IEBGENER

For further information about IEBGENER, refer to *z/OS DFSMSdfp Utilities*, SC26-7414.



## 4.10 IEHLIST

```
//VTOCLIST JOB ...
//STEP1 EXEC PGM=IEHLIST
//SYSPRINT DD SYSOUT=A
//DD2 DD UNIT=3390,
      VOLUME=SER=TOTTSB,DISP=SHR
//SYSIN DD *
      LISTVTOC VOL=3390=APSG90,
      INDEXDSN=SYS1.VTOCIX.TOTTSB
/*
```

Figure 4-12 IEHLIST

### Using IEHLIST

IEHLIST is a system utility used to list entries in a CVOL, entries in the directory of one or more partitioned data sets or PDSEs, or entries in an indexed or non-indexed volume table of contents. Any number of listings can be requested in a single execution of the program.

IEHLIST lists all CVOL (SYSCTLG data set) entries that are part of the structure of a fully qualified data set name.

**Note:** IEHLIST will not list integrated catalog facility or VSAM catalogs. To list integrated catalog facility or VSAM catalogs, use access method services. For more information, see *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

IEHLIST can list up to 10 partitioned data set or PDSE directories at a time.

The directory of a partitioned data set is composed of variable-length records blocked into 256-byte blocks. Each directory block can contain one or more entries that reflect member or alias names and other attributes of the partitioned members. IEHLIST can list these blocks in edited and unedited format.

The directory of a PDSE, when listed, will have the same format as the directory of a partitioned data set.

## 4.11 IEHLIST LISTVTOC output

```

                                SYSTEMS SUPPORT UTILITIES---IEHLIST
DATE: 2004.224  TIME: 15.29.27
          CONTENTS OF VTOC ON VOL TOTTSB  <THIS VOLUME IS NOT SMS MANAGEI
          THERE IS A 2 LEVEL VTOC INDEX
          DATA SETS ARE LISTED IN ALPHANUMERIC ORDER
-----DATA SET NAME-----  CREATED  DATE.EXP  FILE TYPE  SI
$$$WF1                        1998.175   00.000   SEQUENTIAL
$$$WK1                        1998.176   00.000   SEQUENTIAL
ADLER.MW500MAP.MDL           2004.132   00.000   SEQUENTIAL
ALEX.DSNURCT.CNTL           1998.148   00.000   SEQUENTIAL
ALEX.SC53.SPFL0G4.LIST       1999.347   00.000   SEQUENTIAL
ALEX.SC61.SPFL0G3.LIST       1998.295   00.000   SEQUENTIAL
ALEX.SC62.SPFL0G1.LIST       1998.041   00.000   SEQUENTIAL
ALEX1.SC61.SPFL0G8.LIST      1998.107   00.000   SEQUENTIAL
ALEX1.SC62.SPFL0G4.LIST      1998.146   00.000   SEQUENTIAL
AMCHENG.ACTEST               1999.082   00.000   SEQUENTIAL
AMCHENG.BROADCAST           1999.070   00.000   SEQUENTIAL
AMCHENG.JCL                  1999.109   00.000   SEQUENTIAL
AMCHENG.SC55.ISPF42.ISPPROF  1999.070   00.000   PARTITIONED
ANANIA.SC67.ISPF42.ISPPROF  2003.120   00.000   PARTITIONED
ANDRE.WASDM05.SAVECFG        2003.199   00.000   SEQUENTIAL
ANDREW.SC62.ISPF42.ISPPROF  2002.268   00.000   PARTITIONED
ANGELO.DEVRXCFD.LIST         1999.126   00.000   SEQUENTIAL
ATTID8.BROADCAST            2000.250   00.000   SEQUENTIAL
WTSCPLX1.SC69.TRACE          1999.309   00.000   SEQUENTIAL
YCJRES4.DWJ.ISPFILE          2002.077   00.000   PARTITIONED
YYY.DBCSLCS.LOCAL.SIDLLM00   1999.123   00.000   PARTITIONED
ZOSR05.SEQ.Z05RD1.ETC        2004.191   00.000   SEQUENTIAL
THERE ARE 123 EMPTY CYLINDERS PLUS 1475 EMPTY TRACKS ON THIS VOLUME
THERE ARE 3710 BLANK DSCBS IN THE VTOC ON THIS VOLUME
THERE ARE 596 UNALLOCATED VIRS IN THE INDEX
***** BOTTOM OF DATA *****
```

Figure 4-13 IEHLIST LISTVTOC output

### Obtaining the VTOC listing

Running the job shown in Figure 4-12 on page 113 produces a SYSOUT very similar to that shown in Figure 4-13.

If you include the keyword `FORMAT` in the `LISTVTOC` parameter, you will have more detailed information about the DASD and about the data sets, and you can also specify the `DSNAME` that you want to request information about.

**Note:** This information is at the DASD volume level, and does not have any interaction with the catalog.

## 4.12 IEHINITT

### ❑ Initializing Tape Cartridges

- Create tape label - (EBCDIC or ASCII)
- Consider placement in an authorized library

```
//LABEL      JOB      ...
//STEP1     EXEC PGM=IEHINITT
//SYSPRINT  DD      SYSOUT=A
//LABEL1    DD      DCB=DEN=2,UNIT=(tape,1,DEFER)
//LABEL2    DD      DCB=DEN=3,UNIT=(tape,1,DEFER)
//SYSIN     DD      *
LABEL1     INITT     SER=TAPE1
LABEL2     INITT     SER=001234,NUMBTAPE=2
/*
```

### ❑ DFSMSrmm EDGINERS the newer alternative

Figure 4-14 IEHINITT

### IEHINITT utility

IEHINITT is a system utility used to place IBM volume label sets (no data set labels) written in EBCDIC (BCD for 7-track), or ISO/ANSI/FIPS volume label sets written in ASCII (American Standard Code for Information Interchange) onto any number of magnetic tapes mounted on one or more tape units.

**Note:** Because IEHINITT can overwrite previously labeled tapes regardless of expiration date and security protection, IBM recommends that the security administrator use PROGRAM protection with the following sequence of RACF commands:

```
RDEFINE PROGRAM IEHINITT ADDMEM('SYS1.LINKLIB'//NODPADCHK) UACC NONE
```

```
PERMIT IEHINITT CLASS(PROGRAM) ID(users or group who should have access)
ACCESS(READ)
```

```
SETROPTS WHEN(PROGRAM) REFRESH [omit REFRESH if you do not have this option
active previous]
```

IEHINITT should be moved into an authorized password-protected private library, and deleted from SYS1.LINKLIB.

To further protect against overwriting the wrong tape, IEHINITT asks the operator to verify each tape mount.

In the example, two groups of serial numbers, (001234, 001235, 001236, and 001334, 001335, 001336) are placed on six tape volumes. The labels are written in EBCDIC at 800 bits per inch. Each volume labeled is mounted, when it is required, on a single 9-track tape unit.

```
//LABEL3 JOB ...
//STEP1 EXEC PGM=IEHINITT
//SYSPRINT DD SYSOUT=A
//LABEL DD DCB=DEN=2,UNIT=(tape,1,DEFER)
//SYSIN DD *
LABEL INITT SER=001234,NUMTAPE=3
LABEL INITT SER=001334,NUMBTAPE=3
/*
```

Figure 4-15 IEHINITT example to write EBCDIC labels in different densities

In Figure 4-16, serial numbers 001234, 001244, 001254, 001264, 001274, and so forth are placed on eight tape volumes. The labels are written in EBCDIC at 800 bits per inch. Each volume labeled is mounted, when it is required, on one of four 9-track tape units.

```
//LABEL4 JOB ...
//STEP1 EXEC PGM=IEHINITT
//SYSPRINT DD SYSOUT=A
//LABEL DD DCB=DEN=2,UNIT=(tape,4,DEFER)
//SYSIN DD *
LABEL INITT SER=001234
LABEL INITT SER=001244
LABEL INITT SER=001254
LABEL INITT SER=001264
LABEL INITT SER=001274
LABEL INITT SER=001284
LABEL INITT SER=001294
LABEL INITT SER=001304
/*
```

Figure 4-16 IEHINITT Place serial number on eight tape volumes

### DFSMSrmm EDGINERS utility

The EDGINERS utility program verifies that the volume is mounted before writing a volume label on a labeled, unlabeled, or blank tape. EDGINERS checks security and volume ownership, and provides auditing. DFSMSrmm must know that the volume needs to be labelled. If the labelled volume is undefined, then DFSMSrmm defines it to DFSMSrmm and can create RACF volume security protection.

Detailed procedures for using the program are described in *z/OS DFSMSrmm Implementation and Customization Guide*, SC26-7405.

**Note:** DFSMSrmm is an optional priced feature of DFSMS. That means that EDGINERS can only be used when DFSMSrmm is licensed. If DFSMSrmm is licensed, IBM recommends that you use EDGINERS for tape initialization instead of using IEHINITT.

## 4.13 IEFBR14

```
//DATASETS  JOB  FREEMAN,MSGLEVEL=1
//STEP1     EXEC PGM=IEFBR14
//D1       DD DSN=ABC,
//         DISP=(NEW,CATLG),UNIT=3390,
//         VOL=SER=333001,
//         SPACE=(CYL,(12,1,1),)
```

Figure 4-17 IEFBR14

### IEFBR14 uses

IEFBR14 is not a utility program. It is a two-line program that clears register 15, thus passing a return code of 0. It then branches to the address in register 14, which returns control to the system. So in other words—this program is dummy program. It can be used in a step to force MVS (specifically, the initiator) to process the JCL code and execute functions such as the following:

- ▶ Checking all job control statements in the step for syntax
- ▶ Allocating direct access space for data sets
- ▶ Performing data set dispositions

**Note:** Although the system allocates space for data sets, it does not initialize the new data sets. Therefore, any attempt to read from one of these new data sets in a subsequent step may produce unpredictable results. Also, IBM does not recommend allocation of multi-volume data sets while executing IEFBR14.

## 4.14 Access method services

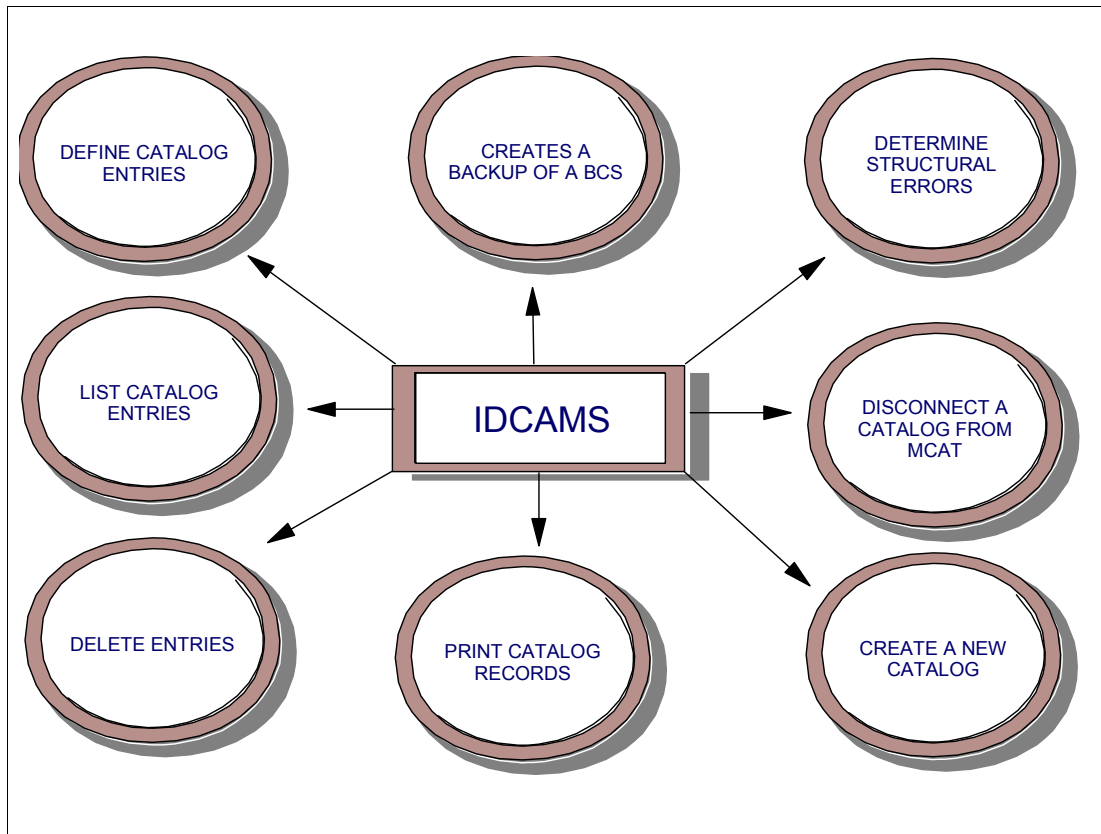


Figure 4-18 Access method services

### Access method services

Access method services is a utility you can use to establish and maintain catalogs and data sets (VSAM and non-VSAM).

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 (it looks like a language). 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 Storage Management Subsystem (SMS) automates many access method services commands and their parameters. 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 greater detail in “System-managed storage” on page 189.

## Invoking 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 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

You can run the IDCAMS program (the execution part of access method services) and include the command and its parameters as input to the program. You can also call the IDCAMS program from within another program and pass the command and its parameters to the IDCAMS program.

Time Sharing Option (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 topic D.0, 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 4-19.

```
//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 4-19 JCL statements to call IDCAMS

## From a Time Sharing Option (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 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. Access method services processes the command completely before TSO lets you continue processing. Except for **ALLOCATE**, all the access method services functional commands are supported in a TSO environment.

For more information, refer to *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



## 4.15 AMS functional commands

- ❑ DEFINE CLUSTER: creates /catalog VSAM data sets
- ❑ DEFINE GENERATIONDATAGROUP: catalog GDG data sets
- ❑ DEFINE PAGESPACE: creates/catalog page data sets
- ❑ EXPORT: export VSAM DS, AI or ICF
- ❑ IMPORT: import VSAM DS, AI or ICF
- ❑ LISTCAT: list catalog entries
- ❑ REPRO: copy VSAM, non-VSAM and catalogs
- ❑ VERIFY: corrects mismatches between catalogs and data sets

Figure 4-20 Functional commands

### IDCAMS functional commands

Table 4-3 lists and describes the utilization of the functional commands.

Table 4-3 Functional commands

Command	Functions
ALLOCATE	Allocates Virtual Storage Access Method (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 data sets.
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.
DEFINE CLUSTER	Creates KSDS, ESDS, RRDS, VRRDS and linear VSAM data sets.

Command	Functions
DEFINE GENERATIONDATAGROUP	Defines a catalog entry for a generation data group.
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 data sets, and non-VSAM data sets.
DIAGNOSE	Scans an integrated catalog facility basic catalog structure (BCS) or a VSAM volume data set (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 key-sequence data set cluster.
EXPORT	Disconnects user catalogs, and exports VSAM data sets and integrated catalog facility catalogs.
EXPORT DISCONNECT	Disconnects a user catalog.
IMPORT	Connects user catalogs, and imports VSAM data sets and integrated catalog facility catalogs.
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:  Copies VSAM and non-VSAM data sets, user catalogs, master catalogs, and volume catalogs.  Splits integrated catalog facility catalog entries between two catalogs.  Merges integrated catalog facility catalog entries into another integrated catalog facility user or master catalog.  Merges tape library catalog entries from one volume catalog into another volume catalog.
SHCDS	Lists SMSVSAM recovery related to online applications and spheres accessed in record-level subscriber (RLS) mode.
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.

For a complete description of all AMS commands, refer to *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394.

## 4.16 AMS modal commands

- ❑ IF-THEN-ELSE command sequence controls command execution on the basis of conditional codes
- ❑ NULL command specifies no action be taken
- ❑ DO-END command sequence specifies more than one functional access method services command and its parameters
- ❑ SET command resets condition codes
- ❑ CANCEL command terminates processing of the current job step
- ❑ PARM command specifies diagnostic aids and printed output options.

Figure 4-21 AMS modal commands

### Conditional execution of functional AMS commands

The access method services modal commands are used for the conditional execution of functional commands. Time Sharing Option (TSO) users can use functional commands only.

**Note:** Figure 4-21 contains a list and brief descriptions of the AMS modal commands. These commands cannot be used when access method services is run in TSO. See *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394, for a complete description of the AMS modal commands.

## 4.17 Data Collection Facility (DCOLLECT)

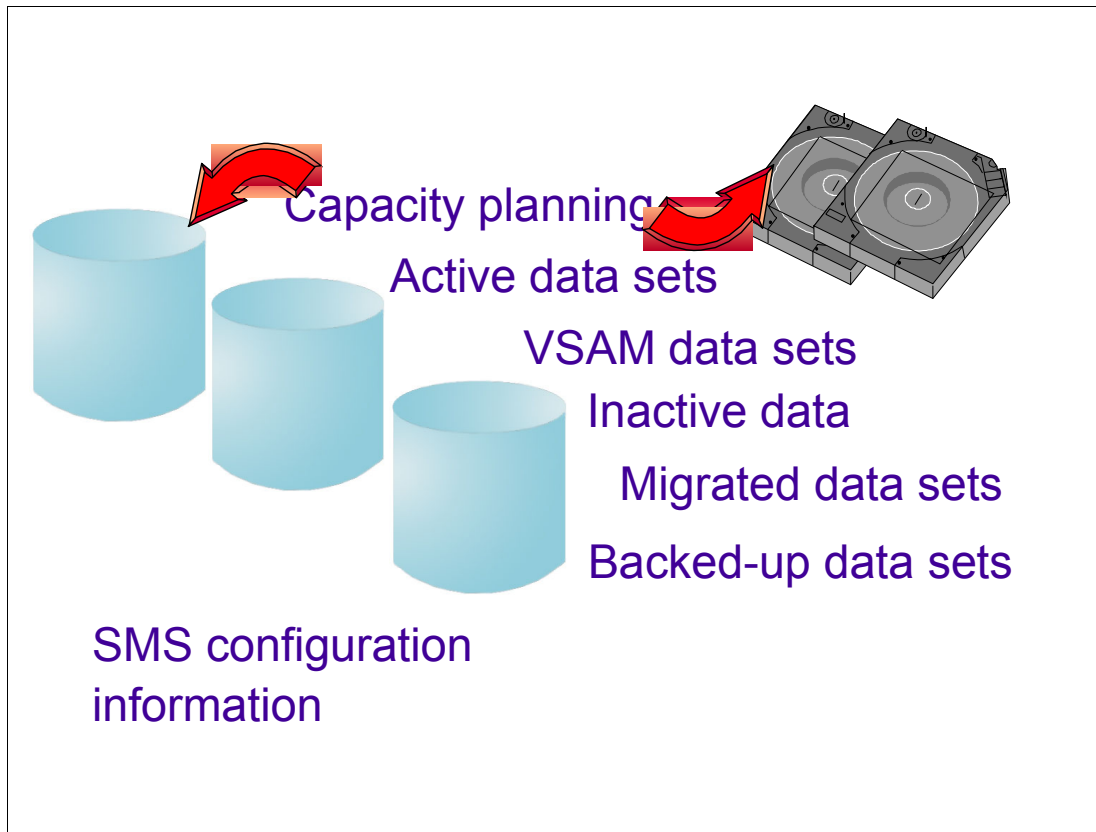


Figure 4-22 Data Collection Facility

### DCOLLECT command

**DCOLLECT** is an IDCAMS command. An installation may use this command to collect information related to:

- ▶ Active data set storage
- ▶ VSAM association name
- ▶ Volume usage
- ▶ DFSMSHsm backup and migration storage
- ▶ DFSMSHsm DASD and tape capacity planning

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

The output of **DCOLLECT** is a sequential data set. Installation can generate reports from the collected data relating to space management, capacity planning, and cost accounting using:

- ▶ Tivoli Decision Support for OS/390, that provides support for **DCOLLECT** data by including a starter set of log, summary, and parameter tables and views. It also includes a report dialog with a variety of predefined reports. For more information see *Tivoli Decision Support for OS/390 System Performance Feature Reference Volume 1*, SH19-6819.
- ▶ DB2 can also be used to hold **DCOLLECT** data. Data resides in a relational database structure, and can be presented to users in a table format.
- ▶ User-written applications can manipulate the **DCOLLECT** sequential output data set to generate reports.

## 4.18 Generation data groups (GDG)

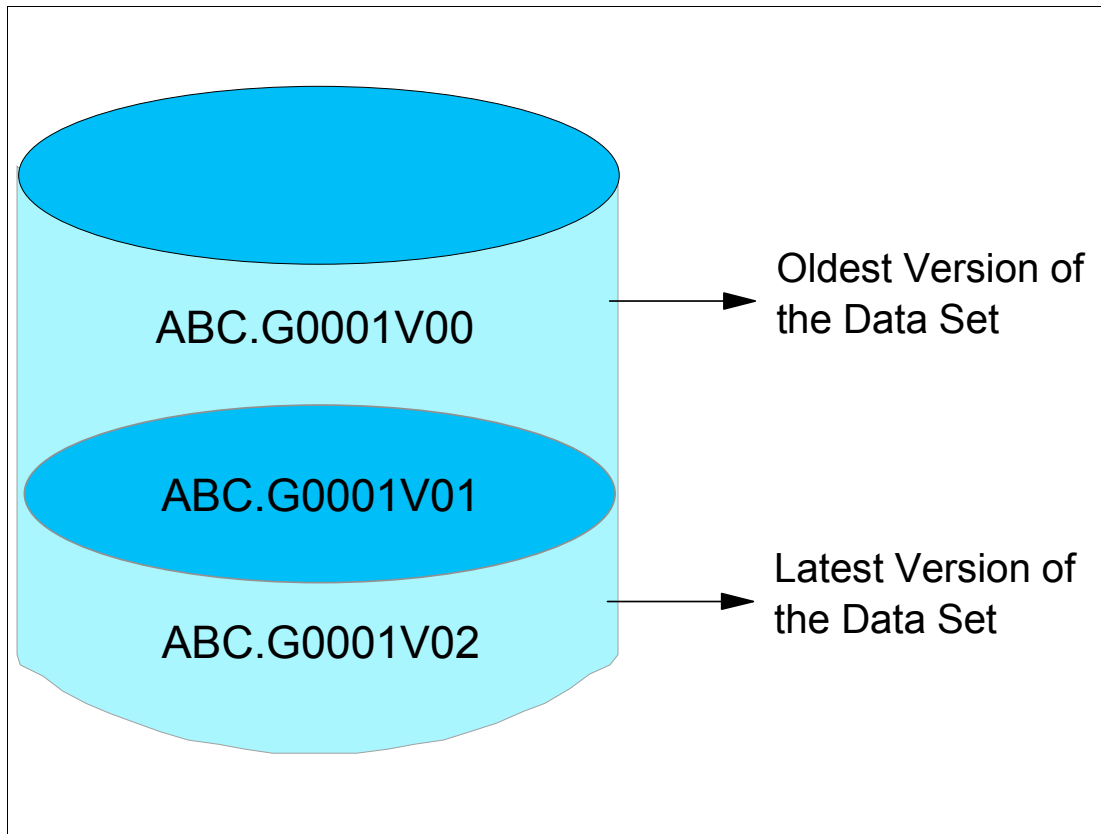


Figure 4-23 Generation data groups (GDG)

### Generation data groups

You can catalog successive updates or generations of related data sets. They are called generation data groups (GDG). Each data set within a GDG is called a generation data set 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, direct, or indexed sequential (an old and less-used data set organization, replaced by VSAM KSDS). They *cannot* be partitioned, HFS, or VSAM. The same GDG may contain SMS and non-SMS data sets.

There are advantages to grouping related data sets. For example, the catalog management routines can refer to the information in a special index called a generation index in the catalog. Thus:

- ▶ All of the data sets in the group can be referred to by a common name.
- ▶ The operating system is able to keep the generations in chronological order.
- ▶ Outdated or obsolete generations can be automatically deleted by the operating system.

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

Generation data sets have sequentially ordered absolute and relative names that represent their age. The catalog management routines use the absolute generation name. Older data sets have smaller absolute numbers. The relative name is a signed integer used to refer to

the latest (0) generation, the next to latest (-1) generation, and so forth. For example, a data set name LAB.PAYROLL(0) refers to the most recent data set of the group; LAB.PAYROLL(-1) refers to the second most recent data set; and so forth. The relative number can also be used to catalog a new generation (+1).

If you create a generation data set with a relative generation number of (+1), the system recognizes any subsequent reference to (+1) throughout the job as having the same absolute generation number.

A GDG base is allocated in an integrated catalog facility or VSAM catalog before the generation data sets are cataloged. Each GDG is represented by a GDG base entry. Use the AMS **DEFINE** command to allocate the GDG base.

The model DSCB must exist on the GDG catalog volume.

You should only use the low-level qualifier GxxxxVyy, structure name, where xxxx and yy are numbers, in the names of generation data sets (to be seen later). You can define a data set with GxxxxVyy as the low-level qualifier of non-generation data sets only if a generation data group with the same base name does not exist. However, we recommend that you restrict GxxxxVyy qualifiers to generation data sets, to avoid confusing generation data sets with other types of non-VSAM data sets.

## 4.19 Defining a generation data group

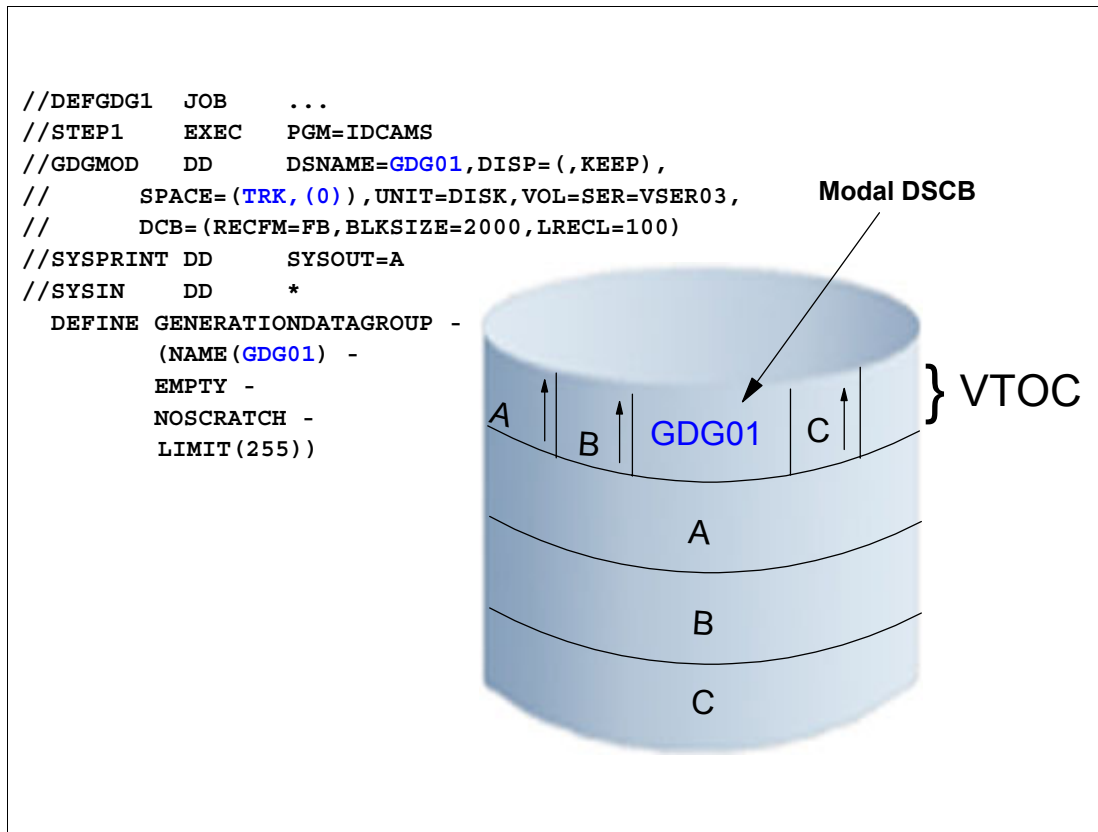


Figure 4-24 Defining a GDG

### Defining a generation data group

The **DEFINE GENERATIONDATAGROUP** command creates a catalog entry for a generation data group (GDG).

Figure 4-25 shows the JCL to define a GDG.

```
//DEFGDG1 JOB      ...
//STEP1   EXEC     PGM=IDCAMS
//GDGMOD  DD       DSNAME=GDG01,DISP=(,KEEP),
//          SPACE=(TRK,(0)),UNIT=DISK,VOL=SER=VSER03,
//          DCB=(RECFM=FB,BLKSIZE=2000,LRECL=100)
//SYSPRINT DD      SYSOUT=A
//SYSIN   DD       *
DEFINE GENERATIONDATAGROUP -
  (NAME(GDG01) -
  NOEMPTY -
  NOSCRATCH -
  LIMIT(255) )
/*
```

Figure 4-25 JCL to define a GDG catalog entry

The **DEFINE GENERATIONDATAGROUP** command defines a GDG base catalog entry GDG01.

Its parameters are:

- ▶ **NAME** specifies the name of the GDG, GDG01. Each GDS in the group will have the name *GDG01.GxxxxVyy*, where *xxxx* is the generation number and *yy* is the version number.
- ▶ **NOEMPTY** specifies that only the oldest generation data set is to be uncataloged when the maximum is reached (recommended).
- ▶ **EMPTY** specifies that all data sets in the group are to be uncataloged by VSAM when the group reaches the maximum number of data sets (as specified by the LIMIT parameter) and one more GDS is added to the group.
- ▶ **NOSCRATCH** specifies that when a data set is uncataloged, its DSCB is not to be removed from its volume's VTOC. Therefore, even if a data set is uncataloged, its records can be accessed when it is allocated to a job step with the appropriate JCL DD statement.
- ▶ **LIMIT** specifies that the maximum number of GDG data sets in the group is 255. This parameter is required.

Figure 4-26 shows a generation data set is defined within the GDG by using JCL statements.

```
//DEFGDG2 JOB    ...
//STEP1   EXEC   PGM=IEFBR14
//GDGDD1  DD     DSN=GDG01(+1),DISP=(NEW,CATLG),
//         SPACE=(TRK,(10,5)),VOL=SER=VSER03,
//         UNIT=DISK
//SYSPRINT DD    SYSOUT=A
//SYSIN   DD     *
/*
```

Figure 4-26 JCL to define a generation data set

The job DEFGDG2 allocates space and catalogs a GDG data set in the newly-defined GDG. The job control statement GDGDD1 DD specifies the GDG data set in the GDG.



## 4.20 Absolute generation and version numbers

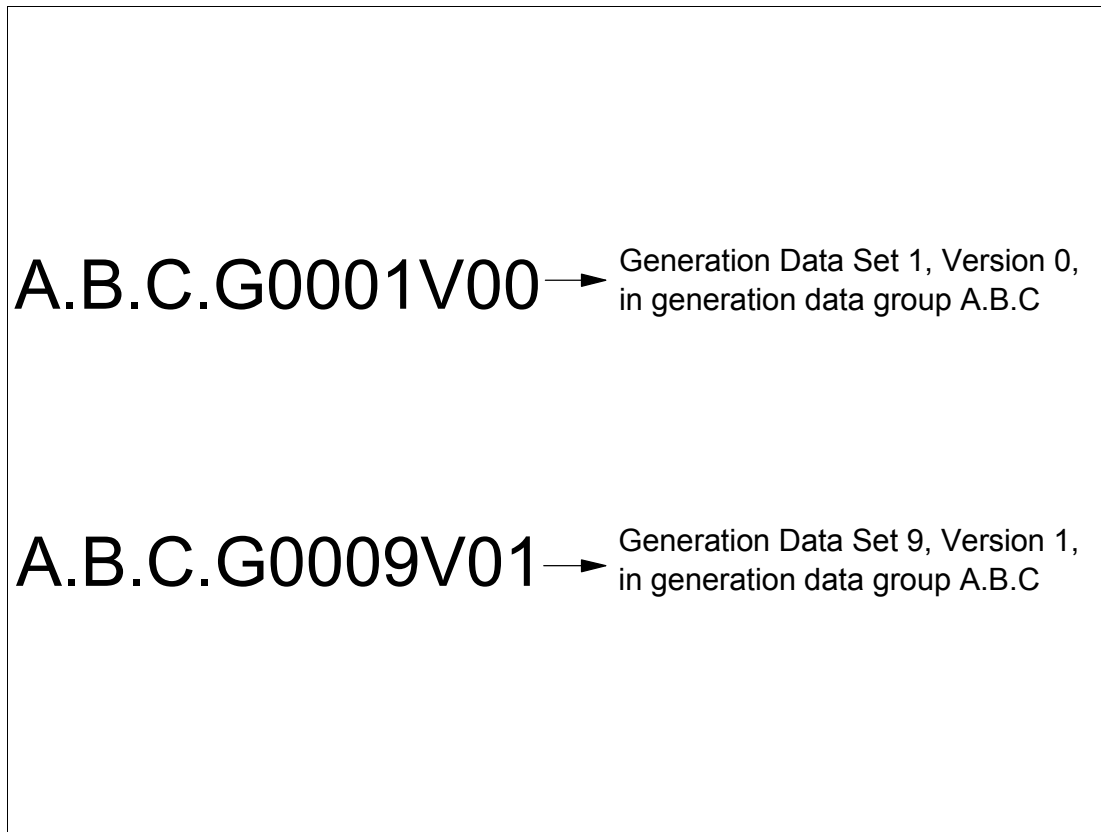


Figure 4-27 Absolute generation and version numbers

### Identifying an specific generation of a GDG

An absolute generation and version number is used to identify a specific generation of a generation data group. A same generation data set may 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, replace generation 2, version 0, with generation 2, version 1. Only one version is kept for each generation.

The generation and version number are in the form *GxxxxVyy*, where *xxxx* is an unsigned four-digit decimal generation number (0001 through 9999) and *yy* is an unsigned two-digit decimal version number (00 through 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; that is, there can be 9,999 generations. Each generation can have 100 versions. The system automatically maintains the generation number.

The number of generations kept depends on the size of the generation index. For example, if the size of the generation index allows ten entries (parameter **LIMIT** in **AMS DEFINE**), the ten latest generations can be maintained in the generation data group (parameter **NOEMPTY** in **AMS DEFINE**).

You can catalog a generation 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.

## 4.21 Relative generation numbers

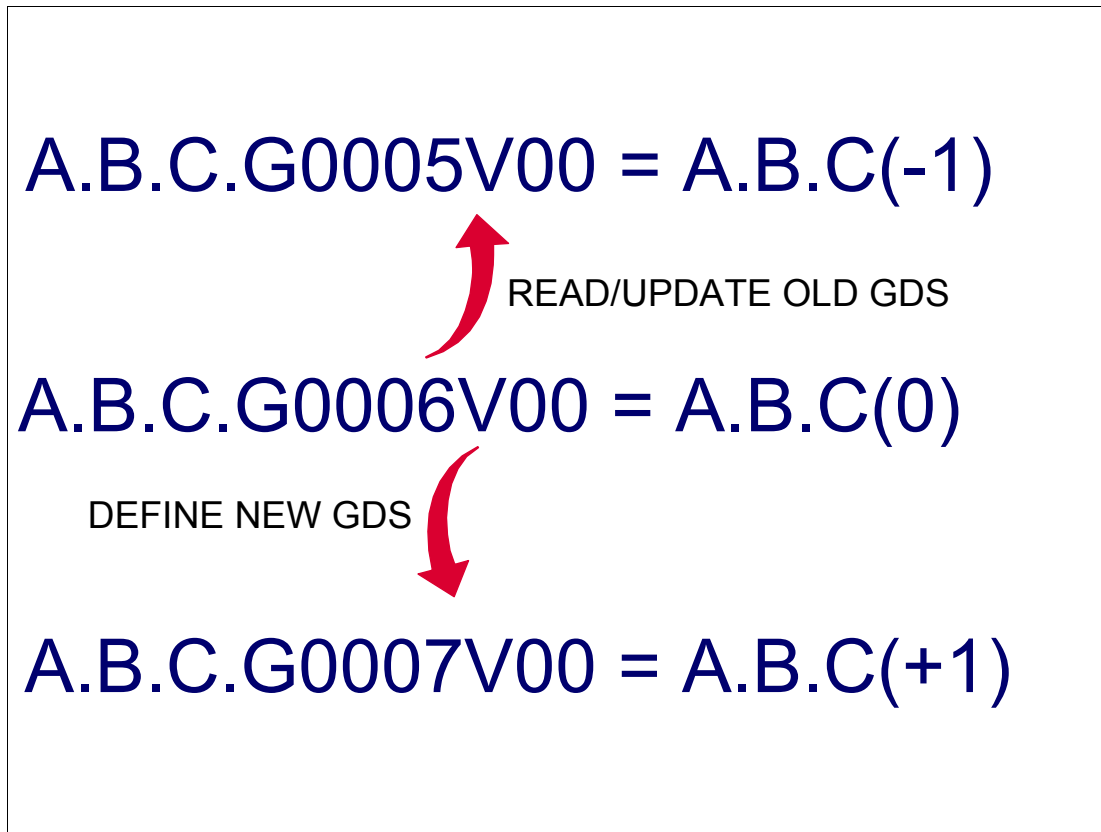


Figure 4-28 Relative generation numbers

### 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; for example, 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.

### **Rolled in and rolled off**

When a generation data group contains its maximum number of active generation data sets, defined in the **LIMIT** parameter, and a new generation data set is rolled in at end-of-job step, the oldest generation data set is rolled off and is no longer active. If a generation data group is defined using **DEFINE GENERATIONDATAGROUP EMPTY** and is at its limit, then when a new generation data set is rolled in, all the currently active generation data sets are rolled off.

The parameters you specify on the **DEFINE GENERATIONDATAGROUP** command determines what happens to rolled-off generation data sets. For example, if you specify the **SCRATCH** parameter, the generation data set is scratched 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.

Generation data sets 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. Generation data sets in a deferred roll-in state can be referred to by their absolute generation numbers. You can use the access method service command **ALTER ROLLIN** to roll in these generation data sets.

For further information about Generation Data Groups, refer to *z/OS DFSMS: Using Data Sets*, SC26-7410.

## 4.22 Access method

- ❑ Performs bufferization
- ❑ Synchronizes between your task and the I/O operation (Wait/Post mechanism)
- ❑ Writes the channel program
- ❑ Optimizes the performance characteristics of the control unit (such as caching, data stripe)
- ❑ Compress and uncompress I/O data
- ❑ Executes software error recovery

Figure 4-29 Access method functions

### Access method functions

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

- ▶ By managing data buffers
- ▶ 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
- ▶ Executing software error recovery

An access method defines the technique by which the data is stored and retrieved. DFSMS access methods have their own data set structures for organizing data, macros to define and process data sets, and utility programs to process data sets.

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 organization can be used to process a data set organized in a different manner. For example, a sequential data set (not extended format data set) created using BSAM can be processed by the basic direct access method (BDAM), and vice versa.

## 4.23 Major DFSMS access methods

- ❑ The major DFSMS access methods:
  - Basic Direct Access Method (BDAM)
  - Object Access Method (OAM)
  - Basic Partitioned Access Method (BPAM)
  - Basic Direct Access Method (BDAM)
  - Basic Sequential Access Method (BSAM)
  - Queued Sequential Access Method (QSAM)
  - Virtual Storage Access Method (VSAM)

Figure 4-30 Major DFSMS access methods

### Basic Direct Access Method (BDAM)

BDAM arranges records in any sequence 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 where a search for the record is to begin. Data sets organized this way are called direct data sets.

**Note:** IBM does not recommend using BDAM because it tends to require using device-dependent code. In addition, using keys is much less efficient than in the virtual sequential access method (VSAM). BDAM is supported by DFSMS only to enable compatibility with other IBM operating systems. For more information, refer to Appendix C, “Processing Direct Data Sets” in *z/OS DFSMS: Using Data Sets*, SC26-7410.

### Object Access Method (OAM)

OAM processes very large named byte streams (objects) that have no record boundary or other internal orientation. These objects can be recorded in a DB2 data base or on an optical storage volume. For information on 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.

## 4.24 Basic Partitioned Access Method (BPAM)

- ❑ BPAM allows access to partitioned data sets
- ❑ A partitioned data set includes a directory that relates member names to locations within the data set, the directory is used to retrieve individual members
- ❑ A member is accessed by BPAM as a sequential file
- ❑ The contents of a member can be:
  - Data
  - Programs
  - Tables (as ISPF)
  - Procedures (as JCL)
- ❑ There are two types: PDS and PDSE

Figure 4-31 BPAM to access PDS and PDSE

### BPAM to access PDS and PDSE

Basic Partitioned Access Method (BPAM) arranges records as members of a partitioned data set (PDS) or a partitioned data set extended (PDSE) on DASD. You can view each member like a sequential data set. A partitioned data set or PDSE includes a directory that relates member names to locations within the data set. The directory is used to retrieve individual members. For program libraries (load modules and program objects), the directory contains program attributes required to load and rebind the member.

## 4.25 PDS data organization

- ❑ Advantages of the PDS organization:
  - Easier management: Processed by member or a whole. Members can be concatenated and processed as sequential files
  - Space savings: small members fit in one DASD track
  - Good usability: Easily accessed via JCL, ISPF, TSO
- ❑ Required Improvements for the PDS organization:
  - When a member is deleted the space is released. No need to compress
  - Expandable directory size
  - Improved directory and member integrity
  - Better performance for directory search
  - Improving sharing facilities

Figure 4-32 PDS data organization

### Partitioned data set (PDS)

PDS is an old MVS data organization that offers such useful features as:

- ▶ Easier management

Grouping of related data sets under a single name makes MVS data management easier. Files stored as members of a PDS can be processed either individually or all the members can be processed as a unit.
- ▶ Space savings

Small members fit in just one DASD track.
- ▶ Good usability

Members of a PDS can be used as sequential data sets, and they can be concatenated to sequential data sets. They are also easy to create with JCL, or ISPF, and they are easy to manipulate with ISPF utilities or TSO commands.

However, there are a few requirements for improvement regarding PDS organization:

- ▶ There is no mechanism to reuse the area that contained a deleted or rewritten member.

This unused space must be reclaimed by the use of the IEBCOPY utility function called compression.
- ▶ Directory size is not expandable, causing an overflow exposure.



The area for members may grow using secondary allocations. This is not true for the directory.

- ▶ A PDS has no mechanism to prevent a directory from being overwritten if a program mistakenly opens it for sequential output.

If this happens, the directory is destroyed and all the members are lost.

Also, PDS DCB attributes can be easily changed by mistake. If you add a member whose DCB characteristics differ from those of the other members, you will change the DCB attributes of the entire PDS, and all the old members will become unusable.

- ▶ Better directory search time.

Entries in the directory are physically ordered by the collating sequence of the names in the members they are pointing to. Any inclusion may cause the full rearrange of the entries.

There is also no index to the directory entries. The search is sequential using a CKD format. If the directory is big, the I/O operation takes more time.

- ▶ Improved sharing facilities.

To update a member of a PDS, you need exclusive access to the entire data set.

All these improvements require almost total compatibility at the program level and the user level with the old PDS.

## 4.26 Partitioned data set extended (PDSE)

- ❑ Comprised of a directory and members allocated in preformatted equal-size 4 KB pages
  - A member may not be stored in contiguous pages
- ❑ The directory has an index structure and is physically mixed with the members
- ❑ Must be an SMS-managed data set
- ❑ Directory is buffered in data space, members in a hiperspace ESO (both managed by SYSBMAS AS) for performance
- ❑ Cross-system locks guarantee integrity when sharing PDSEs among MVSs (managed by SMXC AS)
- ❑ Member may contain data, load modules (called program objects, tables)

Figure 4-33 PDSE structure

### PDSE advantages

Figure 4-33 describes the PDSE structure. The advantages of PDSE when compared with PDS are:

- ▶ Space is reclaimed without a compress. PDSE automatically reuses space, without needing an IEBCOPY compress. A list of available space is kept in the directory. When a PDSE member is updated or replaced, it is written in the first available space. This is either at the end of the data set, or in a space in the middle of the data set marked for reuse.

This space need not be contiguous. The objective of the space reuse algorithm is not to extend the data set unnecessarily.

- ▶ The directory can grow dynamically as the data set expands. Logically, a PDSE directory looks the same as 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.

The directory is like a KSDS index structure (KSDS is covered in “Key sequenced data set (KSDS)” on page 152), making a search much faster. It cannot be overwritten by being opened for sequential output.

- ▶ If you try to add a member with DCB characteristics that differ from the rest of the members, you will get an error.
- ▶ You can open a PDSE member for output or update, without locking the entire data set. The sharing control is at member level, not the data set level.

**Restriction:** You cannot use a PDSE for certain system data sets that are opened in the IPL/NIP time frame.

## 4.27 Sequential access methods

- ❑ Sequential access data organization
  - Physical sequential
  - Extended format
    - Compressed data sets
    - Data striped data sets
  - Hierarchical File System (HFS)
- ❑ These organizations are accessed by the sequential access methods:
  - Queued Access Method (QSAM)
  - Basic Access Method (BSAM)
  - POSIX S/390 UNIX System Services calls

Figure 4-34 Sequential access methods

### BSAM and QSAM

There are two sequential access methods, Basic Sequential Access Method (BSAM) and Queued Sequential Access Method (QSAM). Both methods access data organized in a physical sequential manner; the physical records (containing logical records) are stored sequentially in the order in which they are entered.

A special type of this organization is the *extended format data set*. Extended format data sets have a different internal storage format from a sequential data set that is not extended (fixed block with a 32-byte suffix). This storage format gives extended format data sets additional usability and availability characteristics:

- ▶ They can be allocated in the compressed format (can be referred to as a *compressed format data set*). A compressed format data set is a type of extended format data set that has an internal storage format that allows for data compression.
- ▶ They allow data striping, that is, a multivolume sequential file where data may be accessed in parallel.
- ▶ They are able to recover from padding error situations.

Extended format data sets must be SMS-managed and must reside on DASD. You cannot use an extended format data set for certain system data sets.

Another type of this organization is the *Hierarchical File System*. HFS files are POSIX-conforming files that reside in an HFS data set. They are byte-oriented rather than

record-oriented, as are MVS files. They are identified and accessed by specifying the path leading to them. Programs can access the information in HFS files through z/OS UNIX system calls, such as `open(pathname)`, `read(file descriptor)`, and `write(file descriptor)`.

Programs can also access the information in HFS files through the MVS BSAM, QSAM, and VSAM (Virtual Storage Access Method) access methods. When using BSAM or QSAM, an HFS file is simulated as a multi-volume sequential data set. When using VSAM, an HFS file is simulated as an ESDS. HFS data sets are:

- ▶ Supported by standard DADSM create, rename, and scratch
- ▶ Supported by DFSMSHsm for dump/restore and migrate/recall if DFSMSdss is used as the data mover
- ▶ Not supported by IEBCOPY or the DFSMSdss COPY function

The differences between QSAM and BSAM are:

- ▶ QSAM de-blocks logical records and does look-ahead reads (anticipates reads). In BSAM, these tasks are performed by the calling program.
- ▶ QSAM synchronizes the task with I/O operation (places the task in wait along the I/O operation). In BSAM, this task is performed by the calling program (macro CHECK).

## 4.28 Virtual Storage Access Method (VSAM)

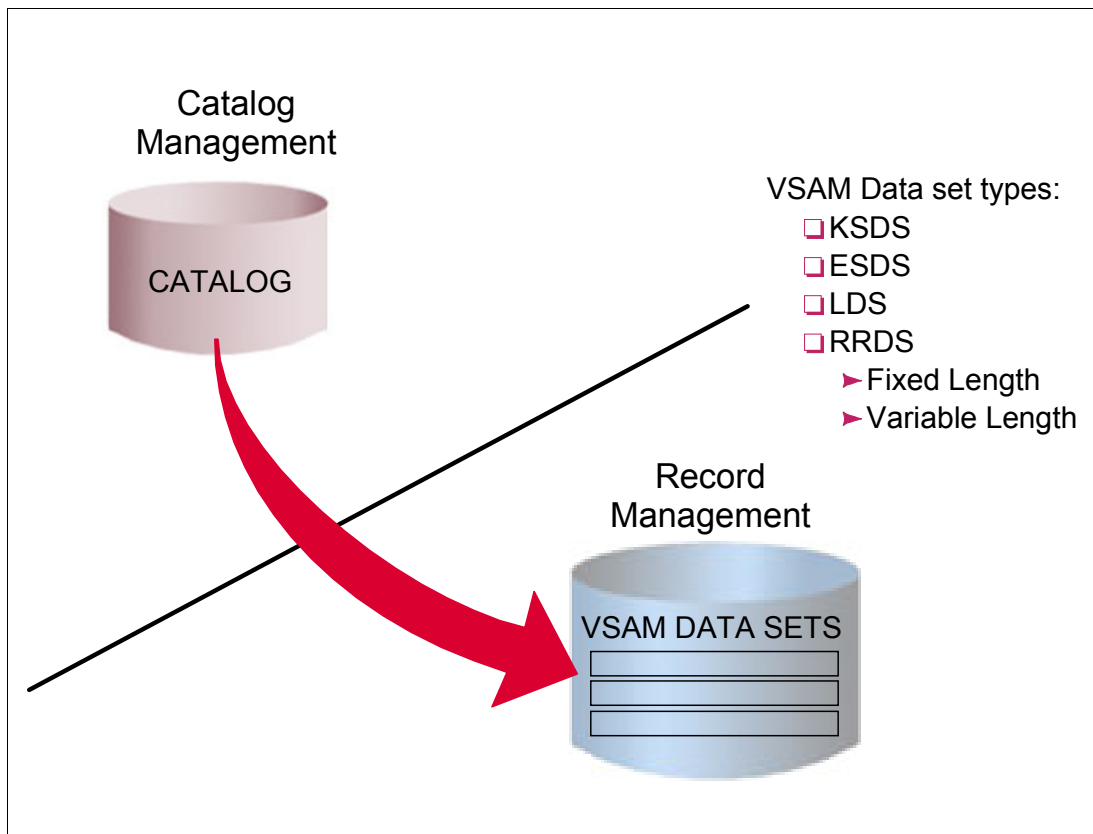


Figure 4-35 Virtual Storage Access Method

### VSAM

VSAM is an access method service used to organize data and maintain information about the data in a catalog.

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 four types of data sets:
  - Key-sequenced (KSDS)
  - Entry-sequenced (ESDS)
  - Linear (LDS)
  - Relative record with fixed or variable length (RRDS)

The primary difference among these types of data sets is the way in which 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 is cataloged for easy retrieval and is stored in one of four types of data sets.

## 4.29 VSAM resource pool and buffering techniques

- ❑ VSAM resource pool is formed by:
  - I/O control blocks
  - Buffer pool (set of equal-sized buffers)
- ❑ VSAM resource pool can be shared by VSAM sphere data sets, improving the effectiveness of these buffers
- ❑ Three types of VSAM resource pools:
  - Non-shared resource (NSR)
  - Local shared resource (LSR)
  - Global shared resource (GSR)

Figure 4-36 VSAM resource pool

### VSAM resource pool

VSAM resource pool is a set of VSAM I/O control blocks plus a buffer pool. A *buffer pool* is a collection of same-sized I/O buffers plus control information describing the occupancy of such buffers. The objective of a buffer pool is to avoid I/O operations and consequently to improve performance.

For more efficient use of virtual storage, buffer pools can be shared among data sets (except linear data sets), using globally or locally shared buffer pools. There are three types of resource pools, depending on the type of the associated buffer pool. These options are declared in the ACB macro of the VSAM data set (MACRF keyword) and are described in the following section.

### Non-shared resource (NSR)

An NSR resource pool has the following characteristics:

- ▶ Implicitly constructed at data set open time.
- ▶ Not shared among VSAM data sets.
- ▶ 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 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.
- ▶ Buffers are *not* revisited.
- ▶ There is dynamic addition of strings. Strings are like cursors; each string represents a position in the data set for the requested record.

NSR is the VSAM default and it is used by high level languages (HLL). As buffers are managed via a sequential algorithm, NSR is not the best choice for random processing. For applications using NSR, consider using system-managed buffering (SMB); see “System-managed buffering (SMB)” on page 145.

### **Local shared resource (LSR)**

An LSR resource pool is suitable for random processing. The LSR has the following characteristics:

- ▶ Shared among VSAM data sets accessed by tasks in the same address space.
- ▶ Located in the private area and ESO hiperspace. With hiperspace, VSAM buffers are located in expanded storage to improve the processing of VSAM data sets.
- ▶ Explicitly constructed via macro BLDVRP, before the OPEN.
- ▶ Buffers are managed via the last recently used (LRU) algorithm.
- ▶ Buffers *are* revisited.

### **Global shared resource (GSR)**

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 this must be in the supervisor state.
- ▶ Buffers cannot use hiperspace.
- ▶ The separate index resource pools are not supported for GSR.

GSR has many disadvantages, so you should consider the use of VSAM RLS instead.



## 4.30 System-managed buffering (SMB)

- ❑ Only for SMS-managed extended format data sets
  - RECORD\_ACCESS\_BIAS in DATACLASS
  - ACCBIAS subparameter of AMP, in JCL DD statement
- ❑ Only for applications using NSR buffering management
- ❑ For SYSTEM, VSAM decisions based on MACRF parameter of ACB
- ❑ Optimum number of index and data buffers
- ❑ For random access, VSAM changes buffering management technique from NSR to LSR

Figure 4-37 System-managed buffering

### System-managed buffering (SMB)

SMB is a feature of DFSMSdfp and was introduced in DFSMS V1R4. SMB enables VSAM to:

- ▶ Determine the optimum number of index and data buffers
- ▶ Change the buffer management declared in the application program, in the ACB MACRF parameter, from NSR to LSR.

Usually SMB allocates many more buffers than 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; no application changes are needed.

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

- ▶ SMS-managed data set
- ▶ Extended format VSAM data sets:
  - Data set name type **EXT** in the data class
- ▶ Application opens the data set for NSR processing

SMB is invoked or disabled through one of the following methods:

1. Record Access Bias data class field
2. 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 invoked when **SYSTEM** or an SMB processing technique is used in the fields described. SMB is disabled when **USER** is entered instead (USER is the default). Since JCL information takes precedence over data class information, installations can enable or disable SMB for some executions.

The SMB processing techniques are:

- DO** SMB optimizes for totally random record access. When this technique is used, VSAM changes the buffering management from NSR to LSR.
- DW** The majority is direct access to records, with some sequential.
- SO** Totally sequential access.
- SW** The majority is sequential access, with some direct access to records.

When **SYSTEM** is used in JCL or in the data class, SMB chooses the processing technique based on the MACRF parameter of the ACB.

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

## 4.31 VSAM terminology and concepts

- ❑ Logical record
  - Key field
- ❑ Physical record
- ❑ Control interval
  - Record definition field
  - Control interval definition field
- ❑ Control area
- ❑ Components
  - Data component
  - Index component
    - Index set
    - Sequence set
- ❑ Cluster
- ❑ Alternate Index
- ❑ Sphere

Figure 4-38 VSAM terminology and concepts

### Logical record

A *logical record* is a unit of information used to store data in a VSAM data set. 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.

### Physical record

A *physical record* is device-dependent. VSAM calculates the physical record size at the time the data set is defined. All physical records have the same length. A physical record is also referred to as a *physical block* or simply a *block*.

## 4.32 Control interval (CI)

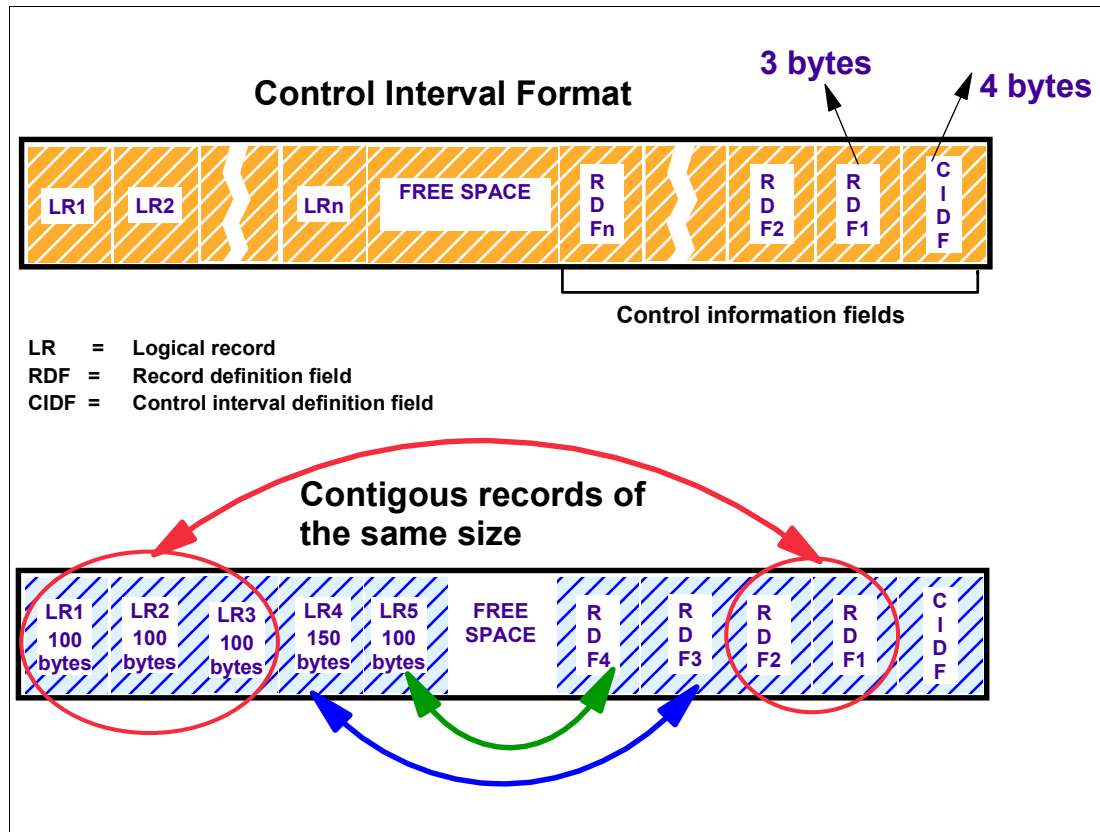


Figure 4-39 Control interval format

### Control interval (CI)

*Control interval* is a VSAM-unique concept. 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 processor during one I/O operation. Whenever a record is retrieved from direct access storage, the entire CI containing the record is read into a VSAM I/O buffer in virtual storage. The desired record is transferred from the VSAM buffer to a user-defined buffer or work area.

Based on the CI size, VSAM calculates the best size of the physical block in order to better use the 3390/3380 logical track. The CI size can be from 512 bytes to 32 KB. A CI consists of:

- ▶ Logical records stored from the beginning to the end of the CI
- ▶ Free space, for data records to be inserted into or lengthened
- ▶ Control information, which is made up of two types of fields:
  - One control interval definition field (CIDF) per CI. CIDF is a 4-byte field. CIDF contains information about the amount and location of free space.
  - Several record definition fields (RDF) describing the logical records. RDF is a 3-byte field and describes the length of records. For fixed length records there are two RDFs, one with the length, and other with how many with the same length.

The size of CIs can vary from one file to another, but all the CIs within the data component of a particular data set must be of the same length. The CI components and properties may

vary, depending on the data set organization. For example, an LDS does not contain CIDs and RDFs in its CI. All of the bytes in the LDS CI are data bytes.

### **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. 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; in other words, 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.

### **Control area (CA)**

Control area is also a VSAM unique concept. 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. A spanned record cannot be larger than a CA.

### **Splits**

CI splits and CA splits occur as a result of data record insertions (or increasing the length of an already existing record) in KSDS and VRRDS organizations. If a 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 end of the data component. This movement creates free CIs in the original CA, then the record to be inserted causes a CI split.

## 4.33 VSAM data set components

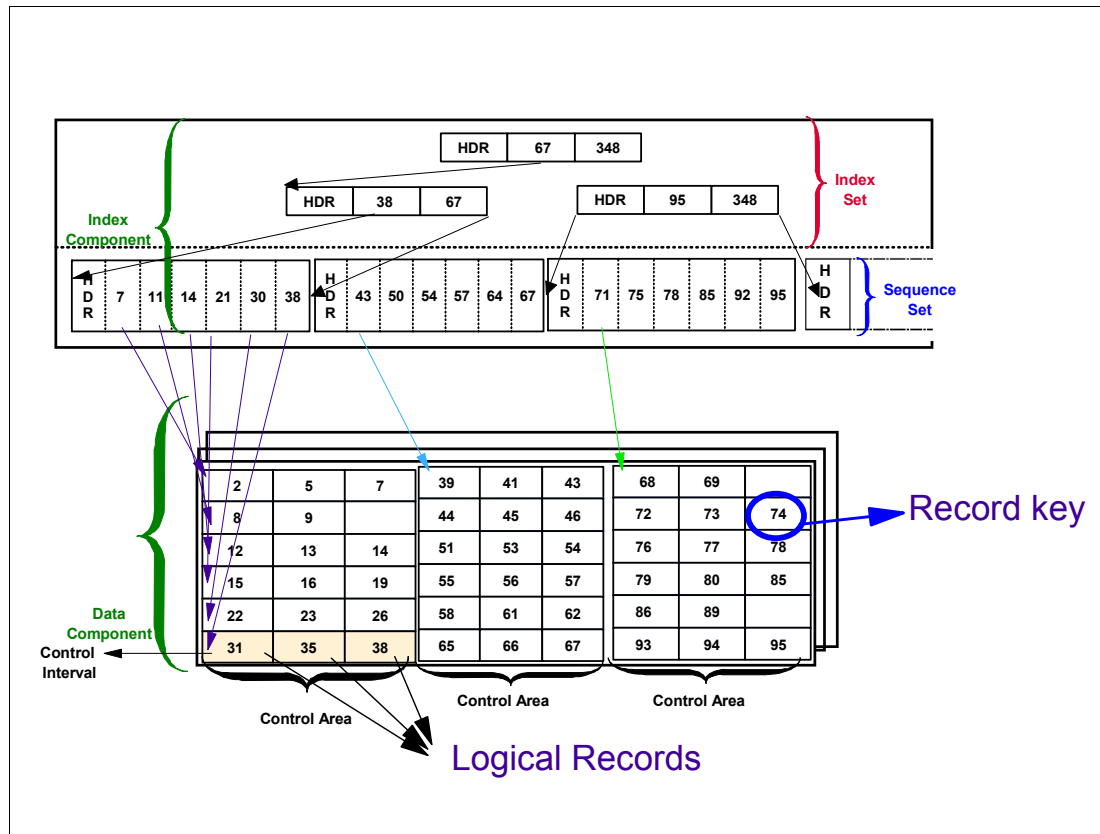


Figure 4-40 VSAM data set components

### VSAM data set components

A *component* is an individual part of a VSAM data set. Each component has a name, an entry in the catalog and an entry in the VTOC. There are two types of components, the data component and the index component. Some VSAM organizations have only the data component.

#### Data component

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

#### Index component

The *index component* is a collection of records containing data keys and pointers (relative byte address, or 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 for a record with a certain key. A VSAM index can consist of more than one level (balanced 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. Let's look at these in more detail now.

## Sequence set

The sequence set is the lowest level of index, and it directly points (through an RBA) to the data CI in the CA. Each index record:

- ▶ Occupies one index CI
- ▶ Maps one CA in the data component
- ▶ Contains pointers and high key information for each data CI
- ▶ Contains horizontal pointers from one sequence set CI to the next higher 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 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.

## Cluster

A *cluster* is the combination of the data component (data set) and the index component (data set) for a KSDS. The cluster provides a way to treat index and data components as a single component with its own name.

## Alternate index (AIX)

Alternate index allows logical records of a KSDS or ESDS to be accessed sequentially and directly by more than one key field. The cluster which the AIX is built on is called 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 consisting of an index component and a data component.

Any field in the base cluster record can be used as an alternate key. It may also overlap the primary key (in a KSDS), or any other alternate key. The same base cluster may have several alternate indexes varying the alternate key. There may 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 may have several employee numbers.

The records in the AIX data component contain the alternate key value 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.

The AIX data set is created with the **DEFINE ALTERNATEINDEX** command, then it is populated via 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. The utility to issue this command is discussed in "Access method services" on page 118.

## Sphere

A *sphere* is a VSAM cluster and its associated data sets. These data sets are alternate indexes of the cluster.

## 4.34 Key sequenced data set (KSDS)

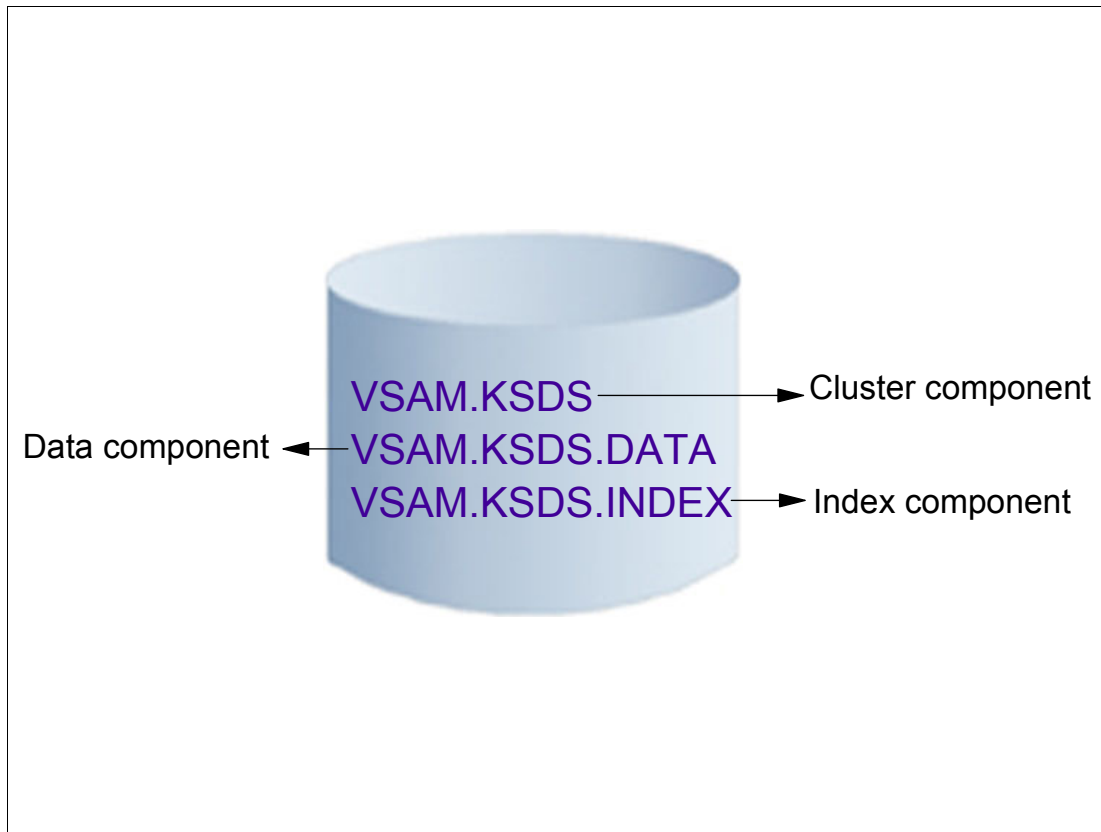


Figure 4-41 Key sequenced data set (KSDS)

### VSAM KSDS data sets

In a 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 data set. The key must be in the same position in each record.

The key data must be contiguous and each key must be unique. After it is specified, the value of the key cannot be altered, but the entire record may be deleted.

When a new record is added to the data set, it is inserted in its collating sequence by key.

A KSDS has a data and an index component. The index component keeps track of the used keys and is used by VSAM to retrieve quickly a record from the data component 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 skip some portions of the data set).



## 4.35 Processing a KSDS data set

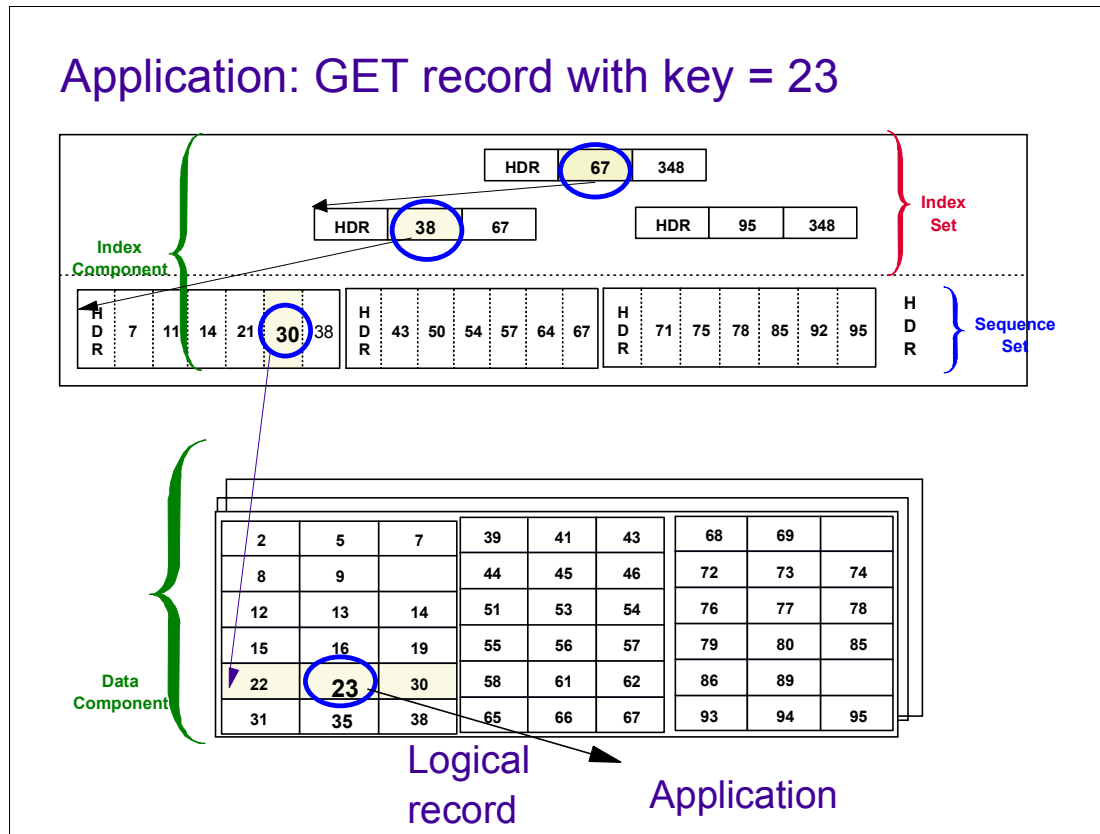


Figure 4-42 Processing an indexed VSAM data set: direct access

### Processing a KSDS data set

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. 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 4-42 shows how VSAM searches the index when an application issues a GET for a logical record with key value 23.

The sequence is as follows:

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 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 highest key 38, searching for a key that is greater or equal to 23.
5. The entry 30 points to the data component CI that holds the desired record.
6. VSAM 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 desired key, the application receives a return code indicating that the record was not found.

## 4.36 Relative record data set (RRDS)

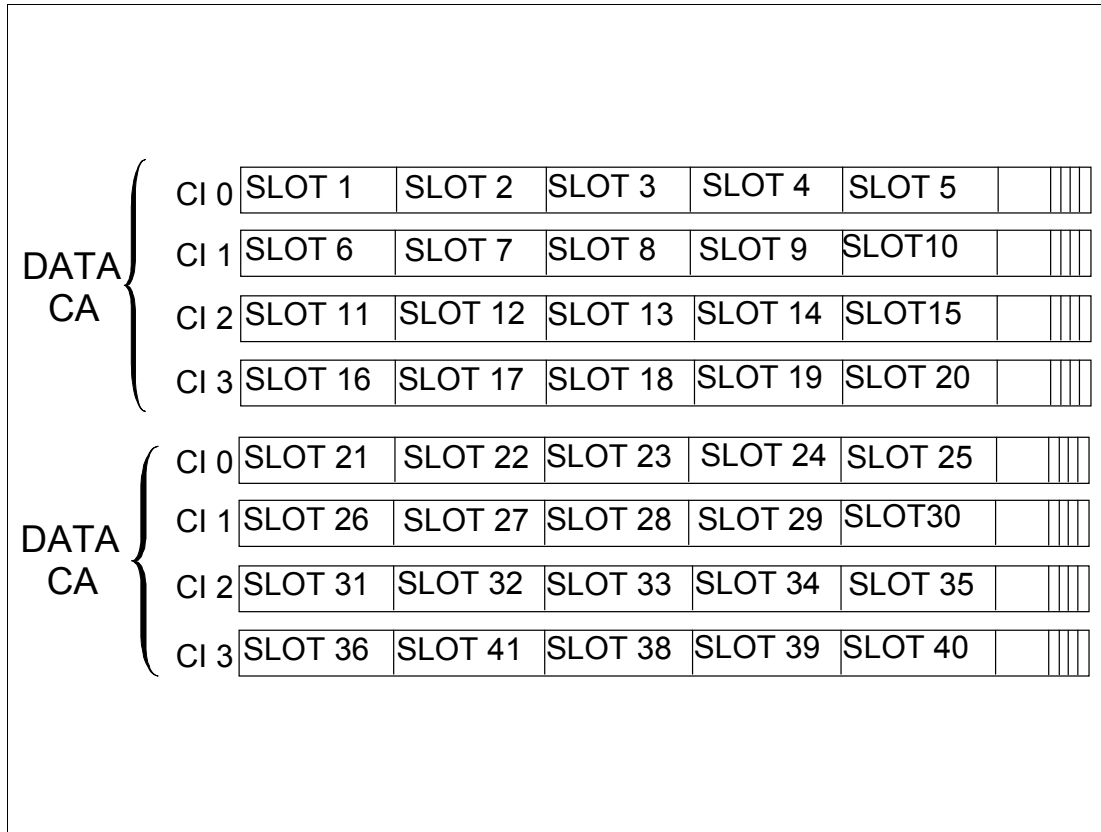


Figure 4-43 Relative record data set

### 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; its relative record number cannot change.

An RRDS has a data component only.

Random load of an RRDS requires a user program.

## 4.37 Typical RRDS processing

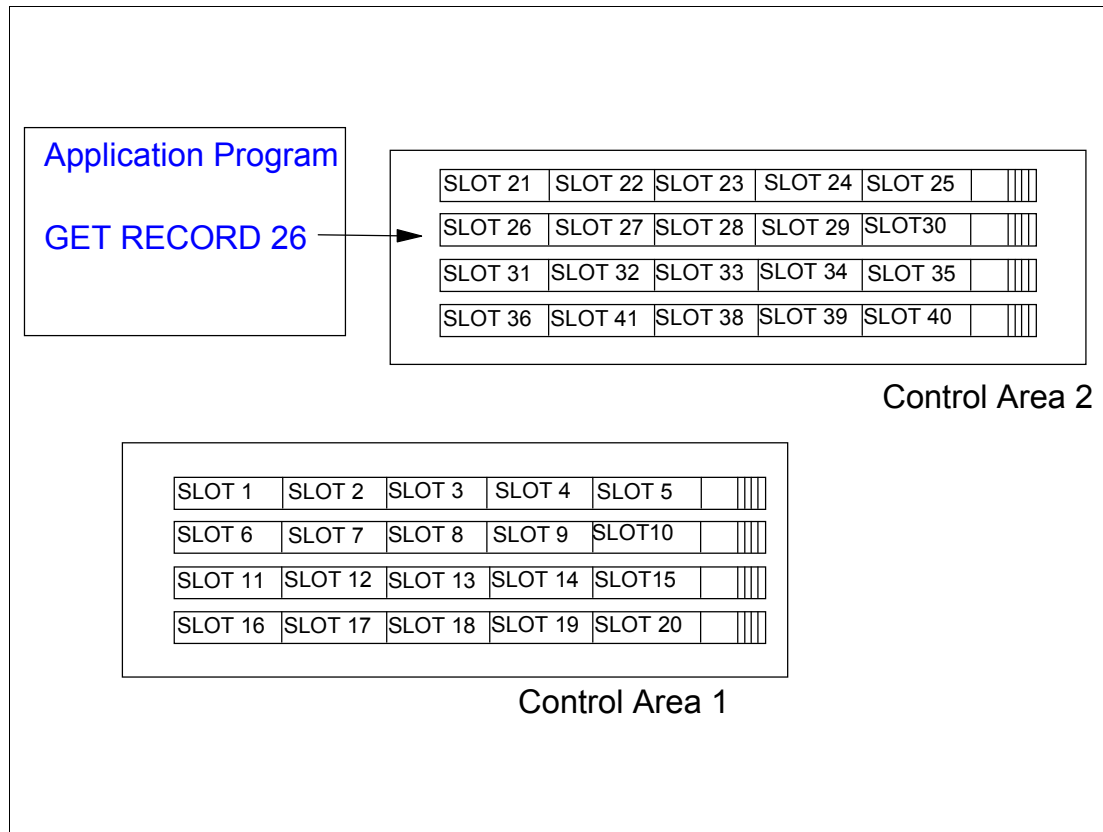


Figure 4-44 Typical RRDS processing

### Processing RRDS data sets

The application program inputs the relative record number of the target record. VSAM is able to find its location very quickly by using a formula that takes into consideration 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.

## 4.38 Linear data set (LDS)

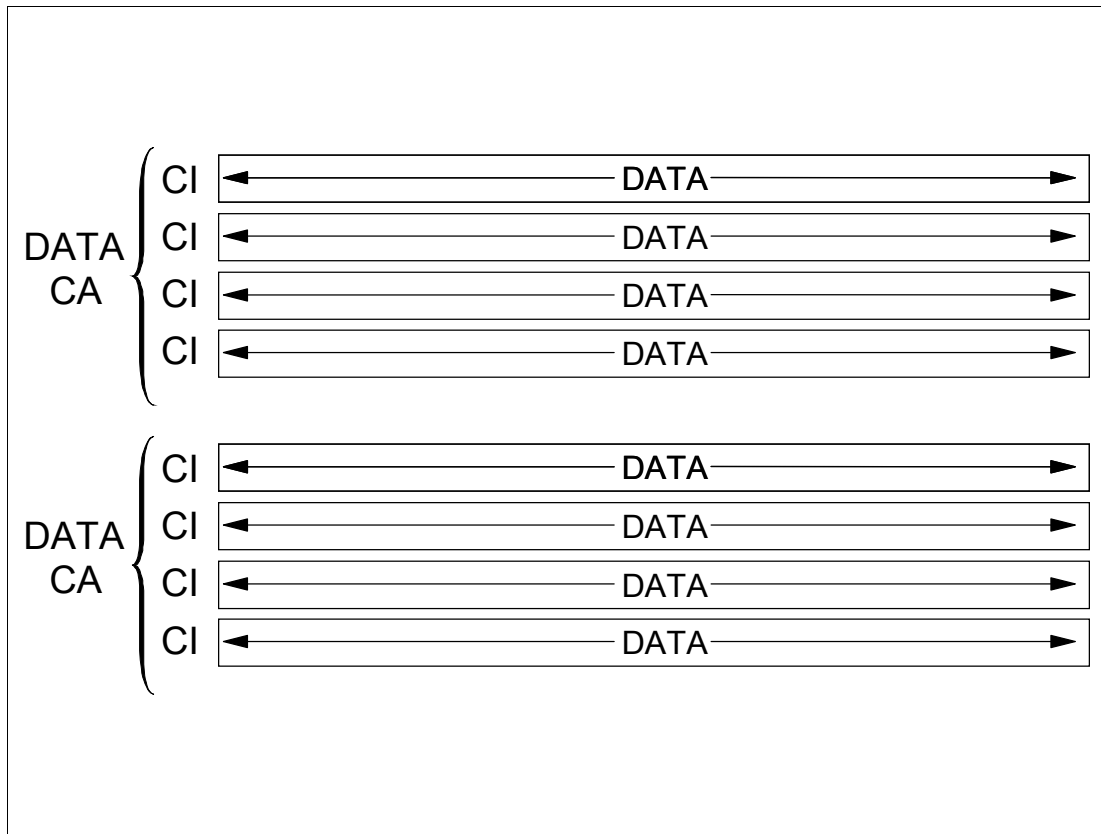


Figure 4-45 Linear data set (LDS)

### Linear data set (LDS)

A *linear data set* is a VSAM data set with a CI size of 4096 bytes. An LDS has no imbedded 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—but 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 comprised of 4 KB blocks, but with a revolutionary buffer technique called data-in-virtual (DIV).

## 4.39 Data-in-virtual

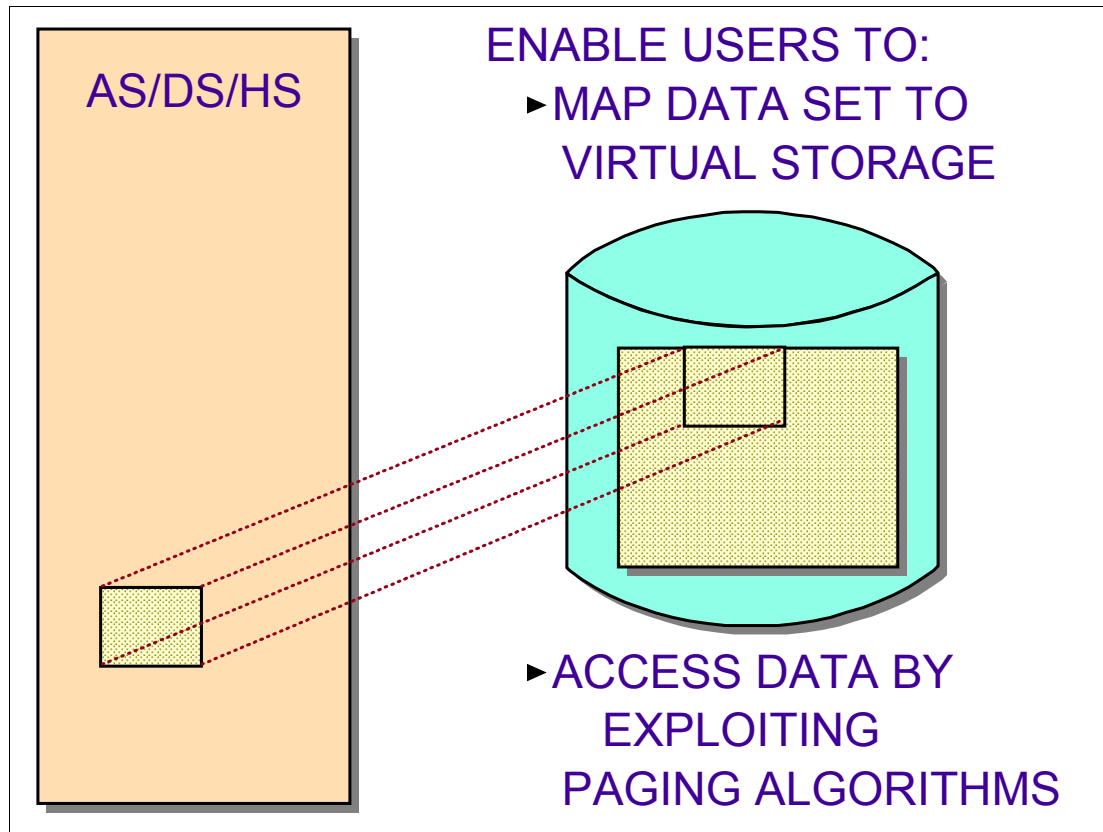


Figure 4-46 Data-in-virtual (DIV)

### Data-in-virtual (DIV)

*Data-in-virtual* (DIV) is an optional and unique buffer technique 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 hiperspace. An LDS cluster is sometimes referred to as a *DIV object*.

Data is read into central storage via the paging algorithms only when that block is actually referenced. During RSM page-steal processing, only changed pages are written to auxiliary storage. Unchanged pages are discarded since 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.

## 4.40 Data-in-virtual objects

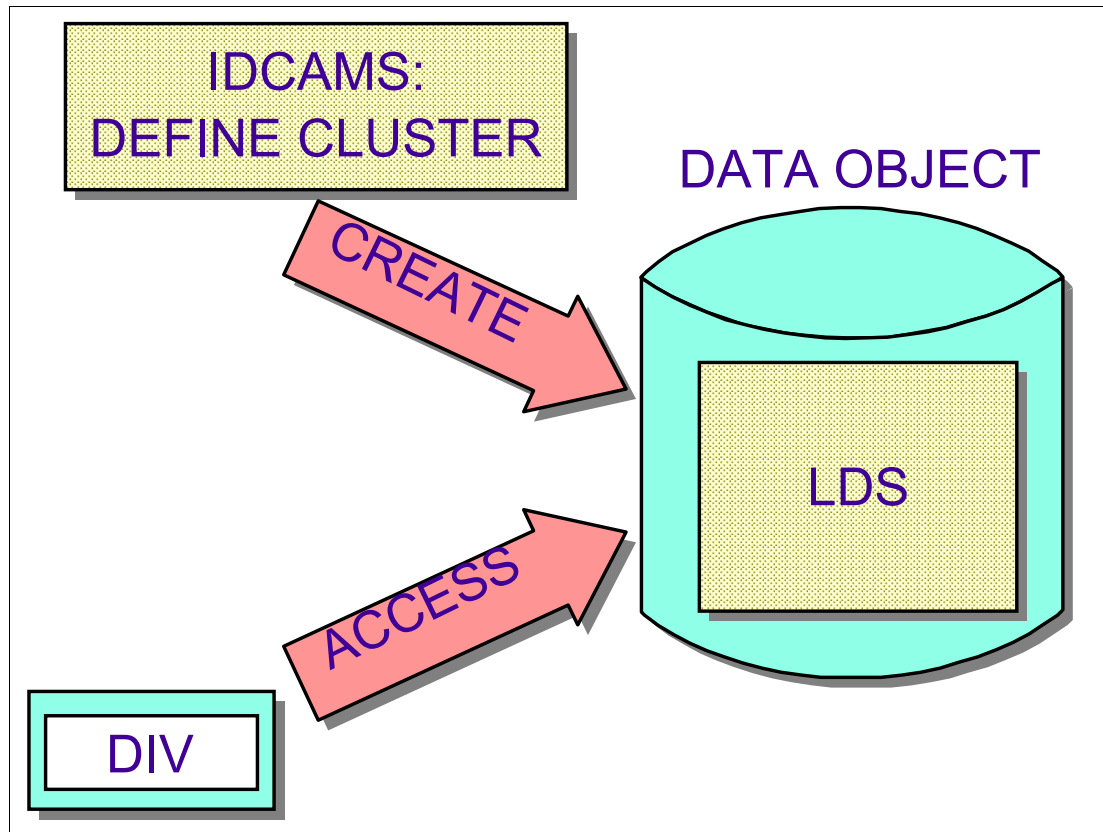


Figure 4-47 Data-in-virtual objects

### Data-in-virtual objects

A linear data set is a VSAM data set with a control interval size of 4096 bytes to 32,768 bytes in increments of 4096 bytes. A linear data set does not have imbedded control information. All linear data set bytes are data bytes. Only integrated catalog facility catalogs can support a linear data set.

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 (CIDFs and RDFs), it cannot be accessed as if 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.

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

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*.

## 4.41 Mapping a linear data set

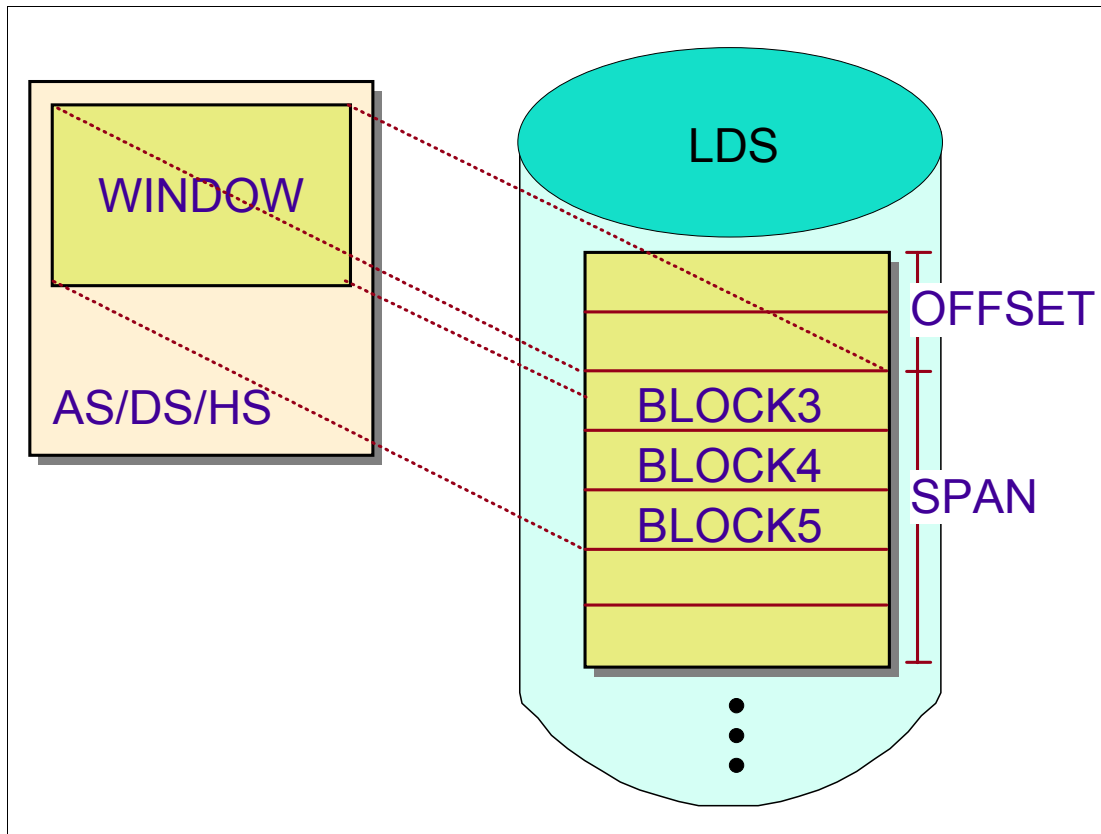


Figure 4-48 Mapping a linear data set

### Mapping a linear data set

To establish a map from a linear data set to a window (a program-provided area in multiples of 4 KB on a 4 KB boundary), the program issues:

- ▶ DIV IDENTIFY to introduce (allocate) a linear data set to data-in-virtual services.
- ▶ DIV ACCESS to cause a VSAM open for the data set and indicate access mode (read/update).
- ▶ DIV MAP to enable the viewing of the data object by establishing an association between a program-provided area and the data object. The area may be in an address space, data space, or hiperspace.

No actual I/O is done until the program references the data in the window. The reference will result in a page fault which causes data-in-virtual services to read the data from the linear data set into the window.

DIV SAVE can be used to write out changes to the data object. DIV RESET can be used to discard changes made in the window since the last SAVE operation.



## 4.42 Entry sequenced data set (ESDS)

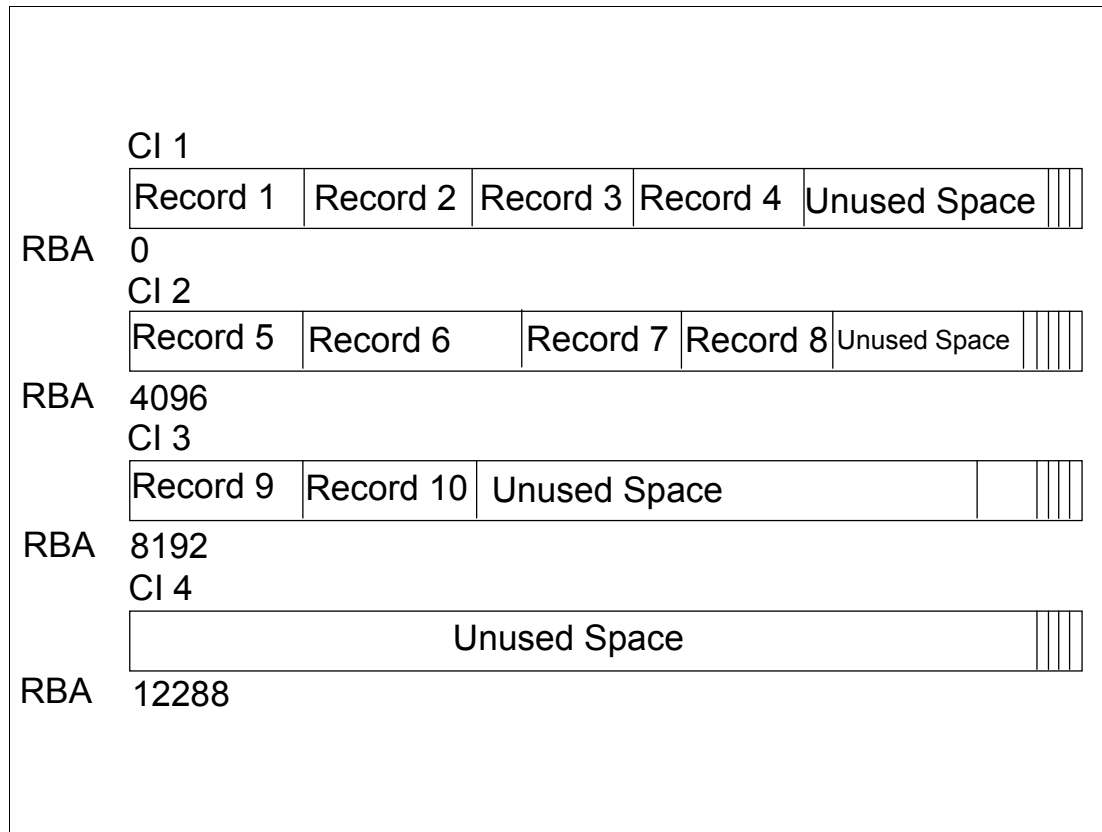


Figure 4-49 Entry sequenced data set (ESDS)

### Entry sequenced data set (ESDS)

An *ESDS* is comparable to a sequential data set. It contains fixed 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 has only a data component.

Records can be accessed sequentially or by relative byte address (RBA). When a record is loaded or added, VSAM indicates its relative byte address (RBA). The RBA is the offset of this 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 to keep track of these RBAs.

## 4.43 Typical ESDS processing

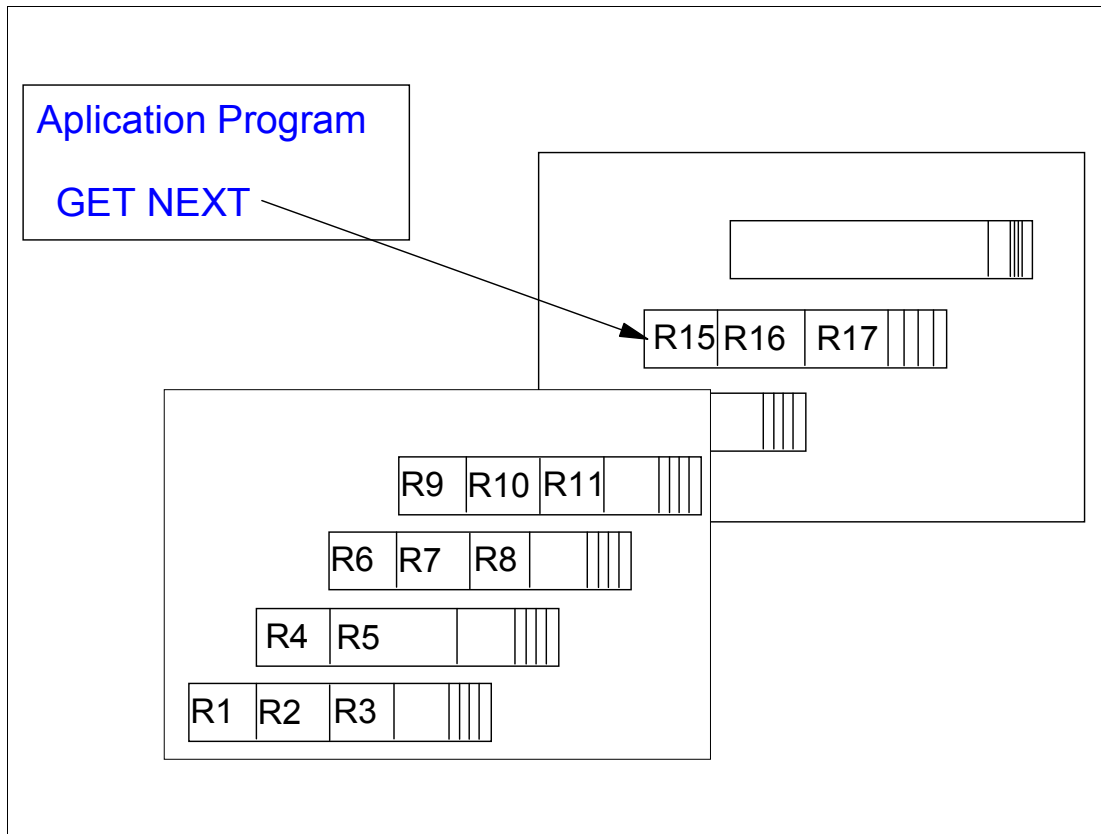


Figure 4-50 Typical ESDS processing (ESDS)

### 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.

## 4.44 DFSORT

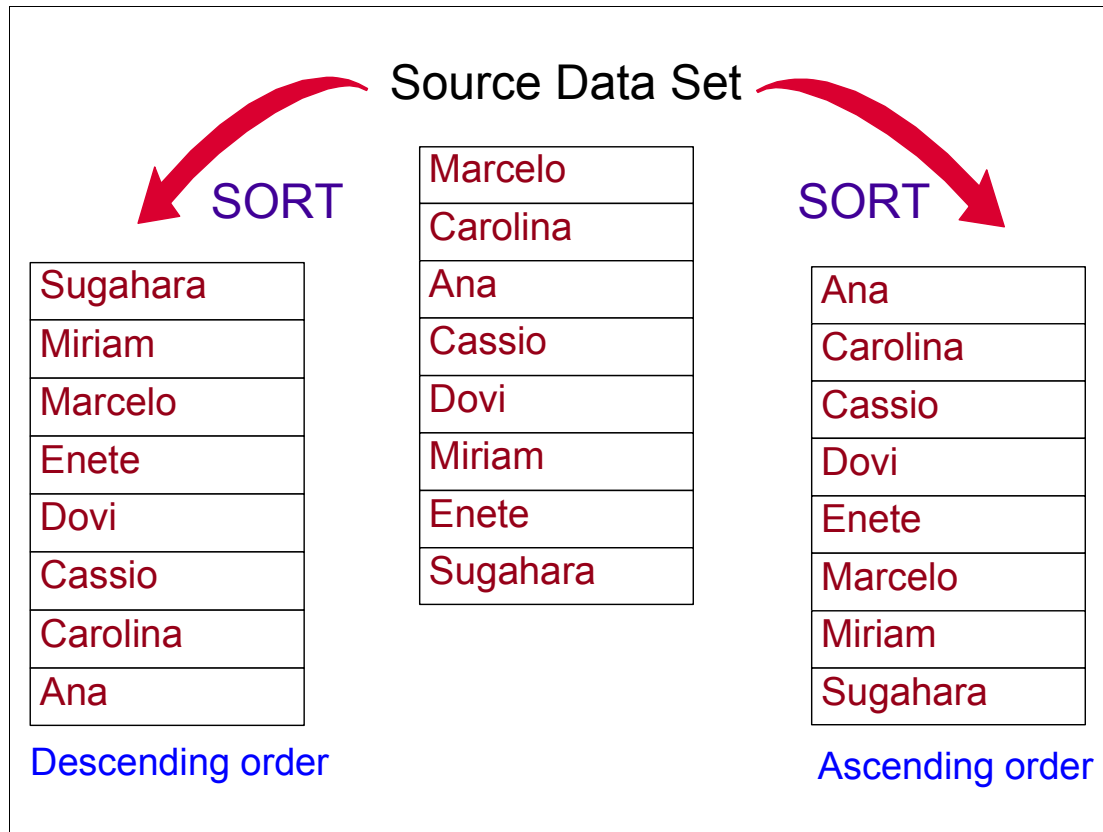


Figure 4-51 DFSORT

### DFSORT

The DFSORT licensed program is a high performance data arranger for z/OS users. With DFSORT, you can sort, merge, and copy data sets using EBCDIC, S/390 decimal or binary keys.

DFSORT merges data sets by combining two or more files of sorted records to form a single data set of sorted records.

You can use DFSORT to do simple 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 DFSORT's record-level editing capability to perform data management tasks.

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

While sorting, merging, or copying data sets, you can also:

- ▶ 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). In 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.

DFSORT 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.

For articles, online books, news, tips, techniques, examples, and more, visit the z/OS DFSORT home page:

<http://www.storage.ibm.com/software/sort/mvs>

For further information about DFSORT, refer to *z/OS DFSORT: Getting Started*, SC26-7527, and other DFSORT books.

## 4.45 DFSMS Network File System

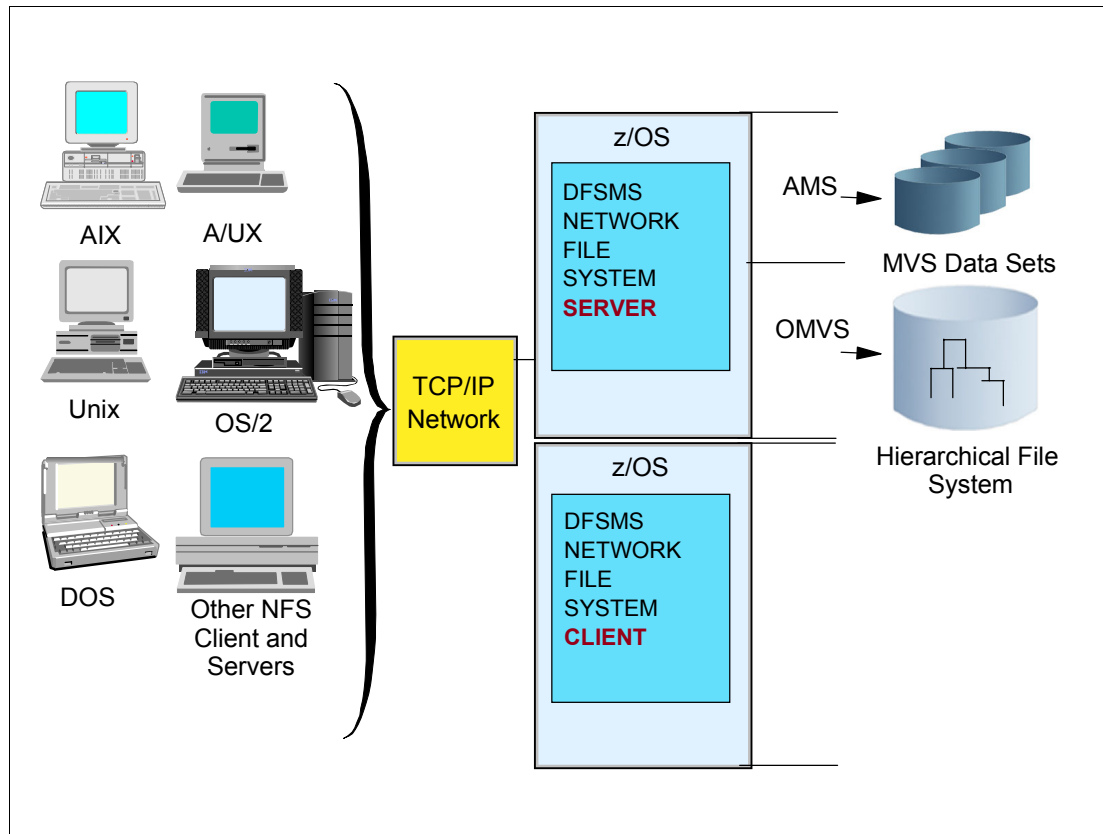


Figure 4-52 DFSMS Network File System

### DFSMS Network File System

A *client* is a computer or process that requests services on the network. A *server* is a computer or process that responds to a request for service from a client. A user accesses a service, which allows the use of data or other resources.

Figure 4-52 illustrates the client-server relationship:

- ▶ The upper center portion shows the DFSMS Network File System (NFS) address space server; the lower portion shows the DFSMS Network File System (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.

With the DFSMS NFS server, you can remotely access z/OS conventional data sets or UNIX server z/OS files from workstations, personal computers, and other systems that run client software for the Sun NFS Version 2 protocols on a TCP/IP network.

The DFSMS NFS server acts as an intermediary to read, write, create, or delete UNIX server z/OS files and z/OS data sets that are maintained on an z/OS host system. The remote z/OS data sets or UNIX server z/OS files are mounted from the host processor to appear as local directories and files on the client system.

This server makes the strengths of an z/OS host processor—storage management, high-performance disk storage, security, and centralized data—available to the client platforms.

With the DFSMS NFS client, you can allow basic sequential access method (BSAM), queued sequential access method (QSAM), virtual storage access method (VSAM), and UNIX server z/OS users and applications to have transparent access to data on systems that support Sun NFS Version 2 protocols.

The remote NFS server can be a z/OS, UNIX, AIX, OS/2, or other system. The DFSMS NFS client is implemented on UNIX server z/OS and implements the client portion of the Sun NFS Version 2 protocols.

The Network File System can be used for:

- ▶ File sharing between platforms
- ▶ File serving (as a data repository)

For further information about NFS, refer to *z/OS Network File System Customization and Operation*, SC26-7417, and *z/OS Network File System User's Guide*, SC26-7419.

## 4.46 DFSMS Optimizer

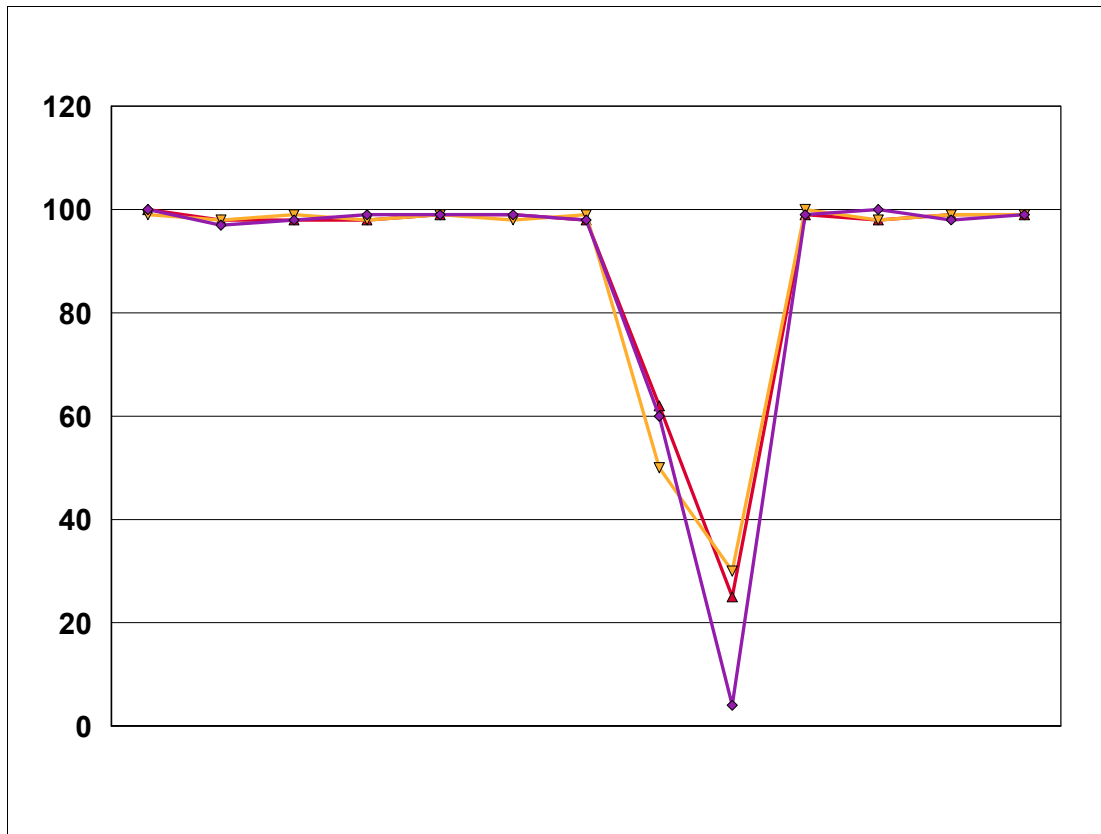


Figure 4-53 DFSMS Optimizer

### DFSMS Optimizer

The DFSMS Optimizer provides analysis and simulation information for both SMS and non-SMS users. The DFSMS Optimizer can help you maximize storage use and minimize storage costs. It provides methods and facilities for you to:

- ▶ Monitor and tune DFSMShsm functions as migration and backup
- ▶ Create and maintain a historical database of system and data activity
- ▶ Fine-tune an SMS configuration by performing in-depth analysis of:
  - Management class policies, including simulations and cost-benefit-analysis using your storage component costs
  - Storage class policies for SMS data, with recommendations for both SMS and non-SMS data
  - High I/O activity data sets, including recommendations for placement and simulation for cache and expanded storage
  - Storage hardware performance of subsystems and volumes including I/O rate, response time, and caching statistics
- ▶ Simulate potential policy changes and understand the costs of those changes
- ▶ Produce presentation-quality charts

For more information about the DFSMS Optimizer, refer to *DFSMS Optimizer User's Guide and Reference*, SC26-7047.

## 4.47 DFSMSDss

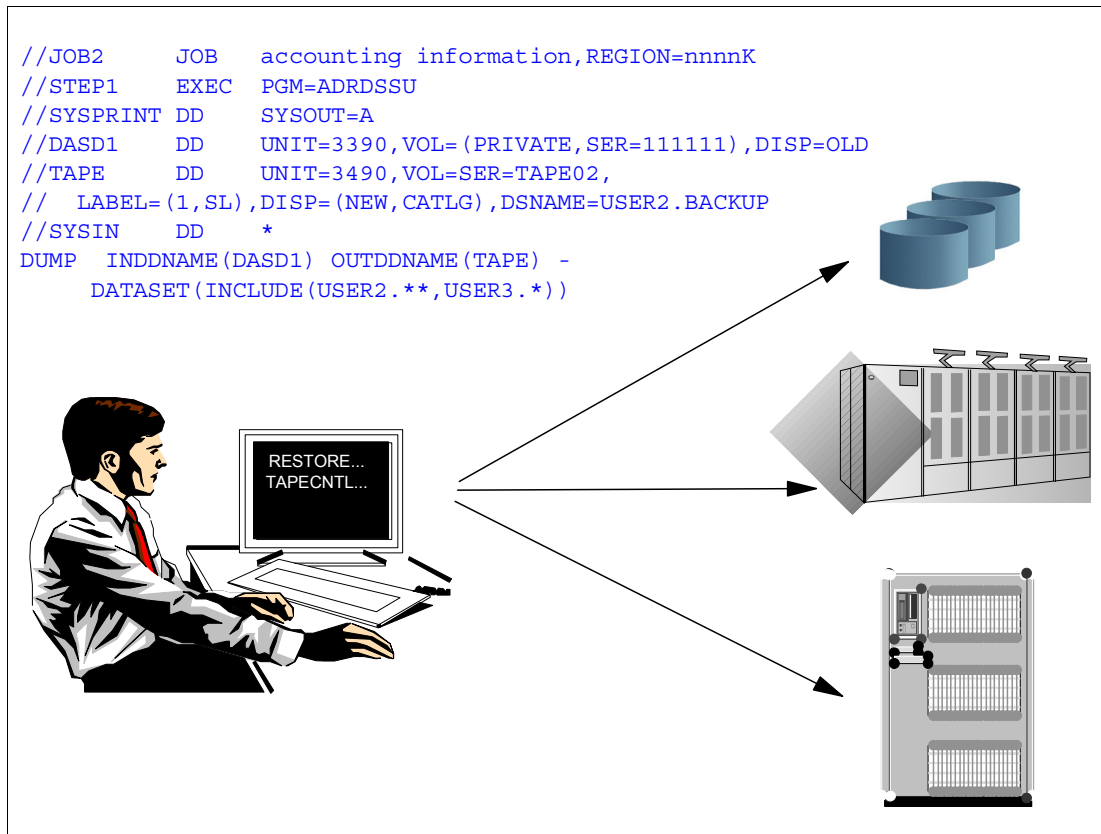


Figure 4-54 DFSMSDss

### DFSMSDss

DFSMSDss is a direct access storage device (DASD) data and space management tool. DFSMSDss works on DASD volumes only in the MVS environment. You can use DFSMSDss to do the following:

- ▶ Copy and move data sets between volumes of like and unlike device types

**Note:** Like devices have the same track capacity and number of tracks per cylinder (for example, 3380 Model D, Model E, and Model K). Unlike DASD devices have different track capacities (for example, 3380 and 3390), a different number of tracks per cylinder, or both.

- ▶ 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 in 9390/3990 control units. If the control unit is a 9393 RVA, a snapshot is transparently generated without any change in the JCL.



## 4.48 DFSMSDss: physical and logical processing

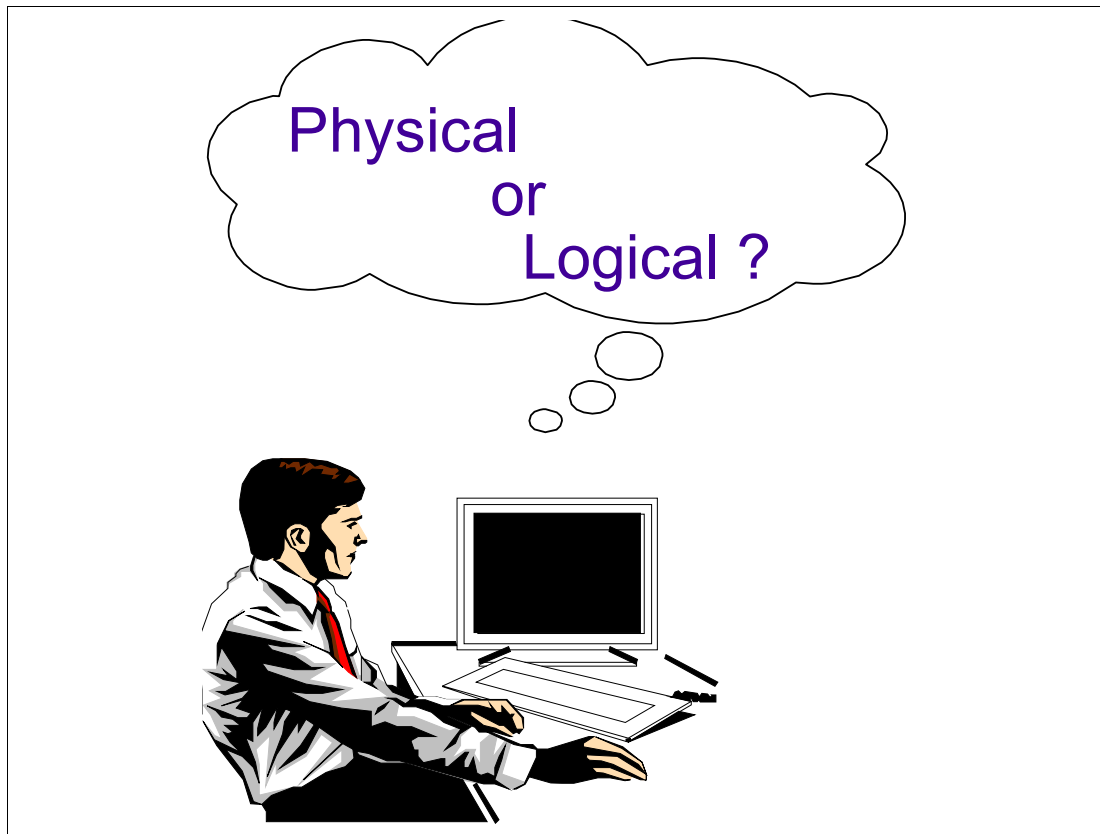


Figure 4-55 DFSMSDss physical and logical processing

### DFSMSDss: physical and logical processing

Before you begin using DFSMSDss, you should understand the difference between logical processing and physical processing. DFSMSDss can perform two kinds of processing when executing **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 it is dumped because physical and logical dump tapes have different formats. If a data set is dumped logically, it is restored logically; 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.

## 4.49 DFSMSDss: logical processing

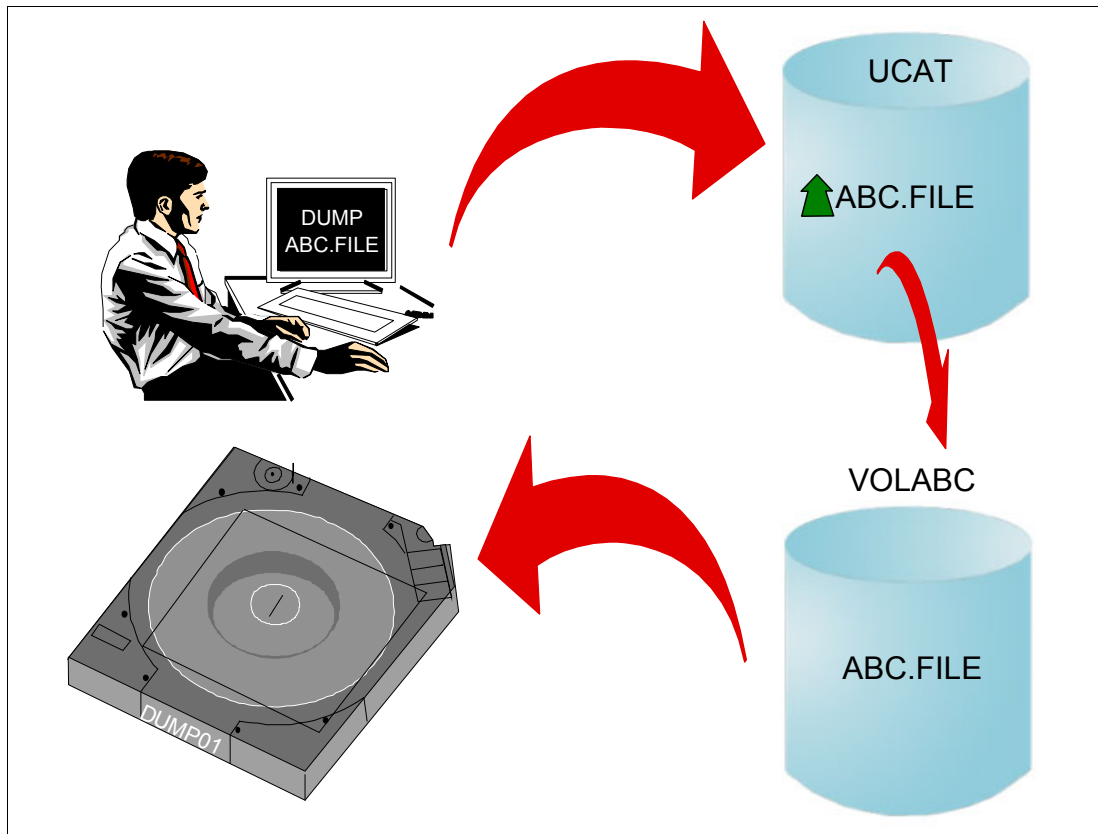


Figure 4-56 DFSMSDss logical processing

### 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 dump operation.

DFSMSDss performs logical processing if:

- ▶ You specify the data set 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 data set keyword with the **DUMP** command, and either no input volume is specified, or **LOGINDDNAME** or **LOGINDYNAM** 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.

### 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 may need to be restored to an unlike device is dumped.
- ▶ Data must be restored the same way it is dumped. This is particularly important to bear in mind when making backups that you plan to retain for a long period of time (such as vital records backups). If a backup is retained for a long period of time, it is possible that the device type it originally resided on will no longer be in use at your site when you want to restore it. This means you will have to 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.

## 4.50 DFSMSDss: physical processing

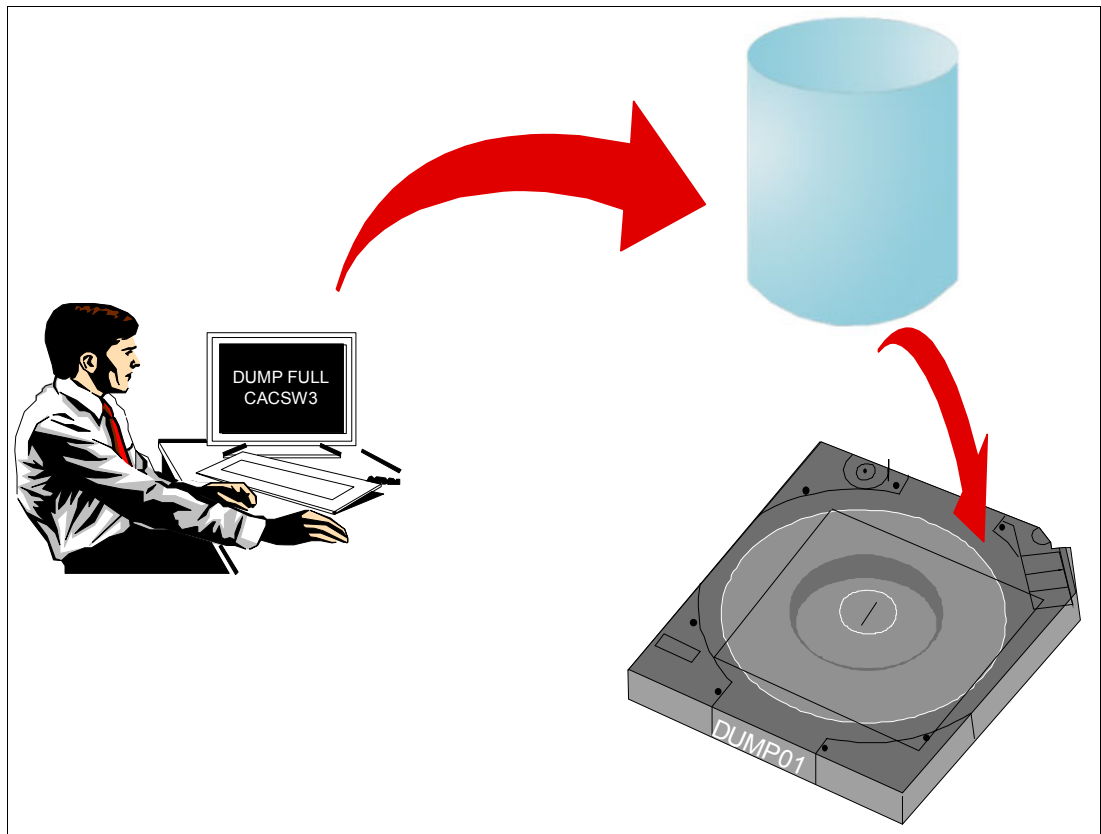


Figure 4-57 DFSMSDss physical processing

### 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 if:

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

**Attention:** Be aware that, when invoking 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 could 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 produces a physical data set dump.
- ▶ The **RESTORE** command is executed and the input volume is created by a physical dump operation.

## When to use physical processing

Use physical processing when:

- ▶ 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 to dump an entire volume is with a physical full-volume command. This 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.

## 4.51 DFSMSDss stand-alone services

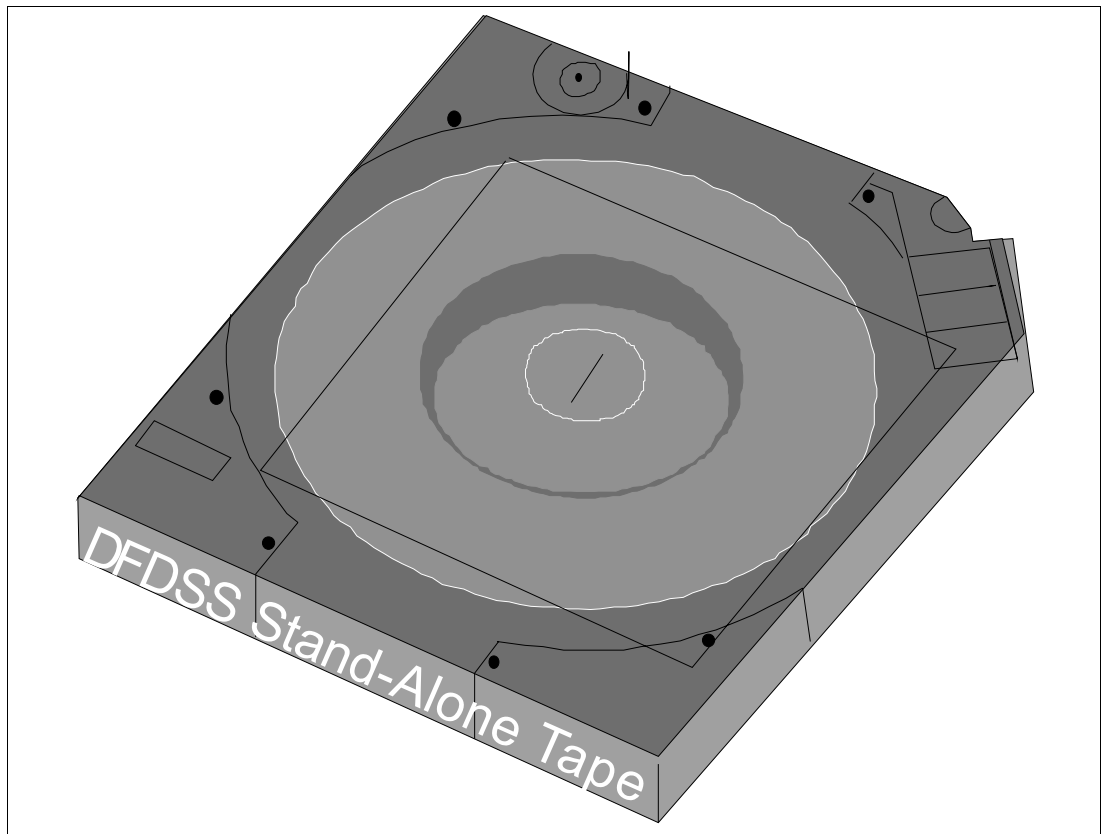


Figure 4-58 DFSMSDss stand-alone services

### Stand-alone services

DFSMS Version 1 Release 4 provided a new stand-alone services function, that is intended for the storage administrator, the system programmer, and anyone who runs the stand-alone services program. This, along with related information in *z/OS MVS System Messages, Volume 1 (ABA-AOM)*, SA22-7631, supports a new stand-alone services program.

Stand-alone services can perform either a full-volume restore or a tracks restore from dump tapes produced by DFSMSDss or DFDSS. It offers the following benefits when compared to the previous DFSMSDss stand-alone functions:

- ▶ Provides user-friendly commands to replace the previous control statements
- ▶ Supports IBM 3494 and 3495 Tape Libraries, and 3590 Tape Subsystems
- ▶ Supports IPLing from a DASD volume, in addition to tape and card readers
- ▶ Allows you to predefine the operator console to be used during stand-alone services processing

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

## 4.52 DFSMSHsm

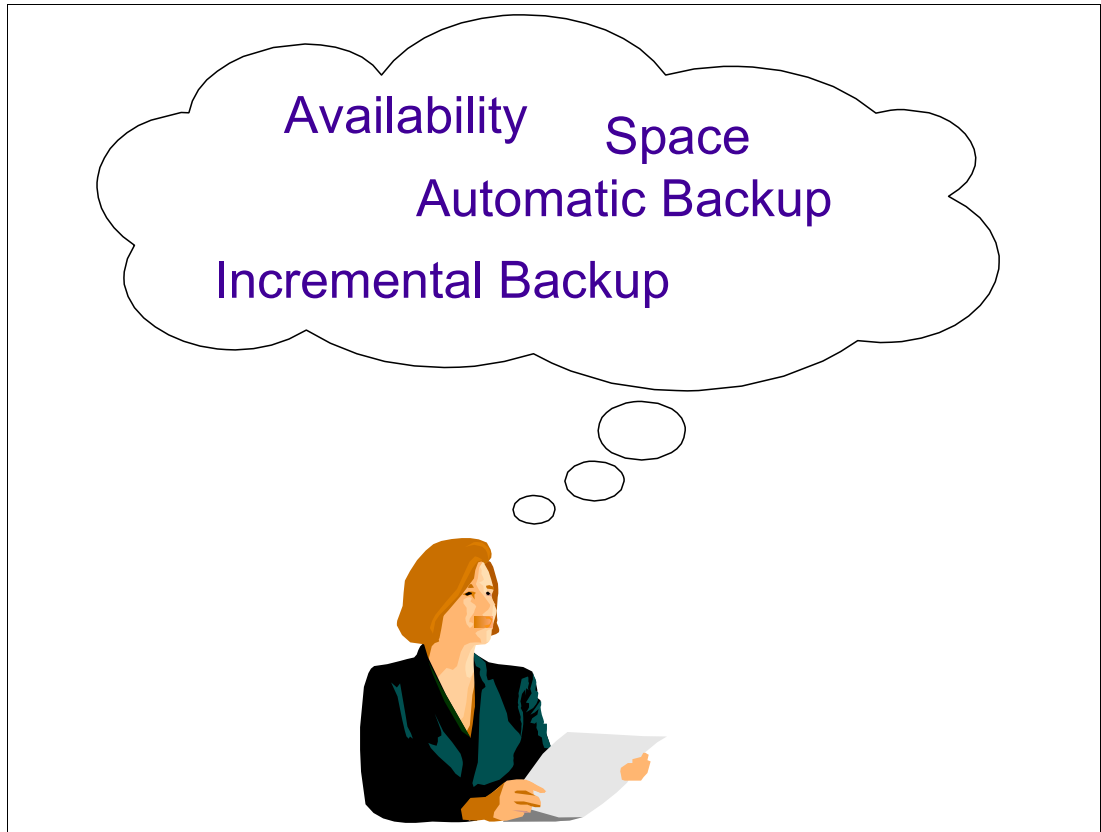


Figure 4-59 DFSMSHsm

### DFSMSHsm

DFSMSHsm is a licensed program that automatically performs *space management* and *availability management* in a storage device 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, refer to *z/OS DFSMSHsm Storage Administration Guide*, SC35-0421, and *z/OS DFSMSHsm Storage Administration Reference*, SC35-0422.

## 4.53 Availability management

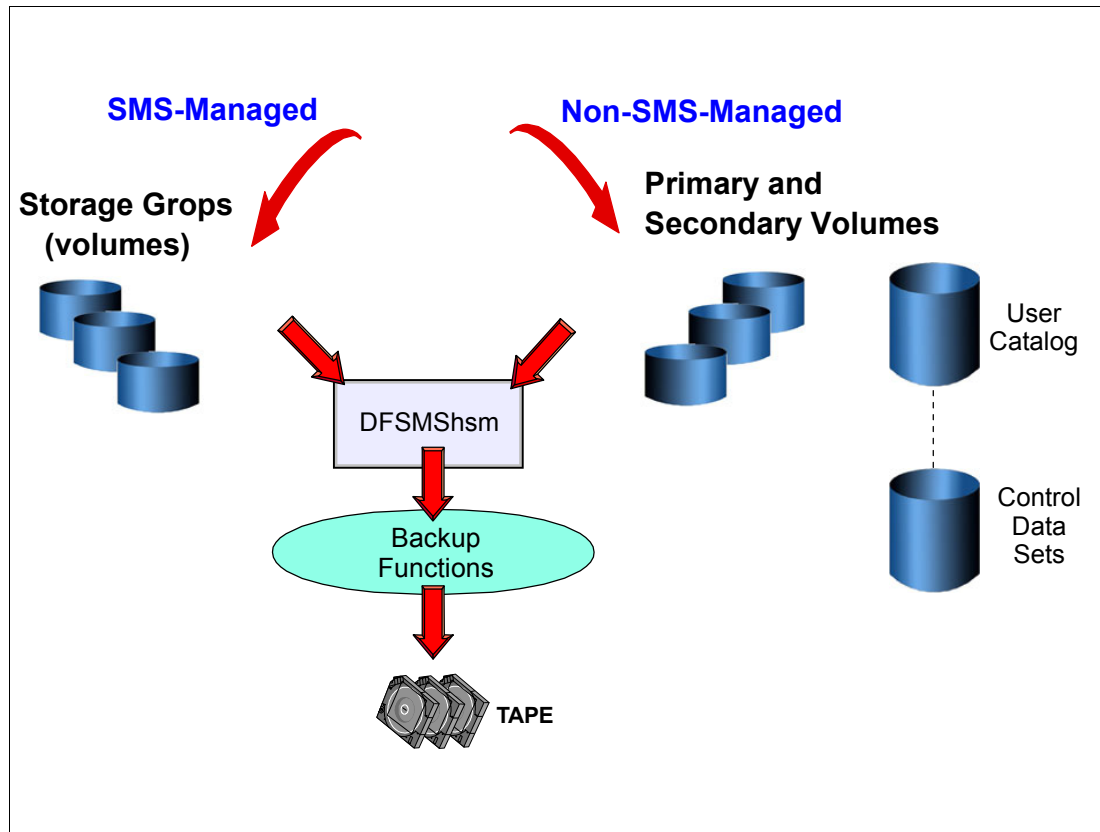


Figure 4-60 DFSMSHsm availability management

### Availability management

*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. To do this, availability management automatically and periodically performs functions that:

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.

Availability management functions are:

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



## 4.54 Space management

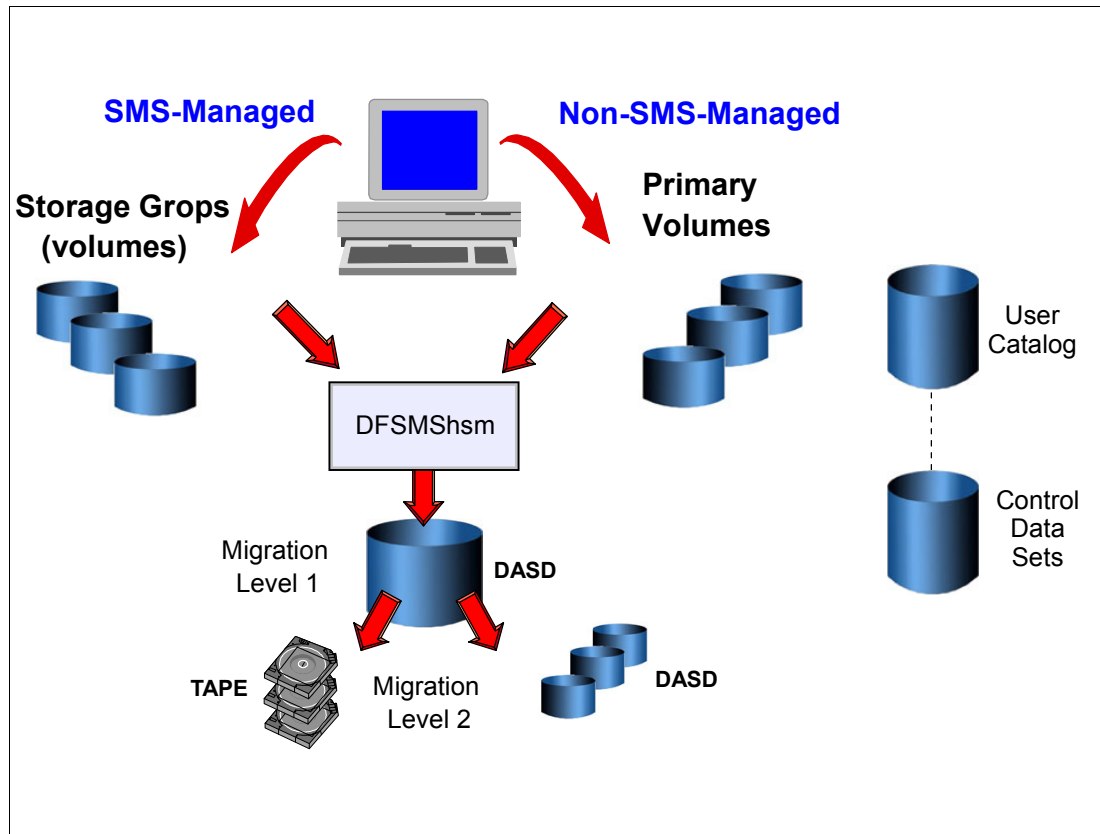


Figure 4-61 DFSMSHsm space management

### Space management

*Space management* is the function of DFSMSHsm that allows you to keep DASD space available for users in order 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 functions that:

1. Move low-activity data sets from user-accessible volumes to DFSMSHsm volumes
2. Reduce the space occupied by data on both the user-accessible volumes and the DFSMSHsm volumes

The DFSMSHsm space management functions are:

- ▶ Automatic primary space management of DFSMSHsm-managed volumes, which includes:
  - Deletion of temporary data sets
  - Deletion of expired data sets
  - Release of unused, over-allocated space
  - Migration to DFSMSHsm-owned migration level 1 (ML1) volumes (compressed)
- ▶ Automatic secondary space management of DFSMSHsm-owned volumes, which includes:
  - ML1 cleanup, including deletion of expired migrated data sets and some migration control data set (MCDS) records

- Moving migration copies from migration level 1 (ML1) to migration level 2 (ML2) volumes
- ▶ Automatic interval migration, initiated when a DFSMSHsm-managed volume exceeds a specified threshold
- ▶ Automatic recall of user data sets back to DASD volumes, when referenced by the application
- ▶ Space management by command
- ▶ Space-saving functions, which include:
  - Data compaction and data compression. Compaction provides space savings through fewer gaps and less control data. Compression provides a more compact way to store data.
  - Partitioned data set (PDS) free space compression.
  - Small data set packing (SDSP) data set facility, which allows small data sets be packaged in just one physical track.
  - Data set rebuilding.

It is possible to have more than one z/OS image sharing the same 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:

- ▶ Hourly space checks
- ▶ During auto backup: CDS BUP, BUP of ML1 data sets to tape
- ▶ During auto dump: expiration of dump copies and deletion of excess dump VTOC copy data sets
- ▶ During 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. Secondary host promotion uses XCF status monitoring to execute the promotion. To indicate a system as a candidate, issue:

- ▶ SETSYS PRIMARYHOST(YES), and
- ▶ SSM(YES)

## 4.55 Storage device hierarchy

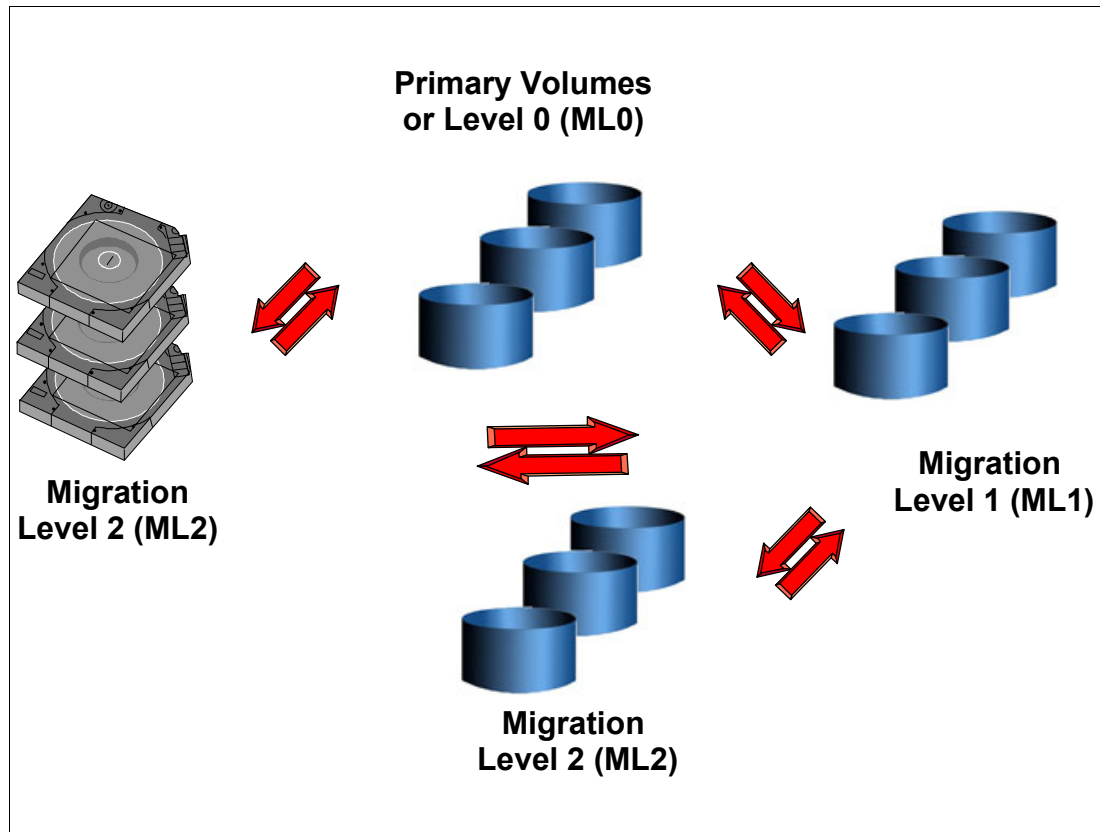


Figure 4-62 Storage device hierarchy

### 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.

DFSMSHsm uses the following three-level storage device hierarchy for space management:

- ▶ Level 0: Are DFSMSHsm-managed storage devices at the highest level of the hierarchy; these devices contain data directly accessible to your application.
- ▶ Level 1 and Level 2: Storage devices at the lower levels of the hierarchy, level 1 and level 2, contain data that DFSMSHsm has compressed and optionally compacted into a format that you cannot use. Devices at this level provide lower cost per byte storage and usually slower response time. Usually L1 is in a cheaper DASD (or the same cost, but with the gain of compression) and L2 is on tape.

**Note:** If you have RVA DASD, you may skip level 1 (ML1) migration because the data in L0 is already compacted/compressed.

## 4.56 HSM volume types

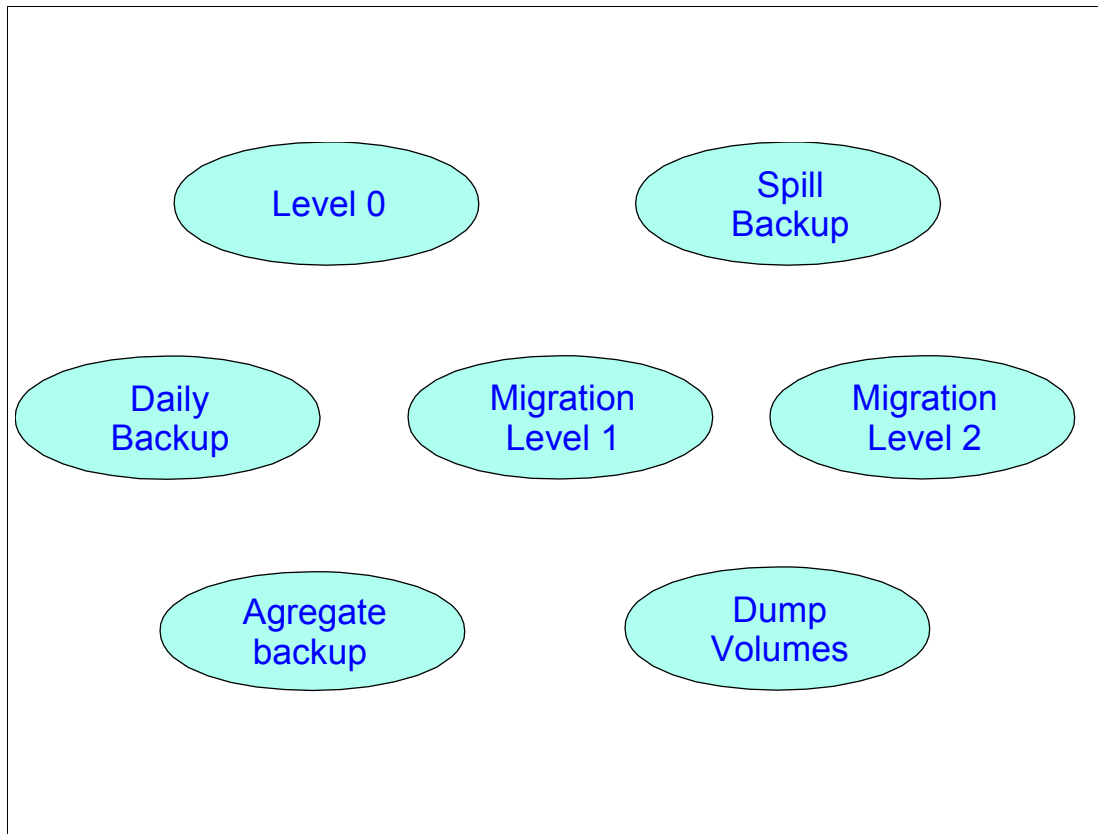


Figure 4-63 DFSMSHsm volume types

### DFSMSHsm volume types

Backing up an individual cataloged data set is performed in the same way as for SMS-managed data sets. However, to back up individual uncataloged data sets, issue the following commands:

```
BACKDS dsname UNIT(unittype) VOLUME(volser)
```

```
HBACKDS dsname UNIT(unittype) VOLUME(volser)
```

The **HBACKDS** form of the command can be used by either non-DFSMSHsm-authorized or DFSMSHsm-authorized users. The **BACKDS** form of the command can be used only by DFSMSHsm-authorized users. The **UNIT** and **VOLUME** parameters are required because DFSMSHsm cannot locate an uncataloged data set without being told where it is.

DFSMSHsm supports the following volume types:

- ▶ Level 0 (L0) volumes contain data sets that are directly accessible to you and the jobs 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:
  - 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) 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 may 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 invoked 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 recovery process (ABARS).

## 4.57 Automatic space management

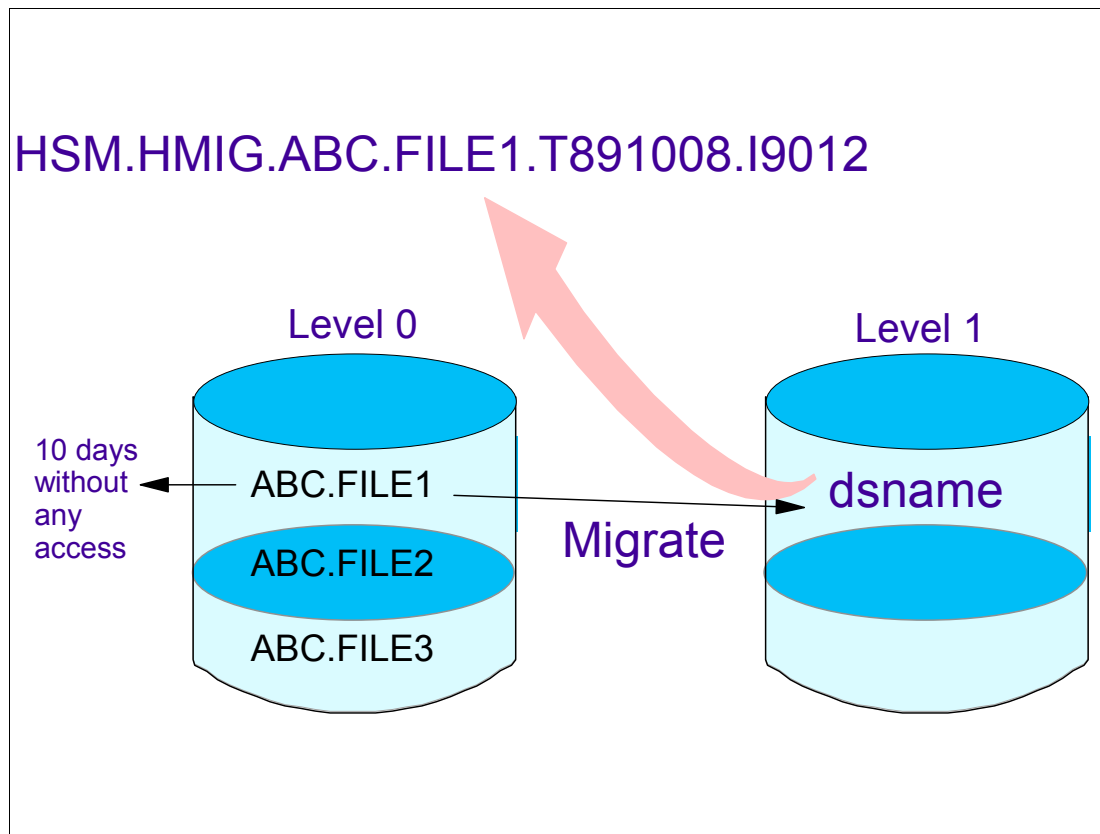


Figure 4-64 DFSMSHsm automatic space management

### Automatic space management

*Automatic space management* prepares the computing system for the addition of new data by freeing space on the DFSMSHsm-managed volumes (L0) and DFSMSHsm-owned volumes (ML1). The functions associated with automatic space management can be divided into two groups

#### **Automatic volume space management**

##### **Primary**

Invoked timely in a daily basis, it cleans L0 volumes by deleting expired and temporary data sets and releasing allocated and not used space. If after that the free space is still below a threshold, then it moves data sets (under control of the management class) from L0 to ML1 or ML2 volumes.

##### **Interval migration**

Executed each hour throughout the day, as needed for all storage groups. In interval migration, DFSMSHsm performs a space check on each DFSMSHsm volume being managed. A volume is considered eligible for interval migration based on the AUTOMIGRATE and THRESHOLD settings.

#### **Automatic secondary space management**

It 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.

## 4.58 Recall

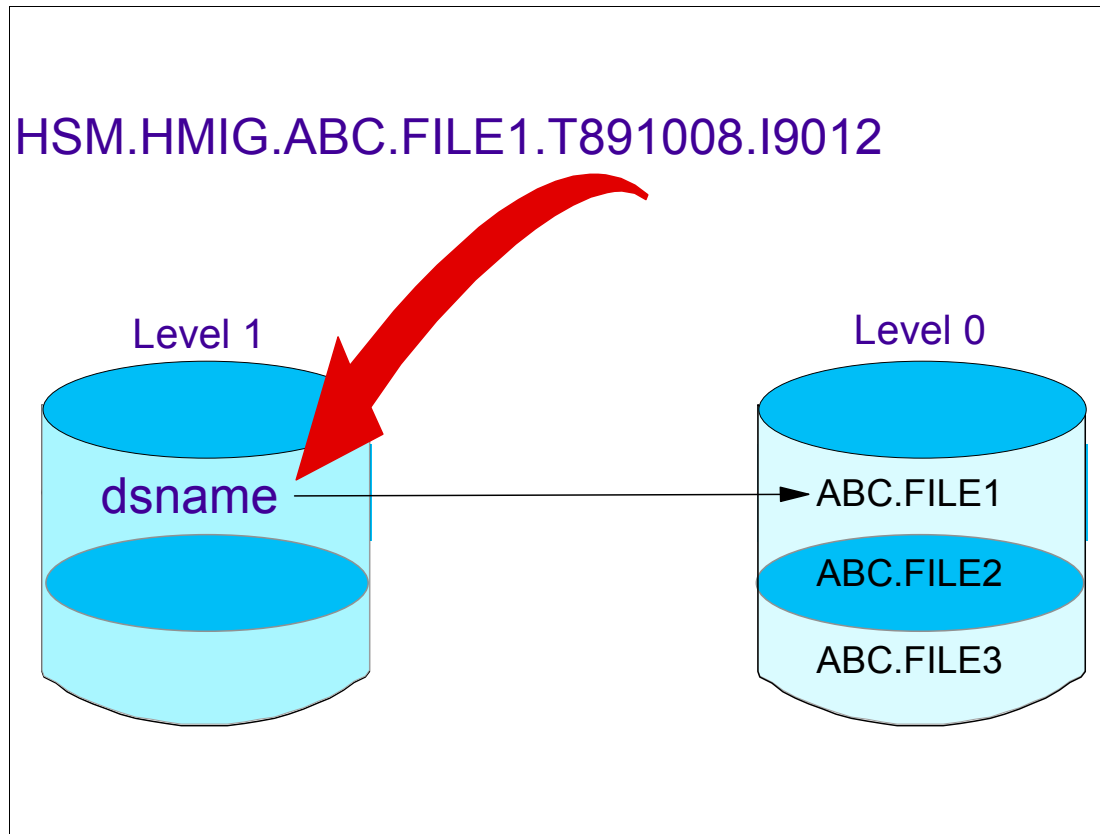


Figure 4-65 Recall

### Automatic and command recall

Recall returns a migrated data set to a user L0 volume. The recall is transparent and the application does not need to know that it happened or where the migrated data set resides. To provide applications with quick access to their migrated data sets, DFSMSHsm allows up to 15 concurrent recall tasks. RMF monitor III shows delays caused by the recall operation.

The MVS allocation routine discovers that the data set is migrated when, while accessing the catalog, it finds the word MIGRAT instead of the volser.

*Automatic recall* returns your migrated data set to a DFSMSHsm-managed volume when you refer to it. The catalog is updated accordingly.

*Command recall* returns your migrated data set to a user volume when you enter the **HRECALL** DFSMSHsm command through an ISMF panel or by directly keying in the command.

For both automatic and command recall, DFSMSHsm working with SMS invokes the automatic class selection (ACS) routines. Data sets that were not SMS-managed at the time they were migrated may be recalled as SMS-managed data sets. The ACS routines determine whether the data sets should be recalled as SMS-managed, and if so, the routines select the classes and storage groups in which the data sets will reside. The system chooses the appropriate volume for the data sets.

DFSMSHsm working without SMS returns a migrated data set to a DFSMSHsm-managed non-SMS level 0 volume with the most free space.

## 4.59 Removable media manager (DFSMSrmm)

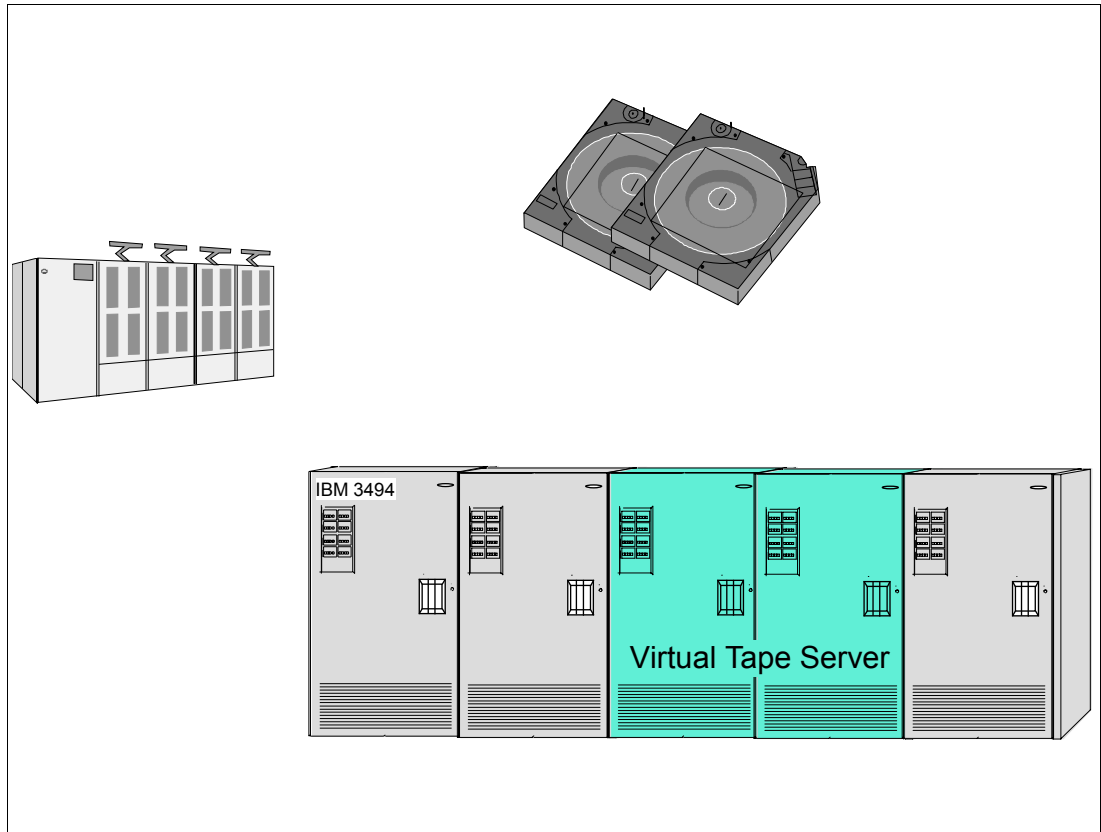


Figure 4-66 DFSMSrmm

### 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 of global control information, these systems must have accessibility to some 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 (such as cartridge system tapes and 3420 reels), as well as other removable media you define to it. For example, DFSMSrmm can record the shelf location for optical disks and track their vital record status; however, it does not manage the objects on optical disks.



## 4.60 Libraries and locations

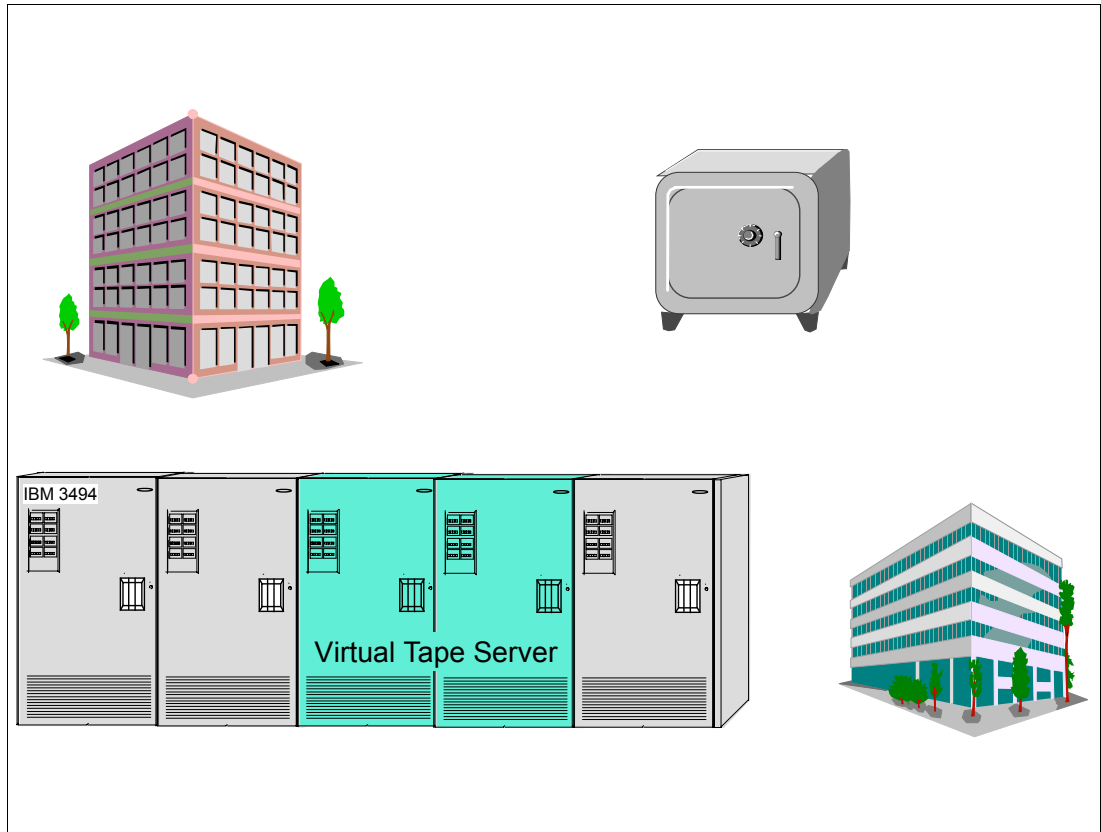


Figure 4-67 Libraries and locations

### 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 your customers and vendors.

## 4.61 What DFSMSrmm can manage

- ❑ Removable media library
  - System-managed tape libraries
    - Automated tape libraries
    - Manual tape libraries
  - Non-system-managed tape libraries or traditional tape libraries
- ❑ Storage locations
  - Installation defined
  - DFSMSrmm built-in
    - Local
    - Distant
    - Remote

Figure 4-68 What DFSMSrmm can manage

DFSMSrmm can manage the following libraries and storage locations:

### Removable media library

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

- ▶ System-managed libraries, such as automated or manual tape 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 sub commands 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 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 dataser* is a device consisting of robotic components, cartridge storage areas (or shelves), tape subsystems, and controlling hardware and software, together with the set of tape volumes that reside in the library and can be mounted on the library tape drives. The IBM automated tape libraries are the automated IBM 3494 and IBM 3495 Library Dataservers.
- ▶ A *manual tape library dataser* is a set of tape drives and the set of system-managed volumes the operator can mount on those drives. The IBM manual tape library is the manual IBM 3495 Tape Library Dataserver, which supports 3490 and 3490E Magnetic Tape Subsystems.

### **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 dataser or manual tape library dataser. You might know this library as the traditional tape library. 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 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, including 3420 reels, 3480, and 3590 cartridge system tapes.

### **Storage location**

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

## 4.62 Managing libraries and storage locations

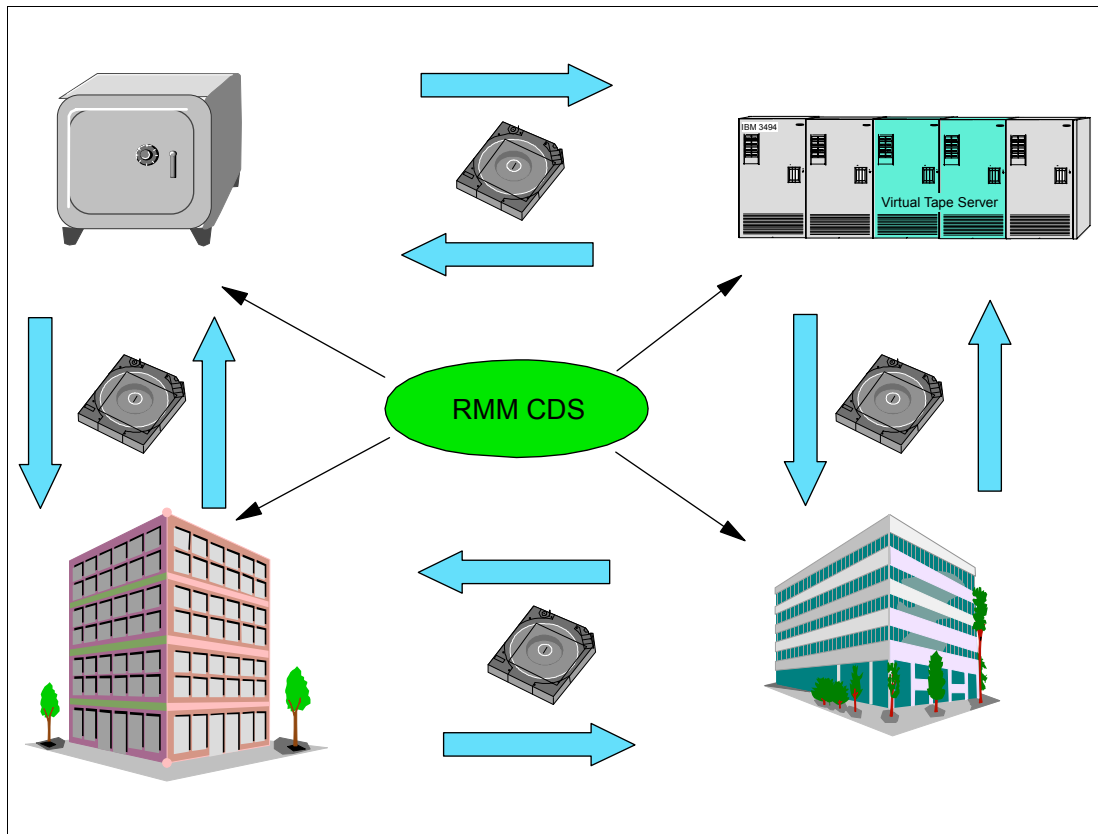


Figure 4-69 DFSMSrmm: managing libraries and storage locations

### Managing libraries and storage locations

DFSMSrmm records the complete inventory of the removable media library and storage locations in the DFSMSrmm control data set, which is a VSAM key-sequenced data set. In the control data set, DFSMSrmm records all changes made to the inventory (such as adding or deleting volumes), and also keeps track of all movement between libraries and storage locations. DFSMSrmm manages the movement of volumes among all library types and storage locations. This lets you control where a volume—and hence, a data set—resides, and how long it is retained.

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

- ▶ Automatically initializing and erasing volumes
- ▶ Recording information about volumes and data sets as they are used
- ▶ 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.

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



## System-managed storage

As your business expands, so do your needs for storage to hold your applications and data, and the costs of managing that storage. Storage costs include 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 people 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 them. Removable media, such as optical and tape storage, cost less per gigabyte (GB) than online storage, but 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, in this chapter we present:

- ▶ The z/OS storage-managed environment
- ▶ Benefits of a system-managed environment
- ▶ An overview of how DFSMS manages a storage environment based on installation policies
- ▶ How to set up a minimal SMS configuration and activate a DFSMS subsystem
- ▶ How to manage data using a minimal SMS configuration
- ▶ How to use Interactive Storage Management Facility (ISMF), an interface for defining and maintaining storage management policies

## 5.1 Storage management

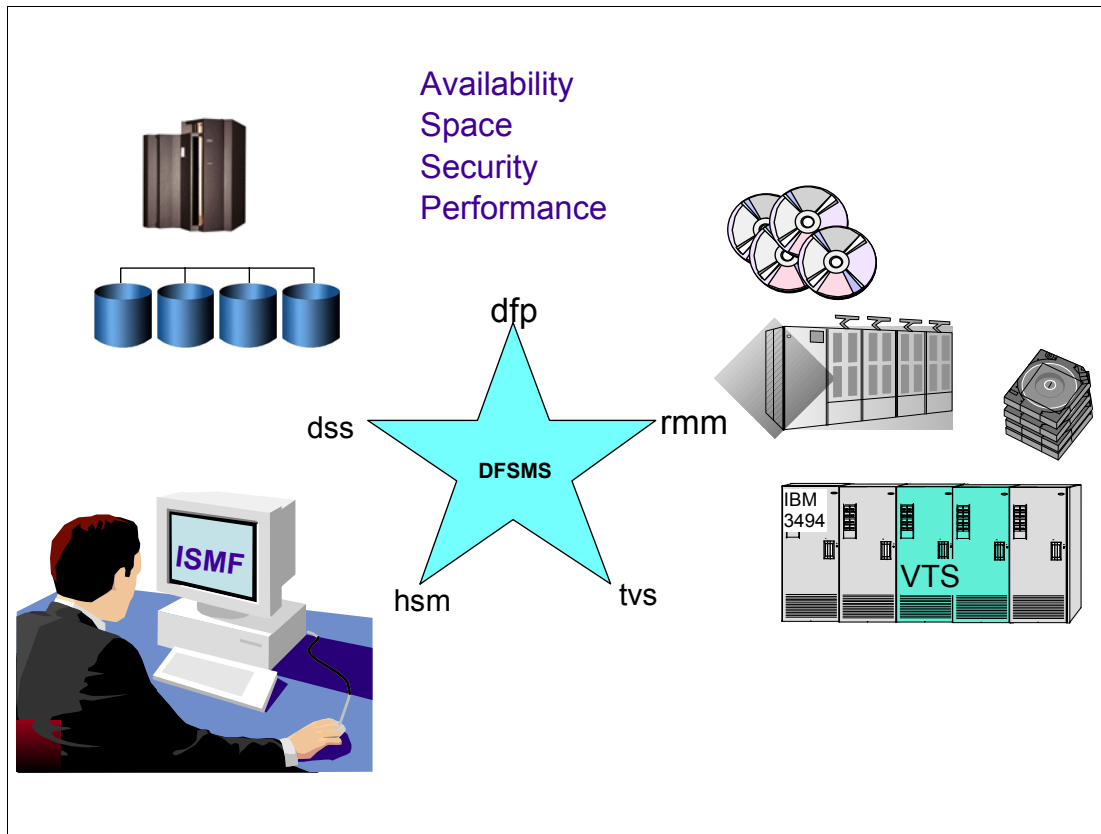


Figure 5-1 Managing storage with DFSMS

### 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.

### Managing storage with DFSMS

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 Interactive Storage Management Facility (ISMF) provides the user interface for defining and maintaining the policies.

Before we go further, let us distinguish DFSMS from the DFSMS *environment*.

## 5.2 DFSMS and DFSMS environment

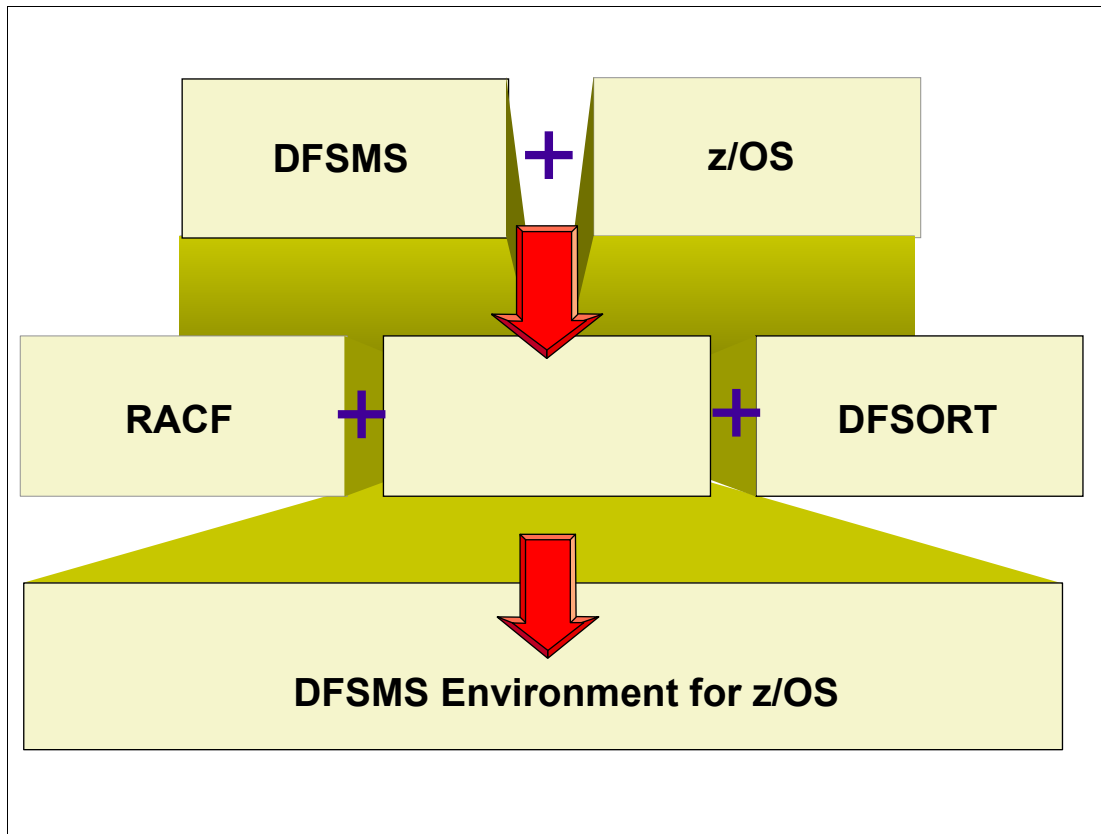


Figure 5-2 SMS environment

### DFSMS functional components

DFSMS is a set of products, and one of these products, DFSMSdfp, is mandatory for running z/OS. DFSMS comprises the base z/OS operating system, where DFSMS 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.

### DFSMS environment

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

The 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 ACS routines, ISMF code, and constructs, is located in DFSMSdfp. DFSMSshm and DFSMSdss are involved in the management class construct.

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*.

## 5.3 Benefits of system-managed storage

- ❑ Simplified data allocation
- ❑ Improved allocation control
- ❑ Improved I/O performance management
- ❑ Automated DASD space management
- ❑ Automated tape/optical space management
- ❑ Improved data availability management
- ❑ Simplified conversion of data to different device types

*Figure 5-3 Benefits of system-managed storage*

### **Benefits of SMS**

With SMS, you can define performance goals and data availability requirements, create model data definitions for typical data sets, and automate data backup. SMS can automatically assign, based on installation policy, 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.

### **Goals of system-managed storage**

The goals of system-managed storage are:

- ▶ 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 on managing data.

In the next section we describe the benefits of system-managed storage, which may be integrated with the goals. These benefits include simplified data allocation, improved allocation control, improved I/O performance management, and more. Let's look at them in more detail.



## **Benefits of system-managed storage**

### ***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 would have to specify the unit and volume on which the system should allocate the data set. The user would also have to calculate the amount of space required for the data set in terms of tracks or cylinders. This means the user has to know the track size of the device which will contain the data set. With system-managed storage, users can let the system select the specific unit and volume for the allocation. They can also specify size requirements in terms of megabytes or kilobytes. This means the user does not need to know anything about the physical characteristics of the devices in the installation.

### ***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 containing adequate free space. DFSMS 1.4 offers enhancements to avoid out-of-space by the relief of the SPACE requirements. You can also set a threshold for scratch tape volumes in tape libraries, to ensure enough cartridges are available in the tape library for scratch mounts.

### ***Improved Input/Output (I/O) performance management***

System-managed storage enables you to improve DASD I/O performance across the installation and at the same time reduce the need for manual tuning by defining performance goals for each class of data. You can use cache statistics recorded in System Management Facility (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 the IBM 3990 Model 3 and Model 6 Storage Controls, as well as other new models.

### ***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 reside 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***

System-managed storage enables you to fully use the capacity of your tape cartridges and to automate tape mounts. Using tape mount management techniques, DFSMSHsm can fill tapes to their capacity. With 3490E tape devices, Enhanced Capacity Cartridge System Tape, 36-track recording mode, and the improved data recording capability, you can increase the amount of data that can be written on a single tape cartridge.

You can also use the IBM 3495 or 3494 Tape Library Dataserver to automatically mount tape volumes and manage the inventory in an automated tape library. If you do not have an automated tape library dataserver, you can still take advantage of system-managed tape by using manual tape libraries and the 3495 Model M10 Tape Library Dataserver.

### ***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 different backup requirements to data residing on the same DASD volume. Thus, you do not have to treat all data on a single volume the same way.

You can use DFSMSHsm to automatically back up CICS/ESA and DATABASE 2 (DB2) databases, partitioned data sets extended (PDSEs), and physical sequential, partitioned, virtual storage access method (VSAM), hierarchical file system (HFS), and direct access data sets. You can also back up other types of data and use concurrent copy to maintain access to critical data sets while they are being backed up. Concurrent Copy, along with Backup-While-Open, has an added advantage in that it avoids the invalidation of a backup of a CICS VSAM KSDS due to a control area or control interval split.

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

### ***Simplified conversion of data to different 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 DFSMS environment do not need to specify the unit and volume which contains their data, it does not matter to them if their data resides on a specific volume or device type. This 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.

## 5.4 Establishing service level objectives

- ❑ What performance objectives are required by data
- ❑ When and how to back up data
- ❑ Whether data sets should be kept available for use during backup or copy
- ❑ How to manage backup copies kept for disaster recovery
- ❑ What to do with data that is obsolete or seldom used

Figure 5-4 Implementing your storage management policies

### Storage management policies

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.

The policies defined by your installation represent decisions about your resources, such as:

- ▶ What performance objectives are required by the transactions accessing the data  
Based on these objectives, you can try to better exploit 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 should 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  
Related data sets should be backed up in aggregated tapes. Each application should 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 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. Some examples are old masters, listings, and permanent work files. To select obsolete data for deletion using DFSMSdss, issue the DUMP command and the DELETE parameter, and force OUTDDNAME to DUMMY.

## 5.5 Implementing SMS policies

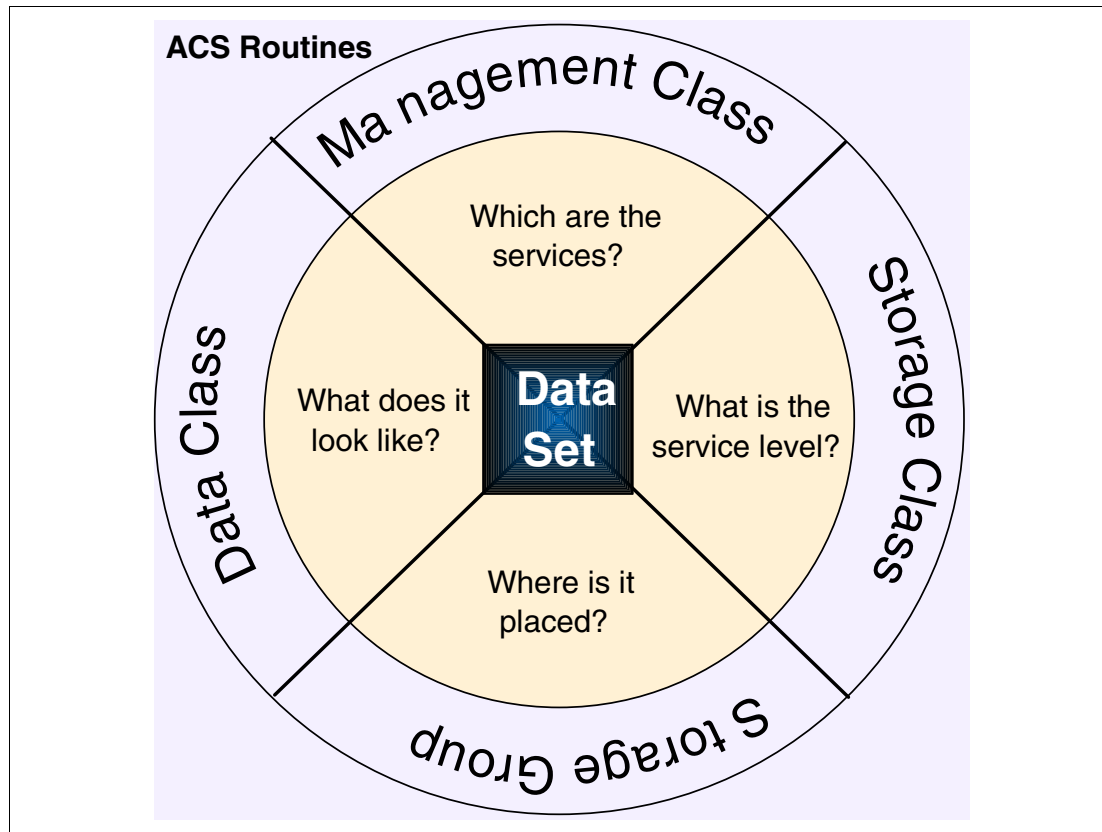


Figure 5-5 Creating SMS policies

### Implementing SMS policies

To implement a policy for managing storage, the storage administrator defines classes of space management, performance, and availability requirements for data sets. The storage administrator uses:

<b>Data class</b>	Data class is used to define model allocation characteristics for data sets.
<b>Storage class</b>	Storage class is used to define performance and availability goals.
<b>Management class</b>	Management class is used to define backup and retention requirements.
<b>Storage group</b>	Storage group is used to create logical groupings of volumes to be managed as a unit.
<b>ACS routines</b>	Automatic Class Selection (ACS) routines are used to assign class and storage group definitions to data sets and objects.

DFSMS facilitates all of these tasks by providing menu-driven, fill-in-the-blank panels with the Interactive Storage Management Facility (ISMF). ISMF panels make it easy to define classes, test and validate ACS routines, and perform other tasks to analyze and manage your storage. Note that many of these functions are available in batch through the NaviQuest tool.

For example, the administrator can define one storage class for data entities requiring high performance, and another for those requiring standard performance. Then, the administrator

writes Automatic Class Selection (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.

## 5.6 Monitoring SMS policies

- Monitor DASD use
- Monitor data set performance
- Decide when to consolidate free space on DASD
- Set policies for DASD or tape
- Use reports to manage your removable media

*Figure 5-6 Monitoring your SMS policies*

### **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 if 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.

## 5.7 Assigning data to be system-managed

	<b>DASD</b>	<b>Optical</b>	<b>Tape</b>
<b>Data Set (1)</b>	Assign Storage Class (SC)	Not applicable	Not system-managed (2)
<b>Object (3)</b>	Stored	Stored	Stored
<b>Volume</b>	Assign System Group (SG)	Define OAM Storage Groups (SG)	Assign Storage Group (SG) (4)

Figure 5-7 How to be system-managed

### How to be system-managed

Using SMS, you can automate storage management for individual data sets and objects, and for DASD, optical, and tape volumes. Figure 5-7 shows how a data set, object, DASD volume, tape volume, or optical volume becomes system-managed. The numbers shown in parentheses are associated with the following notes:

1. A DASD data set is system-managed if you assign it a storage class. If you do not assign a storage class, the data set is directed to a non-system-managed DASD or tape volume—one that is not assigned to a storage group.
2. You can assign a storage class to a tape data set to direct it to a system-managed tape volume. However, only the tape volume is considered system-managed, not the data set.
3. Objects are also known as byte-stream data, and 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, once written, the data is not changed or updated. However, the data can be referenced many times during its lifetime. Objects are processed by OAM. Each object has a storage class; therefore, objects are system-managed. The optical or tape volume on which the object resides is also system-managed.
4. Tape volumes are added to tape storage groups in tape libraries when the tape data set is created.



## 5.8 Using data classes

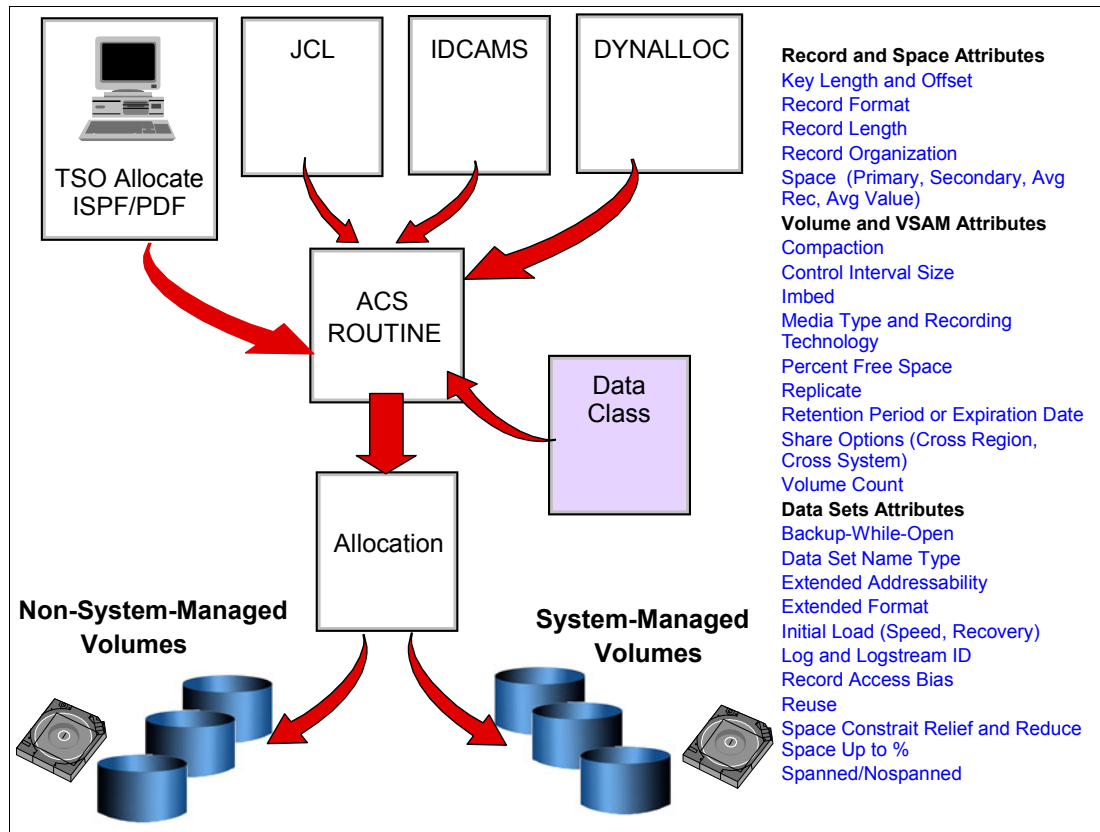


Figure 5-8 Using data classes

### Using data classes

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, some 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 need only specify the appropriate data classes to create standardized data sets.

You can assign a data class through:

- ▶ The DATACLAS parameter of JCL DD statement, **ALLOCATE** or **DEFINE** commands.
- ▶ Data class ACS routine to automatically assign a data class when the data set is being created. For example, data sets with the low-level qualifiers LIST, LISTING, OUTLIST, or LINKLIST are usually utility output data sets with similar allocation requirements, and can all be assigned the same data class.

You can override some data set attributes assigned in the data class, but you cannot change the data class name assigned through an ACS routine.

Even though data class is optional, we usually recommend that you 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. 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, we recommend that you do not assign a data class via 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.

## 5.9 Using storage classes

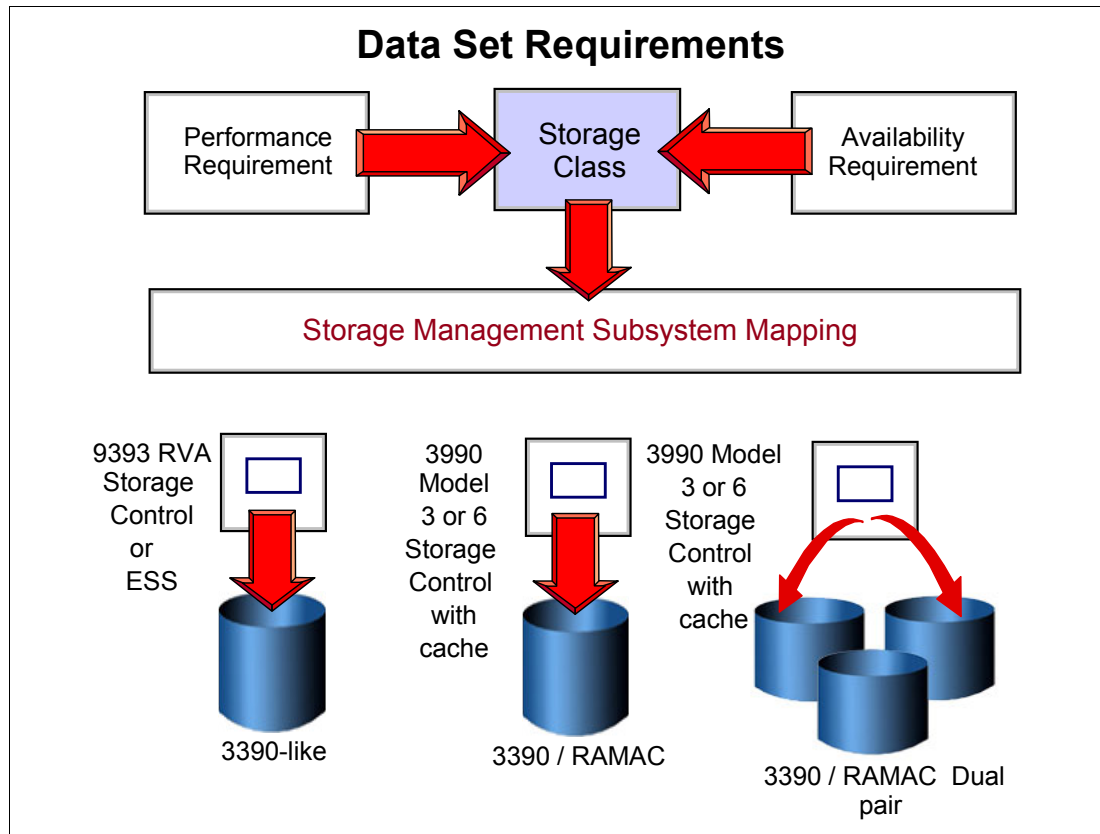


Figure 5-9 Choosing volumes that met availability requirements

### Using storage classes

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 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 cache and dual copy) can only be met by DASD volumes attached through one of the following storage control units or a similar device:

- ▶ 3990-3 or 3990-6
- ▶ RAMAC Array Subsystem
- ▶ Enterprise Storage Server (ESS)

Figure 5-9 shows storage control unit configurations and their storage class attribute values.

With 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 the same, 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.

The 3990-6 and 9390 concurrent copy function enables you to take point-in-time copies of data by using a cache sidefile that is loaded with the time-zero version of the data. Time-zero refers to the state of the data when the concurrent copy session is started and before it gets updated. Before a record is updated, it is copied to the cache sidefile, thus creating a *before update* version or time-zero version of the record.

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) in order 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:

- ▶ The STORCLAS parameter of the JCL DD statement, **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.

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 subsequently opened. However, the definition changes do not affect the location or allocation characteristics of existing data sets.

## 5.10 Using management classes

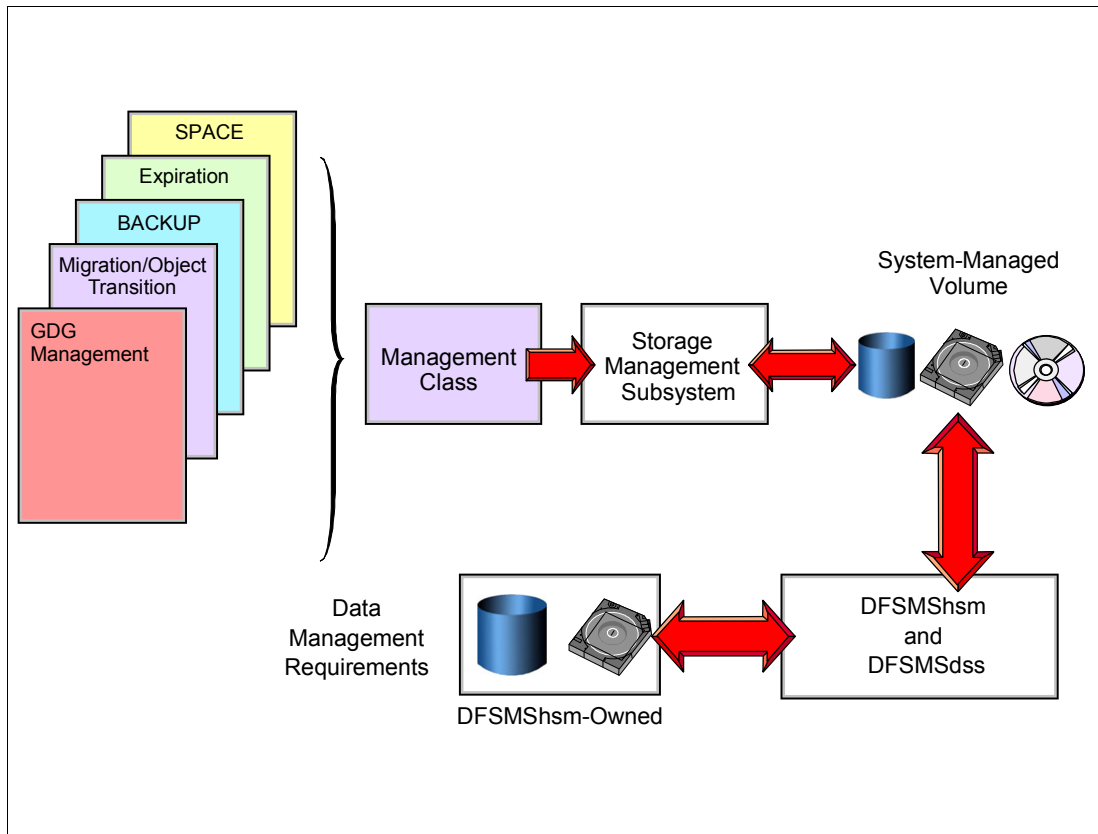


Figure 5-10 Using management classes

### Using management classes

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

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

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 executed by DFSMSHsm and DFSMSDss programs. Figure 5-11 on page 207 shows the sort of functions an installation may define in a management class.

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

- ▶ 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, **ALLOCATE** or **DEFINE** command. You cannot override management class *attributes* via 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.

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:

- ▶ Assign a management class when it is stored, or
- ▶ Assign a new management class when the object is moved, or
- ▶ Change the management class by using the OAM Application Programming Interface (OSREQ CHANGE function)

The ACS routines can override this assignment for objects.

## 5.11 Management class functions

- Allow early migration for old generations of GDG
- Delete selected old/unused data sets from DASD volumes
- Release allocated but unused space from data sets
- Migrate unused data sets to tape or DASD volumes
- Specify how often to back up data sets, and whether concurrent copy should be used for backups
- Specify how many backup versions to keep for data sets
- Specify how long to save backup versions
- Specify the number of versions of ABARS to keep and how to retain those versions
- Establish the expiration date/transition criteria for objects
- Indicate if automatic backup is needed for objects

Figure 5-11 Management class functions

### 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. For example:

- ▶ Control the migration of CICS user databases, DB2 user databases and archive logs.
- ▶ Test systems and their associated data sets.
- ▶ IMS archive logs.
- ▶ 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 should have a backup copy made when the OAM Storage Management Component (OSMC) is executing.

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

## 5.12 Using storage groups

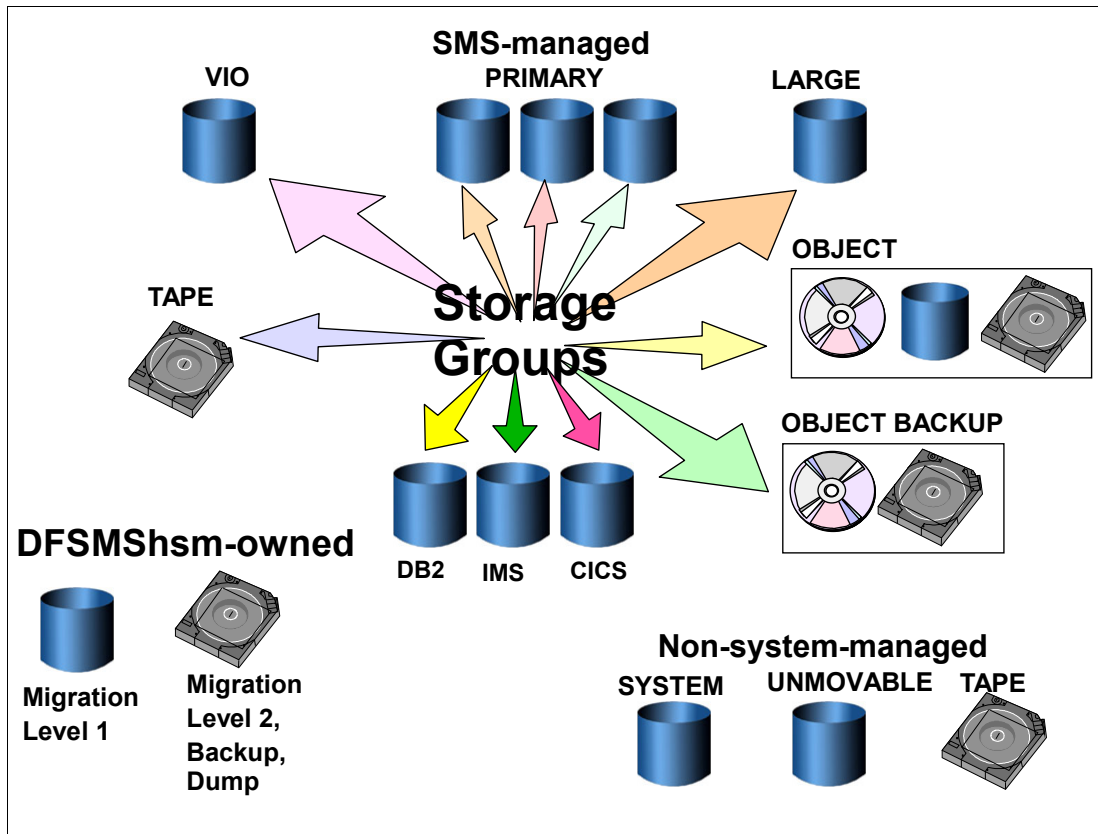


Figure 5-12 Grouping storage volumes for specific purposes

### Storage groups

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

- ▶ 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 which 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 some of the storage group attributes to determine if the volumes in the storage group are eligible for automatic space or availability management.

Figure 5-12 shows an example of how an installation can group storage volumes according to their objective. 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.

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 or not to honor their unit and volume request is an installation decision, but we recommend that you 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 automatic class selection routines.

For objects, there are 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:

<b>Primary</b>	Volumes online, below threshold, that meet all the specified criteria in the storage class.
<b>Secondary</b>	Volumes that do not meet all the criteria for primary volumes.
<b>Tertiary</b>	When the number of volumes in the storage group is less than the number of volumes that are requested.
<b>Rejected</b>	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, when no volumes are available, SMS selects from the tertiary list.

SMS interfaces with the system resource 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 jobstep. This is useful for batch processing when the data set is accessed immediately after creation. It is, however, not useful for database data that is reorganized at off-peak hours.

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*. Once 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.

## 5.13 Using aggregate backup and recovery support (ABARS)

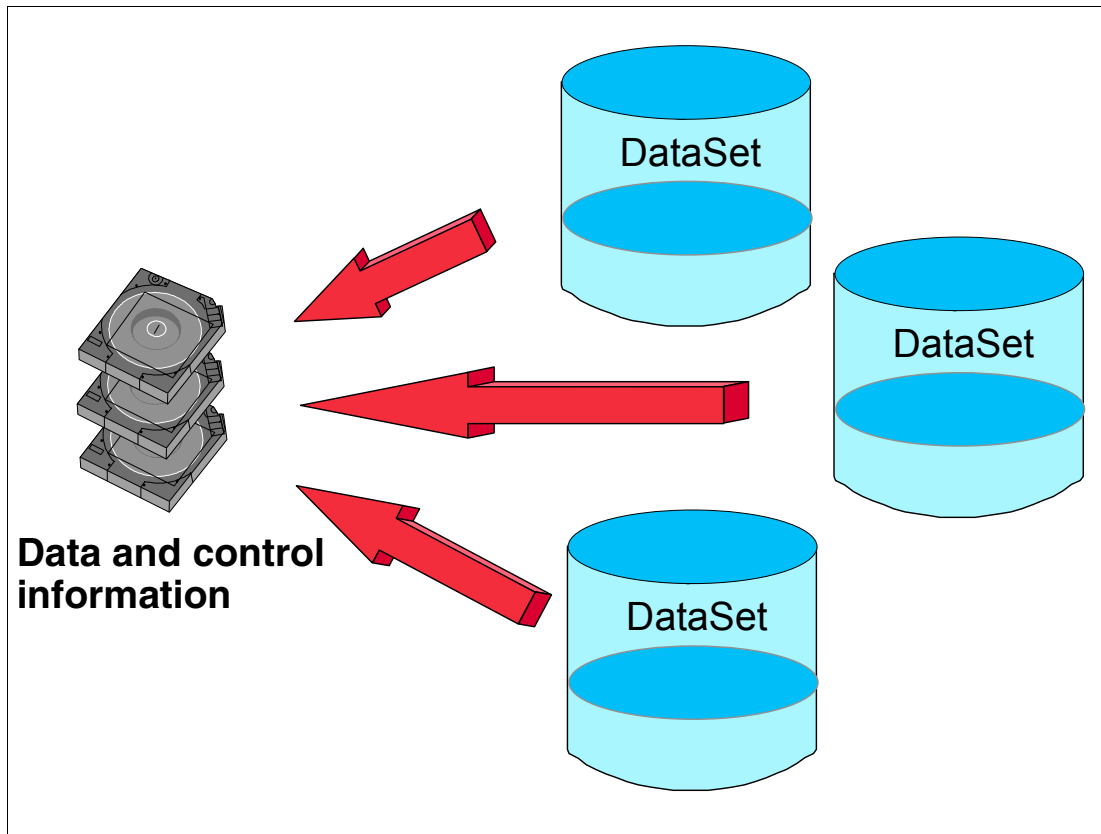


Figure 5-13 ABARS

### Aggregate backup and recovery support (ABARS)

*Aggregate backup and recovery support*, 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:

- ▶ 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 in conjunction with personnel moves or workload balancing
- ▶ Duplicate a problem at another site

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, provided that the groups do not contain data which requires that SMS be active, such as PDSEs. You can use aggregate groups to transfer applications to other data processing installations, or to migrate applications to newly-installed DASD volumes. You can transfer the application's migrated data, along with its active data, without recalling the migrated data.

## 5.14 Automatic Class Selection (ACS) routines

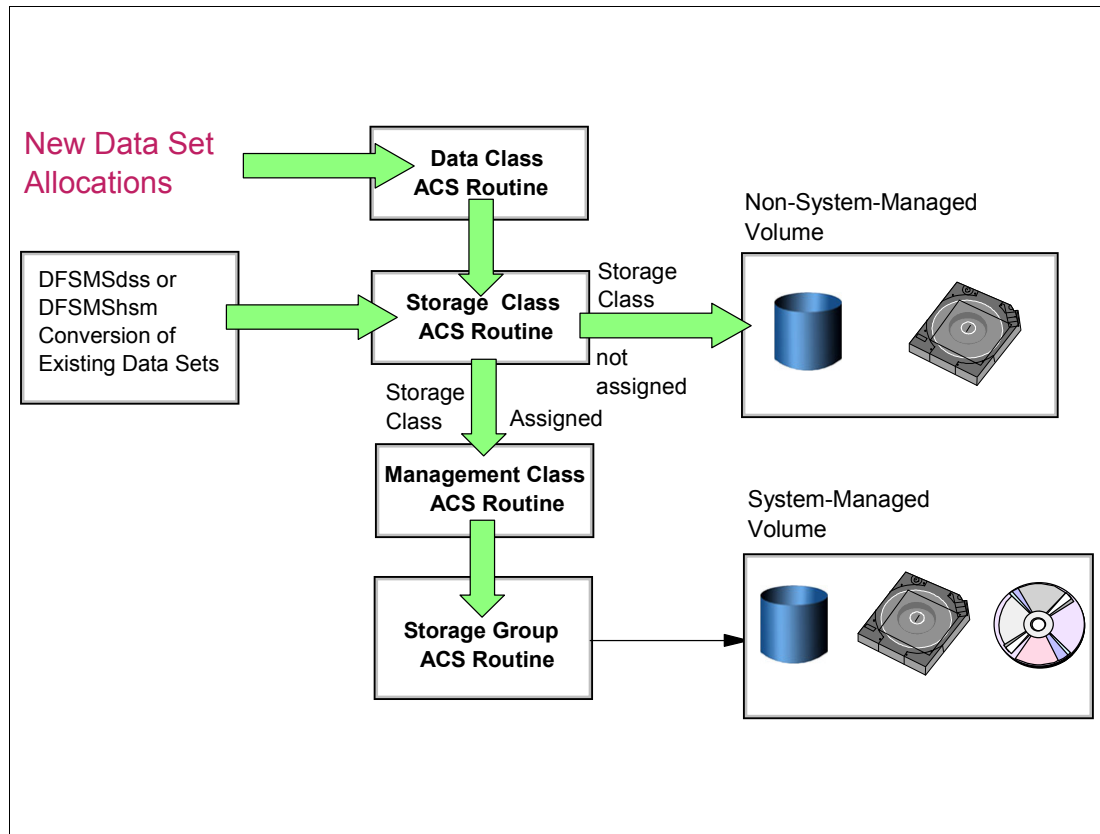


Figure 5-14 Using ACS routines

### Using Automatic Class Selection routines

You use automatic class selection (ACS) routines to assign classes (data, storage, and management) and storage group definitions to data sets, database data, and objects. You write ACS routines using the ACS language, which is a high-level programming language. Once written, you use the ACS translator to translate the routines 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 `&LLQ` to make assignments based on the low-level qualifier of the data set or object collection name.

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

You 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, you use the `&STORCLAS` variable to assign a storage class to a data set or object.

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 write 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 5-14 on page 212 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.

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 should not be system-managed, 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.

## 5.15 SMS configuration

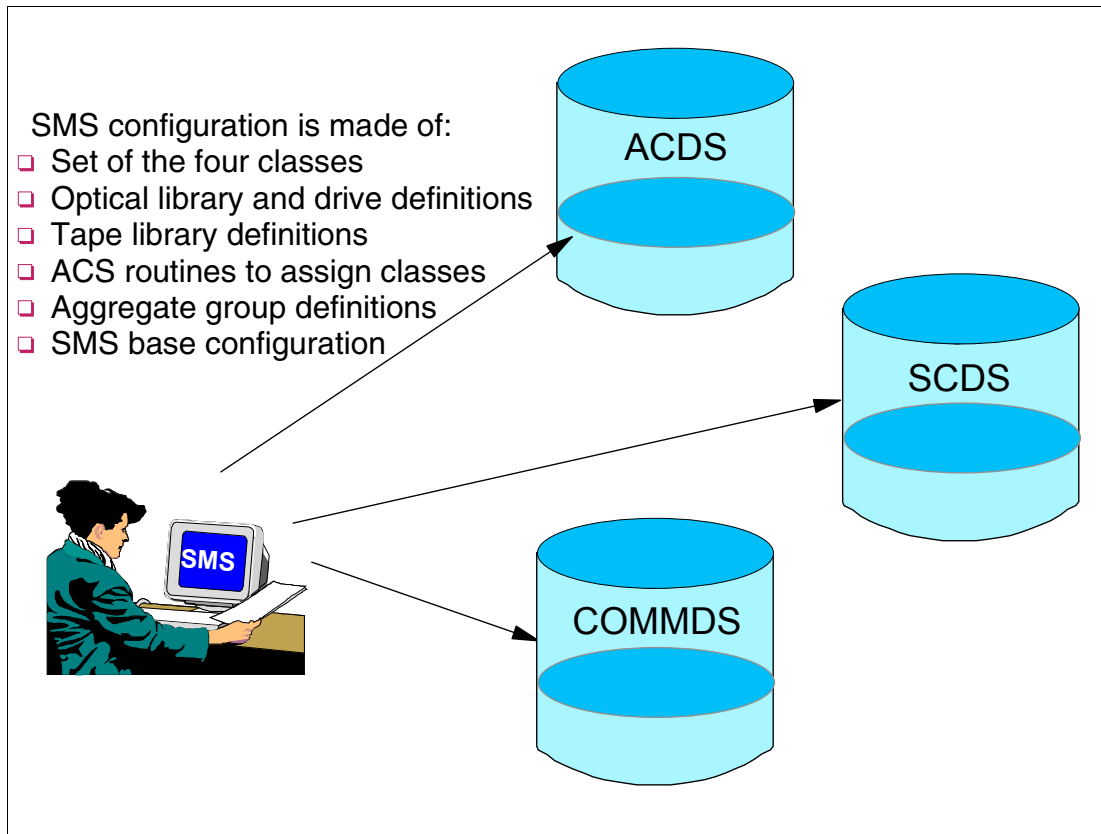


Figure 5-15 Defining the SMS configuration

### SMS configuration

An SMS configuration is composed of:

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

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)

## Source Control Data Set (SCDS)

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.

You use the SCDS to develop and test but, before activating a configuration, retain at least one prior configuration should you need to regress to it because of error. The SCDS is never used to manage allocations.

## Active Control Data Set (ACDS)

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:

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

The ACDS must reside on a shared device to ensure that all systems in the installation use the same active configuration.

We recommend that you have extra ACDSs in case a hardware failure causes the loss of your primary ACDS. It must reside on a shared device, accessible to all systems, to ensure that they share a common view of the active configuration. Do not have the ACDS reside 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. You should also create a backup ACDS in case of hardware failure or accidental data loss or corruption.

## Communications Data Set (COMMDS)

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. Although only one COMMDS is used at a time for an SMS installation, we recommend that you have more COMMDSs on different volumes for recovery purposes.

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.

## SMSplex

An ACDS and COMMDS must reside on a shared volume, accessible for all systems in the SMS complex (SMSplex). DFSMS components exploit the Coupling Facility capabilities to provides services such as the following:

- ▶ Enhanced Catalog Sharing (ECS), which uses the CF cache structure to hold change information for shared catalogs. This eliminates catalog-related I/O to the VVDS, resulting in better performance for both user and master catalog requests.
- ▶ Sharing VSAM data sets in a Parallel Sysplex using VSAM RLS and TVS.
- ▶ DFSMSHsm using a common recall queue to recall data sets in the Parallel Sysplex.

## 5.16 Implementing DFSMS

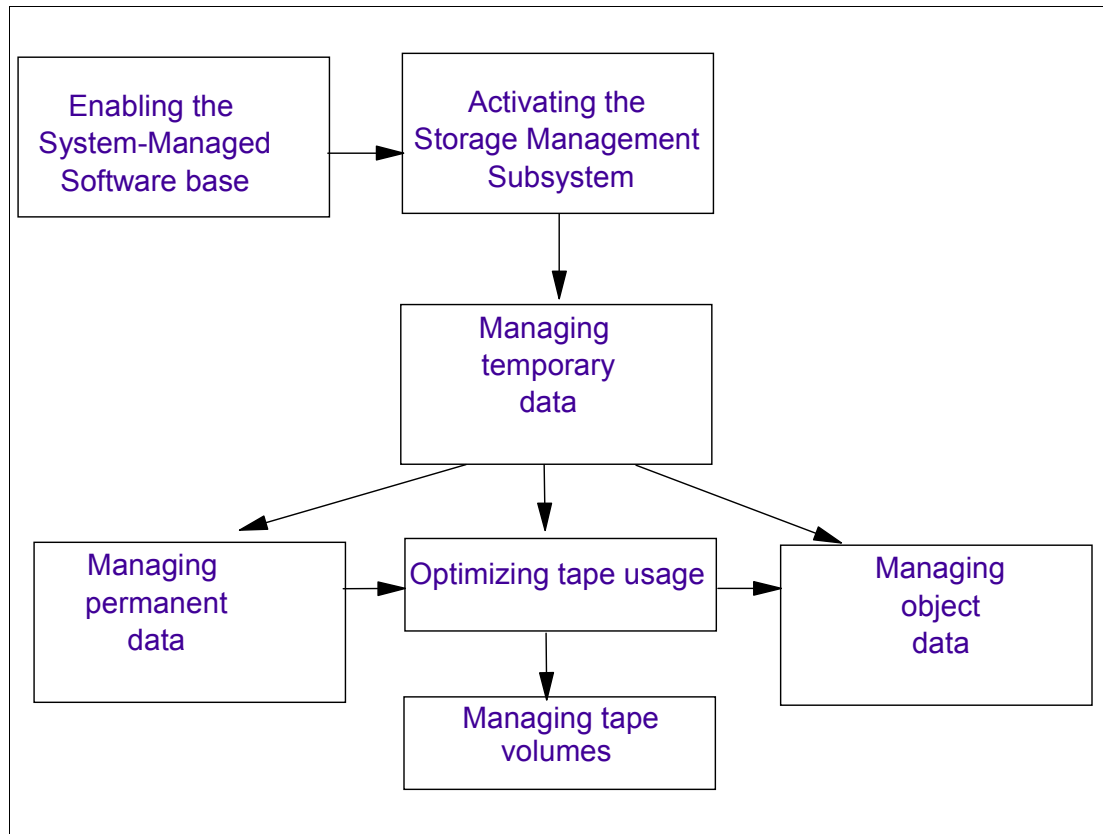


Figure 5-16 SMS implementation phases

### 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 you are most interested in first. For example, you can:

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

### DFSMS implementation phases

There are five major DFSMS implementation phases:

- ▶ Enabling the software base
- ▶ Activating the storage management subsystem
- ▶ Managing temporary data
- ▶ Managing permanent data
- ▶ Managing tape data

In this redbook we present an overview of the steps needed to activate, and manage data with, a minimal SMS configuration, without affecting your JCL or data set allocations. To implement DFSMS in your installation, however, you must refer to *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407.



## 5.17 Steps to activate a minimal SMS configuration

- ❑ Allocate the SMS control data sets
- ❑ Define to GRS the resource names for the SMS control data sets
- ❑ Define the system group
- ❑ Define a minimal SMS configuration:
  - Create the SCDS base data set
  - Create classes, storage groups and respective ACS routines
- ❑ Define the SMS subsystem to z/OS
- ❑ Start SMS and activate the SMS configuration

Figure 5-17 Steps to activate a minimal SMS configuration

### Steps to activate a minimal SMS configuration

Activating a minimal configuration lets you experience managing an SMS configuration without affecting your JCL or data set allocations. This establishes an operating environment for the storage management subsystem, without data sets becoming system-managed.

The steps needed to activate the minimal configuration are presented in Figure 5-17. When implementing DFSMS, beginning by implementing a minimal configuration allows you to:

- ▶ Gain experience with ISMF applications for the storage administrator, since you use ISMF applications to define and activate the SMS configuration.
- ▶ Gain experience with the operator commands that control operation of resources controlled by SMS.
- ▶ Learn how the SMS base configuration can affect allocations for non-system-managed data sets. The base configuration contains installation defaults for data sets:
- ▶ For non-system-managed data sets, you can specify default device geometry to ease the conversion from device-dependent space calculations to the device-independent method implemented by SMS.
- ▶ For system-managed data sets, you can specify a default management class to be used for data sets that are not assigned a management class by your management class ACS routine.
- ▶ Use simplified JCL.
- ▶ Implement allocation standards, since you can develop a data class ACS routine to enforce your standards.

## 5.18 Allocating SMS control data sets

```
//ALLOC EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEFINE CLUSTER(NAME(YOUR.OWN.SCDS) LINEAR VOLUME(D65DM1) -
    TRK(25 5) SHAREOPTIONS(2,3)) -
    DATA(NAME(YOUR.OWN.SCDS.DATA))

  DEFINE CLUSTER(NAME(YOUR.OWN.ACDS) LINEAR VOLUME(D65DM2) -
    TRK(25 5) SHAREOPTIONS(3,3)) -
    DATA(NAME(YOUR.ACDS.DATA))

  DEFINE CLUSTER(NAME(YOUR.OWN.COMMDS) LINEAR VOLUME(D65DM3) -
    TRK(1 1) SHAREOPTIONS(3,3)) -
    DATA(NAME(YOUR.OWN.COMMDS.DATA))
```

Figure 5-18 Using IDCAMS to create SMS control data sets

### Calculating the SCDS and ACDS sizes

The size of the ACDS and SCDS may allow constructs for up to 32 systems. Be sure to allocate sufficient space for the ACDS and SCDS, since insufficient ACDS size can cause errors such as failing SMS activation. See *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402 for the formula used to calculate the appropriate SMS control data set size.

### Calculating the COMMDS size

The size of the communications data set (COMMDS) increased in DFSMS 1.3, because the amount of space required to store system-related information for each volume increased. To perform a precise calculation of the COMMDS size, use the formula provided in *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

### Defining the control data sets

After you have calculated their respective sizes, define the SMS control data sets using access method services. The SMS control data sets are VSAM linear and you define them using IDCAMS **DEFINE** command, as shown in Figure 5-18. Because these data sets are allocated *before* SMS is activated, space is allocated in tracks. Allocations in KBs or MBs are only supported when SMS is active.

Specify **SHAREOPTIONS(2,3)** only for the SCDS. This lets one update-mode user operate simultaneously with other read-mode users between regions.

Specify SHAREOPTIONS(3,3) for the ACDS and COMMDS. These data sets must be shared between systems that are managing a shared DASD configuration in a DFSMS environment.

### **Define GRS resource names for active SMS control data sets**

If you plan to share SMS control data sets between systems, consider the effects of multiple systems sharing these data sets. Access is serialized by the use of RESERVE, which locks out access to the *entire device volume* from other systems until the RELEASE is issued by the task using the resource. This is undesirable, especially when there are other data sets in the volume.

A RESERVE is issued when SMS is updating:

- ▶ COMMDS data set with space statistics at the expiration time interval specified in IGDSMSxx PARMLIB member.
- ▶ ACDS data set due to changes in the SMS configuration.

Place the resource name IGDCDSXS in the RESERVE conversion RNL as a generic entry to convert the RESERVE/RELEASE to ENQueue/DEQueue. This minimizes delays due to contention for resource and prevent deadlocks associated with the **VARY SMS** command.

Prior to DFSMS 1.5, you should not have more than one SMSplex controlled by one GRSplex. The name of the ENQ resource was not associated with the ACDS name, which caused performance problems due to false contention (different resource, but same name). In DFSMS 1.5, the RNAME is appended by the BCDS name.

If there are multiple SMS complexes within a global resource serialization complex, be sure to use unique COMMDS and ACDS data set names to prevent false contention. For information on allocating COMMDS and ACDS data set names, see *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407.

### **Defining the system group**

A *system group* is a group of systems within an SMS complex that have similar connectivity to storage groups, libraries, and volumes. When a Parallel Sysplex name is specified and used as a system group name, the name applies to all systems in the Parallel Sysplex except for those systems defined as part of the Parallel Sysplex that are explicitly named in the SMS base configuration. The system group is defined using ISMF when defining the base configuration.

## 5.19 Defining the SMS base configuration

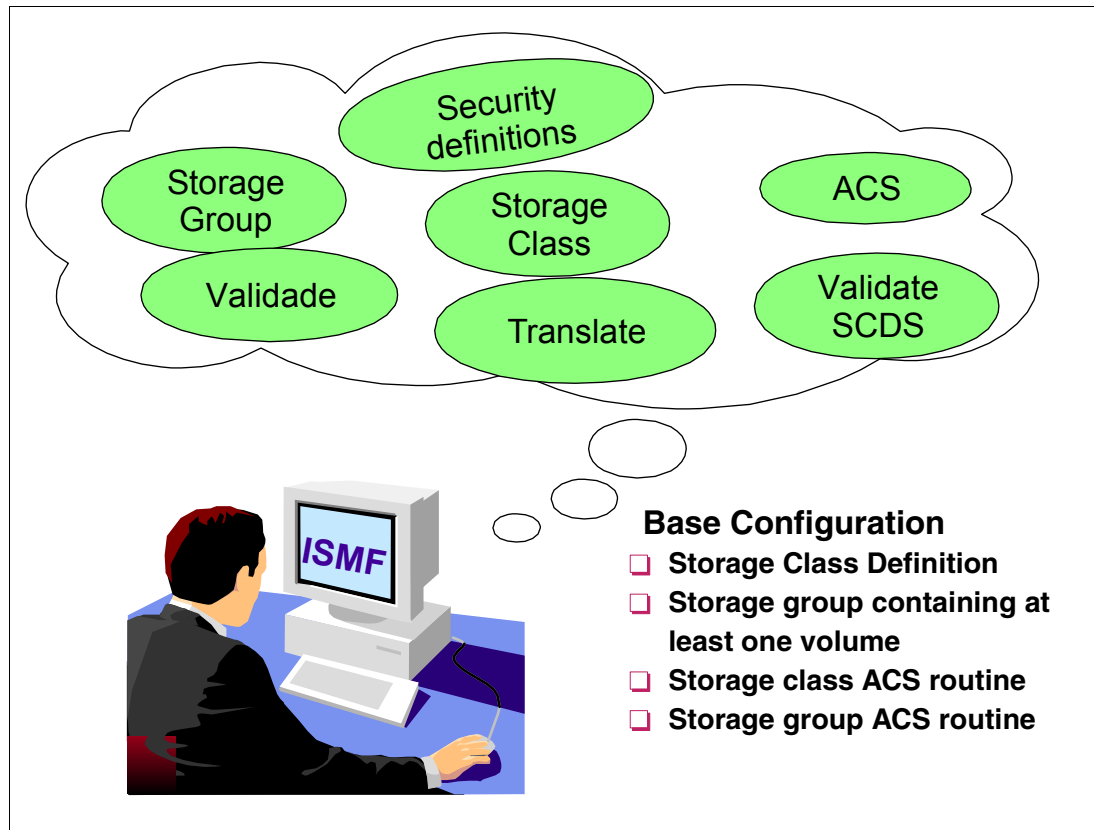


Figure 5-19 Minimal SMS configuration

### Protecting the DFSMS environment

Before defining the SMS base configuration, you have to protect access to the SMS control data sets, programs and functions. For example, some functions in the ISMF are related only to storage administration tasks and you must protect your storage environment from unauthorized access. You can protect the DFSMS environment with RACF.

RACF controls access to the following functions:

- ▶ System-managed data sets
- ▶ SMS control data sets
- ▶ SMS functions and commands
- ▶ Fields in the RACF profile
- ▶ SMS classes
- ▶ ISMF functions

For more information, refer to *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

### Defining the SMS base configuration

After creating the SCDS data set with IDCAMS and setting up the security to the DFSMS environment, you use the ISMF **Control Data Set** option to define the SMS base configuration, which contains information such as:

- ▶ Default management class

- ▶ Default device geometry.
- ▶ The systems in the installation for which SMS manages storage using that configuration

To define a minimal configuration, you must do the following:

- ▶ Define a storage class.
- ▶ Define a storage group containing at least one volume. (The volume does not have to exist, as long as you do not direct allocations to either the storage group or the volume.)
- ▶ Create their respective ACS routines.

Defining a data class, a management class and creating their respective ACS routines are not required for a valid SCDS. However, because of the importance of the *default management class*, we recommend that you include it in your minimal configuration.

For a detailed description of SMS classes and groups, see *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407.

The DFSMS product tape contains a set of sample ACS routines. The appendix of *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402 contains sample definitions of the SMS classes and groups that are used in the sample ACS routines. The starter set configuration can be used as a model for your own SCDS. For a detailed description of base configuration attributes and how to use ISMF to define its contents, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

## Defining the storage class

You must define at least one storage class name to SMS. Because a minimal configuration does not include any system-managed volumes, no performance or availability information need be contained in the minimal configuration's storage class. Specify an artificial storage class, NONSMS. This class is later used by the storage administrator to create non-system-managed data sets on an exception basis.

In the storage class ACS routine, the &STORCLAS variable is set to a null value to prevent users from coding a storage class in JCL before you want to have system-managed data sets.

You define the class using ISMF. Select **Storage Class** in the primary menu. Then you can define the class, NONSMS, in your configuration in one of two ways:

1. Select option **3 Define** in the Storage Class Application Selection panel. The CDS Name field must point to the SCDS you are building.
2. Select option **1 Display** in the Storage Class Application Selection panel. The CDS Name field must point to the starter set SCDS. Then, in the displayed panel, use the **COPY** line operator to copy the definition of NONSMS from the starter set SCDS to your own SCDS.

## Defining the storage group

You must define at least one pool storage group name to SMS, and at least one volume serial number to this storage group. A storage group with no volumes defined is not valid. This volume serial number should be for a *nonexistent volume* to prevent the occurrence of JCL errors from jobs accessing data sets using a specific volume serial number.

Defining a non-existent volume lets you activate SMS without having any system-managed volumes. No data sets are system-managed at this time. This condition provides an opportunity to experiment with SMS without any risk to your data.

Define a storage group (for example, NOVOLS) in your SCDS. A name like NOVOLS is useful because you know it does not contain valid volumes.

## **Defining the default management class**

Define a default management class and name it STANDEF to correspond with the entry in the base configuration. We recommend that you specifically assign all system-managed data to a management class. If you do not supply a default, DFSMSHsm uses two days on primary storage, and 60 days on migration level 1 storage, as the default.

No management classes are assigned when the minimal configuration is active. Definition of this default is done here to prepare for the managing permanent data implementation phase.

The management class, STANDEF, is defined in the starter set SCDS. You can copy its definition to your own SCDS in the same way as the storage class, NONSMS.

## 5.20 Creating ACS routines

```

Storage class ACS routine
PROC STORCLAS
/*****
/* C H A N G E H I S T O R Y                               */
/* -----*/
/* DATE RESP DESCRIPTION OF CHANGE:                       */
/* -----*/
/* PURPOSE:
/* THIS ROUTINE ASSIGNS A NULL STORAGE CLASS TO ALL DATA SETS */
/* INPUT: THE FOLLOWING ACS VARIABLES ARE REFERENCED: NONE   */
/* OUTPUT: NULL STORAGE CLASS.                               */
/* RETURN CODES: 0 IS THE ONLY RETURN CODE.                */
/* DATA SET ALLOCATIONS ARE NOT FAILED IN THIS ROUTINE.   */
*****/
SET &STORCLAS = ''
END
/* END OF STORAGE CLASS ROUTINE PROC*/

PROC STORGRP
/*****
/* C H A N G E H I S T O R Y                               */
/* DATE RESP DESCRIPTION OF CHANGE                       */
/* -----*/
/* IT ONLY EXISTS TO SATISFY THE REQS FOR SG ACS ROUTINE  */
/* A STORAGE GROUP CONTAINING NO REAL DASD VOLUMES IS ASSIGNED, */
/* NOVOLS.
/* INPUT: ACS VARIABLES ARE REFERENCED: NONE
/* OUTPUT: THE NOVOLS STORAGE GROUP IS ASSIGNED.
/* RETURN CODE: 0 IS THE ONLY RETURN CODE
/* ALLOCATIONS ARE NOT FAILED IN THIS ROUTINE.
*****/
SET &STORGRP = NOVOLS /* ASSIGN TO SG WITH NO REAL VOLUMES */
END
- /*****
Storage group ACS routine

```

Figure 5-20 Sample ACS routines for a minimal SMS configuration

### Creating ACS routines

After you define the SMS classes and group, develop their respective ACS routines. For a minimal SMS configuration, in the storage class ACS routine, you assign a null storage class, as shown in the sample storage class ACS routine in Figure 5-20. The storage class ACS routine ensures that the storage class read/write variable is always set to null. This prevents users from externally specifying a storage class on their DD statements (STORCLASS keyword), which would cause the data set to be system-managed before you are ready.

The storage group ACS routine will never run if a null storage class is assigned. Therefore, no data sets are allocated as system-managed by the minimal configuration. However, you must code a trivial one to satisfy the SMS requirements for a valid SCDS. After you have written the ACS routines, use ISMF to translate them into executable form.

1. If you do not have the starter set, allocate a fixed-block PDS or PDSE with LRECL=80 to contain your ACS routines. Otherwise, start with the next step.
2. On the ISMF Primary Option Menu, select **Automatic Class Selection** to display the ACS Application Selection panel.
3. Select option **1 Edit**. When the next panel is shown, enter in the Edit panel the name of the PDS or PDSE data set you create to contain your source ACS routines. A sample storage class ACS routine is shown Figure 5-20.

## Translating the ACS routines

The translation process checks the routines for syntax errors and converts the code into an ACS object. If the code translates without any syntax errors, then the ACS object is stored in the SCDS. For translate:

1. From the ISMF ACS Application Selection Menu panel, select **2 Translate** and press Enter.
2. Enter your SCDS data set name, the PDS or PDSE data set name containing the ACS source routine, and a data set name to hold the translate output listing. When the listing data set does not exist, it is created automatically.

## Validating the SCDS

When you validate your SCDS, you verify that all classes and groups assigned by your ACS routines are defined in the SCDS. To validate the SCDS:

1. From the ISMF Primary Option Menu panel, select **Control Data Set** and press Enter.
2. Enter your SCDS data set name and select **4 Validate**.

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



## 5.21 DFSMS setup for z/OS

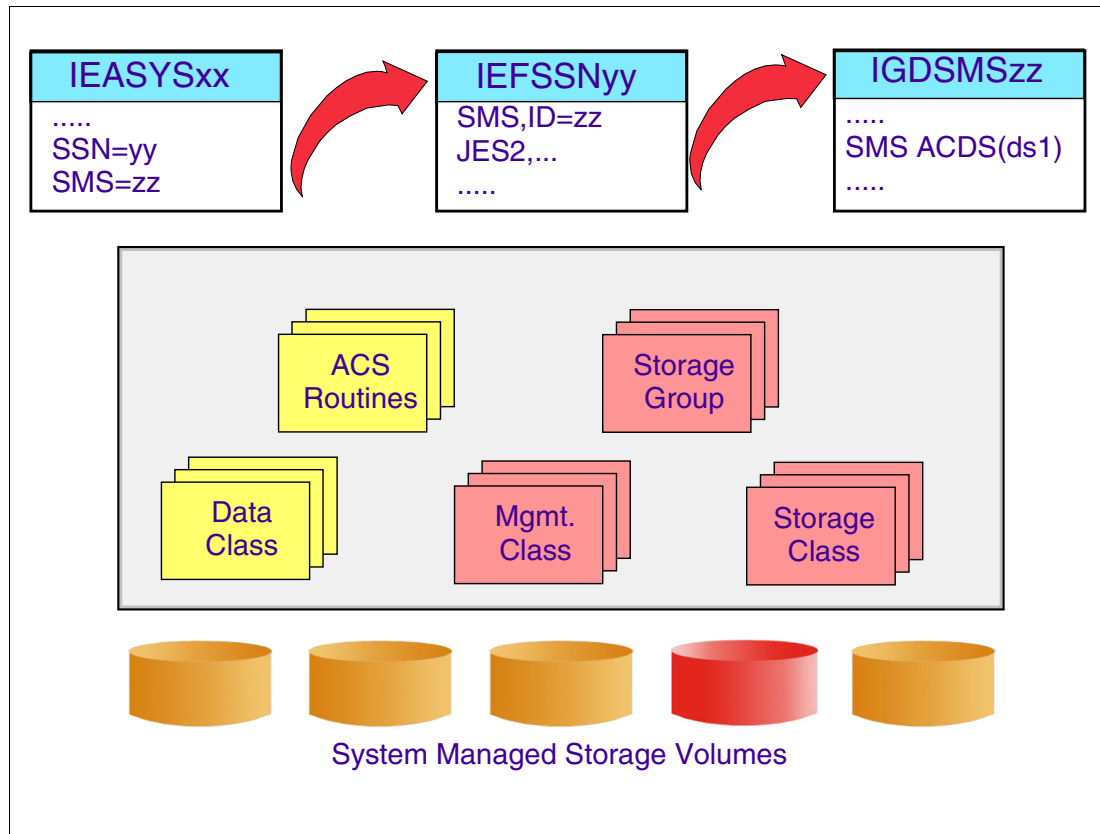


Figure 5-21 DFSMS setup for z/OS

### DFSMS setup

In preparation for starting SMS, update the following PARMLIB members to define SMS to z/OS:

- |                 |   |
|-----------------|---|
| <b>IEASYSxx</b> | Verify the suffix of the IEFSSNyy in use and add the <code>SMS=xx</code> parameter, where xx is the IGDSMS member name suffix.  |
| <b>IEFSSNyy</b> | <p>You can activate SMS only after you define the SMS subsystem to z/OS. To define SMS to z/OS, you must place a record for SMS in the IEFSSNxx PARMLIB member.</p> <p>IEFSSNxx defines how z/OS activates the SMS address space. You can code an IEFSSNxx member with keyword <i>or</i> positional parameters, but not both. We recommend using keyword parameters. We recommend that you place the SMS record <i>before</i> the JES2 record in IEFSSNxx in order to start SMS before starting the JES2 subsystem.</p> |
| <b>IGDSMSzz</b> | For each system in the SMS complex, you must create an IGDSMSxx member SYS1.PARMLIB. The IGDSMSzz member contains SMS initialization control information. The suffix has a default value of 00.   |

Every SMS system must have an IGDSMSzz member in SYS1.PARMLIB that specifies a required ACDS and COMMDS control data set pair. This ACDS and COMMDS pair is used if the COMMDS of the pair does not point to another COMMDS.

If the COMMDS points to another COMMDS, the referenced COMMDS is used. This referenced COMMDS might contain the name of an ACDS that is different from the one specified in the IGDSMSzz. If so, the name of the ACDS is obtained from the COMMDS rather than from the IGDSMSzz to ensure that the system is always running under the most recent ACDS and COMMDS.

If the COMMDS of the pair refers to another COMMDS during IPL, it means a more recent COMMDS has been used. SMS uses the most recent COMMDS to ensure that you cannot IPL with a down-level configuration.

The data sets that you specify for the ACDS and COMMDS pair must be the same for every system in an SMS complex. Whenever you change the ACDS or COMMDS, update the IGDSMSzz for every system in the SMS complex so that it specifies the same data sets.

IGDSMSzz has many parameters. For a complete description of SMS parameters, see *z/OS MVS Initialization and Tuning Reference, SA22-7592*, and *z/OS DFSMSdfp Storage Administration Reference, SC26-7402*.

## 5.22 Starting SMS

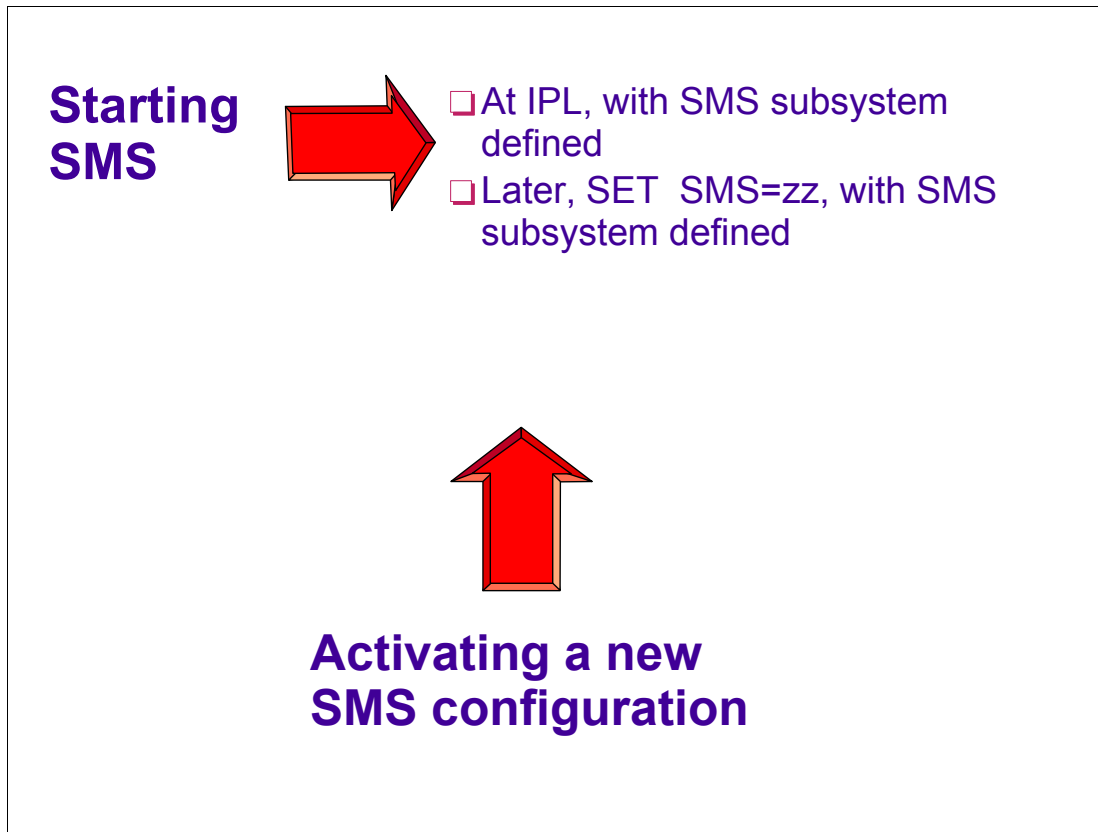


Figure 5-22 Starting SMS and activating a new SMS configuration

### 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 starts SMS automatically.
- ▶ With SMS defined as a valid subsystem to z/OS, IPL the system. Start SMS later, using the **SET SMS=yy** MVS operator command.

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

### 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. Note that SMS must be active before you use one of these methods:

1. Activating an SMS configuration from ISMF
  - From the ISMF Primary Option Menu panel, select **Control Data Set**.
  - 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.

## 2. Activating an SMS configuration from the operator console

- From the operator console, enter the 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.

## 5.23 Control SMS processing with operator commands

```

❑ SETSMS
❑ SET SMS=xx
❑ VARY SMS
❑ DEVSERV
❑ DISPLAY
DEVSERV P,25C1,9
IEE459I 11.39.06 DEVSERV PATHS 836
UNIT DTYPE M CNT VOLSER CHPID=PATH STATUS
RTYPE SSID CFW TC DFW PIN DC-STATE CCA DDC ALT CU-TYPE
25C1,33903 ,0,000,MHLV11,7A=+ 7B=+ 7E=+ 7F=+
339038 00FF Y YY. YY. N SIMPLEX C1 01 3990-3
25C2,33903 ,0,000,MHLV12,7A=+ 7B=+ 7E=+ 7F=+
339038 00FF Y YY. YY. N SIMPLEX C2 02 3990-3
25C3,33903 ,0,000,MHLV13,7A=+ 7B=+ 7E=+ 7F=+
339038 00FF Y YY. YY. N SIMPLEX C3 03 3990-3
25C4,33903 ,0,000,MHLV14,7A=+ 7B=+ 7E=+ 7F=+
339038 00FF Y YY. YY. N SIMPLEX C4 04 3990-3
25C5,33903 ,0,000,MHLV15,7A=+ 7B=+ 7E=+ 7F=+
339038 00FF Y YY. YY. N SIMPLEX C5 05 3990-3
25C6,33903 ,0,000,TOTDCM,7A=+ 7B=+ 7E=+ 7F=+
339038 00FF Y YY. YY. N SIMPLEX C6 06 3990-3
25C7,33903 ,0,000,TOTDCN,7A=+ 7B=+ 7E=+ 7F=+
339038 00FF Y YY. YY. N SIMPLEX C7 07 3990-3
25C8,33903 ,0,000,TOTDCO,7A=+ 7B=+ 7E=+ 7F=+
339038 00FF Y YY. YY. N SIMPLEX C8 08 3990-3
25C9,33903 ,0,000,TOTDCP,7A=+ 7B=+ 7E=+ 7F=+
339038 00FF Y YY. YY. N SIMPLEX C9 09 3990-3
***** SYMBOL DEFINITIONS *****
0 = ONLINE + = PATH AVAILABLE
***** BOTTOM OF DATA *****

```

Figure 5-23 SMS operator commands

### 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, refer to *z/OS DFSMSdfp Storage Administration Reference*, 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:

- Changes options set on the IGDSMSxx PARMLIB member
- Restarts SMS if it has terminated
- Updates the SMS configuration

Table 5-1 on page 230 lists the differences between the **SETSMS** and **SET SMS** commands.

**VARY SMS** This command changes storage group, volume, library, or drive status. You can use this command to:

- Limit new allocations to a volume or storage group
- Enable a newly-installed volume for allocations

**DEVSERV**

This command displays information for a device. Use it to display the status of extended functions in operation for a given volume that is attached to a cache-capable 3990 storage control.

*Table 5-1 Differences between SETSMS and SET SMS commands*

<b>Difference</b>	<b>SET SMS=xx</b>	<b>SETSMS</b>
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, refer to *z/OS MVS System Commands*, SA22-7627.

## 5.24 Displaying the SMS configuration

```

-D SMS,SG(STRIPE),LISTVOL
IGD002I 11:31:41 DISPLAY SMS 823

STORGRP  TYPE      SYSTEM= 1 2 3 4
STRIPE   POOL      + + + +

VOLUME   UNIT      SYSTEM= 1 2 3 4      STORGRP NAME
MHLV11   25C1      + + + +            STRIPE
MHLV12   25C2      + + + +            STRIPE
MHLV13   25C3      + + + +            STRIPE
MHLV14   25C4      + + + +            STRIPE
MHLV15   25C5      + + + +            STRIPE
SBOX28   6312      + + + +            STRIPE
SBOX29   6412      + + + +            STRIPE
SBOX30   6512      + + + +            STRIPE
SBOX31   6013      + + + +            STRIPE
***** LEGEND *****
. THE STORAGE GROUP OR VOLUME IS NOT DEFINED TO THE SYSTEM
+ THE STORAGE GROUP OR VOLUME IS ENABLED
- THE STORAGE GROUP OR VOLUME IS DISABLED
* THE STORAGE GROUP OR VOLUME IS QUIESCED
D THE STORAGE GROUP OR VOLUME IS DISABLED FOR NEW ALLOCATIONS ONLY
Q THE STORAGE GROUP OR VOLUME IS QUIESCED FOR NEW ALLOCATIONS ONLY
> THE VOLSER IN UCB IS DIFFERENT FROM THE VOLSER IN CONFIGURATION
SYSTEM 1 = SC63      SYSTEM 2 = SC64      SYSTEM 3 = SC65
SYSTEM 4 = SC70
***** BOTTOM OF DATA *****

```

Figure 5-24 Display SMS configuration

### Displaying SMS configuration

You can display the SMS configuration in two ways:

- ▶ Using **ISMF Control Data Set**, enter **ACTIVE** in the **CDS Name** field and select **1 Display**.
- ▶ The **DISPLAY SMS:** operator command shows volumes, storage groups, libraries, drives, SMS configuration information, SMS trace parameters, SMS operational options, OAM information, OSMC information, and cache information. Enter this command to:
  - Confirm that the system-managed volume status is correct
  - Confirm that SMS starts with the proper parameters

## 5.25 Managing data with minimal SMS configuration

- Device-independence space allocation
- System-determined block size
- Use ISMF to manage volumes
- Use simplified JCL to allocate data sets
- Manage expiration date
- Establish installation standards and use data class ACS routine to enforce them
- Manage data set allocation
- Use PDSE data sets

Figure 5-25 Managing data with minimal SMS configuration

### Managing data allocation

After the SMS minimal configuration is active, your installation can exploit some SMS capabilities that give you experience with SMS and help you plan the DFSMS full exploitation with system-managed data set implementation.

Inefficient space usage and poor data allocation cause problems with space and performance management. In a DFSMS environment, you can enforce good allocation practices to help reduce some of these problems. The following section highlights how to exploit SMS capabilities.

### Using data class to standardize data allocation

This section describes how to use data class to establish standards for data allocation. For sample data classes, descriptions, and ACS routines, see *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407.

You can define data classes containing standard data set allocation attributes. Users then only need to use the appropriate data class names to create standardized data sets. To override values in the data class definition, they can still provide specific allocation parameters.

Data classes can be determined from the user-specified value on the DATACLAS parameter (DD card, TSO Alloc, Dynalloc macro), from a RACF default, or by ACS routines. ACS



routines can also override user-specified or RACF default data classes, as shown in Figure 5-26 on page 234.

However, you can override a data class attribute (not the data class itself) using JCL or dynamic allocation parameters. However, overriding a subparameter of a parameter overrides *all* of the subparameters for that parameter. For example, SPACE=(TRK,(1)) in JCL will cause primary, secondary, and directory quantities, as well as AVGREC and AVGUNIT, in the data class to be overridden.

DFSMS usually does not change values that are explicitly specified, because doing so would alter the original meaning and intent of the allocation. There is an exception, however—if it is clear that a PDS is being allocated (DSORG=PO or DSNTYPE=PDS is specified), and no directory space is indicated in the JCL—then the directory space from the data class is used even though SPACE=(TRK,(1)) was specified.

Users cannot override the data class attributes of dynamically-allocated data sets if you use the IEFDB401 user exit.

Data classes can also be determined for objects by a specification using the **SETOAM** command in the CBROAMxx member of SYS1.PARMLIB.

## 5.26 Device-independence space allocation

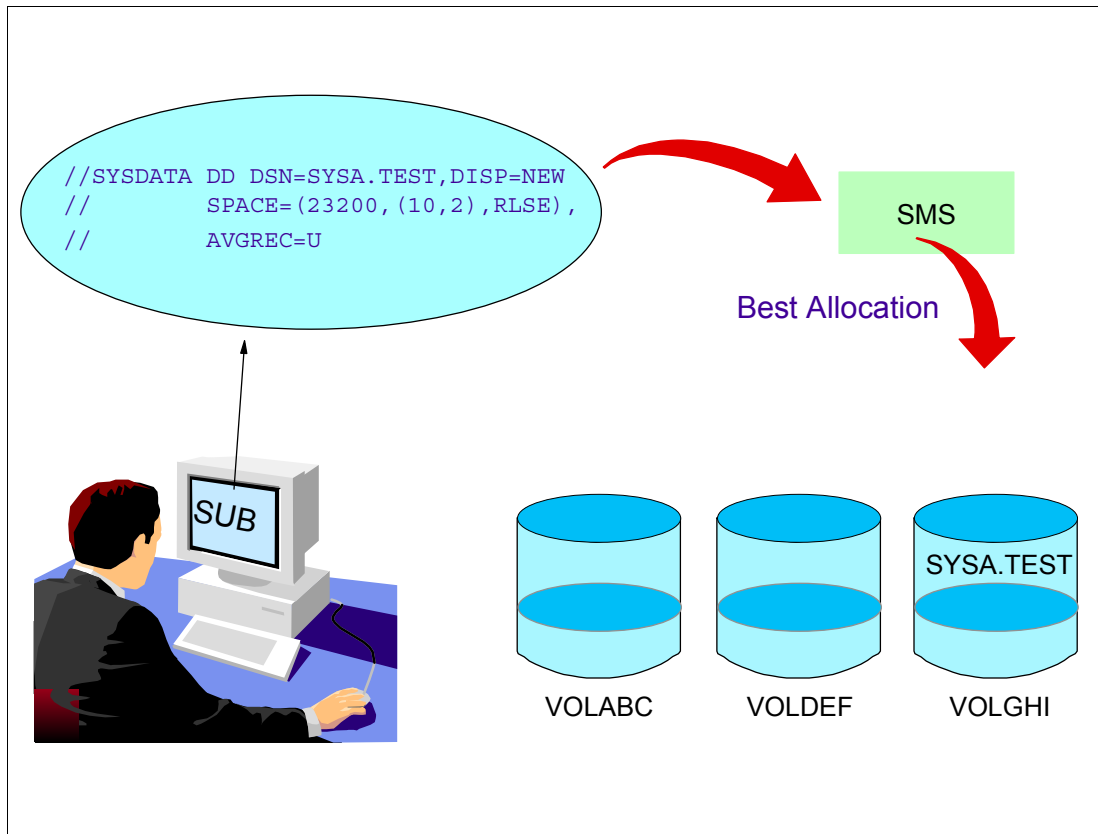


Figure 5-26 Device independence

### Ensuring device independence

The base configuration contains a default unit that corresponds to a DASD esoteric (such as SYSDA). Default geometry for this unit is specified in bytes/track and tracks/cylinder for the predominant device type in the esoteric. If users specify the esoteric, or do not supply the UNIT parameter for new allocations, the default geometry converts 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 default geometry.

### System-determined block size

During allocation, DFSMSdfp assists you to assign a block size that is optimal for the device. When you allow DFSMSdfp to calculate the block size for the data set, you are using a system-determined block size. System-determined block sizes can be calculated for system-managed and non-system-managed primary storage, VIO, and tape data sets.

The use of system-determined block size provides:

- ▶ Device independence, since you do not need to know the track capacity to allocate efficiently
- ▶ Space usage optimization
- ▶ I/O performance improvement
- ▶ Simplifies JCL, since you do not need to code BLKSIZE

You take *full advantage* of system-managed storage when you allow the system to place data on the most appropriate device in the most efficient way, as shown in Figure 5-26, when you

use *system-managed data sets*. In the DFSMS environment, you control volume selection through the storage class and storage group definitions you create, and by ACS routines. This means that users do not have to specify volume serial numbers with the VOL=SER parameter, or code a specific device type with the UNIT= parameter on their JCL.

When converting data sets for use in DFSMS, they do not have to remove these parameters from existing JCL because volume and unit information can be ignored with ACS routines. (However, you should work with users to evaluate UNIT and VOL=SER dependencies before conversion.)

If you keep the VOL=SER parameter for a non-SMS volume, but you are trying to access a system-managed data set, then SMS might not find the data set. All SMS data sets (the ones with a storage class) must reside in a *system-managed volume*.

## 5.27 Developing naming conventions

Setting the high-level qualifier standard		
First character	Second character	Remaining characters
Type of user	Type of data	Project name, code, or userid
A - Accounting Support D - Documentation E - Engineering F - Field Support M - Marketing Support P - Programming \$ - TSO userid	P - Production data D - Development data T - Test data M - Master data U - Update data W - Work data	Example:  3000 = Project code

Figure 5-27 Setting data set HLQ conventions

### Developing a data set naming convention

Whenever you allocate a new data set, you (or the operating system) must give the data set a unique name. Usually, the data set name is given as the dsname in 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 DEPT58.SMITH.DATA3 is composed of three name segments. The first name on the left is called the *high-level qualifier* (HLQ). The last name is the *low-level qualifier* (LLQ).

You must implement a naming convention for your data sets. Although naming convention is not a prerequisite for DFSMS conversion, it makes more efficient use of DFSMS. You can also reduce the cost of storage management significantly by grouping data that shares common management requirements. Naming conventions are an effective way of grouping data. They also:

- ▶ Simplify service-level assignments to data
- ▶ Facilitate writing and maintaining ACS routines
- ▶ Allow data to be mixed in a system-managed environment while retaining separate management criteria
- ▶ Provide a filtering technique useful with many storage management products
- ▶ Simplify the data definition step of aggregate backup and recovery support

Most naming conventions are based on the HLQ and LLQ of the data name. Other levels of qualifiers can be used to identify generation data sets and database data. They can also be used to help users to identify their own data.

## Using a high-level qualifier (HLQ)

Use the HLQ to:

- ▶ Identify the owner or owning group of the data or
- ▶ Indicate data type

Do not embed information that is subject to frequent change in the HLQ, such as department number, application location, output device type, job name, or access method. Set a standard within the HLQ. Figure 5-27 on page 236 shows examples of naming standards.

## 5.28 Setting the low-level qualifier (LLQ) standards

Low-Level Qualifier	Exp Days Non-Usage	Max Ret Period	Partial Release	Migrate Days Non-Usage	Cmd/Auto Migrate	No.GDG Primary	Backup Freqcy	Backup Versns	Retain Days Only BUP	Retain Day Extra BUP
ASM.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
CLIST....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
COB*.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
CNTL.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
DATA.....	400	400	YES	15	BOTH	--	2	2	400	60
*DATA....	400	400	YES	151	BOTH	--	2	2	400	60
FOR*.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
INCL*....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
INPUT....	400	400	YES	15	BOTH	--	2	2	1100	120
ISPROF...	400	400	YES	30	BOTH	--	0	2	60	30
JCL.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
LIST*....	2	2	YES	NONE	NONE	--	NONE	NONE	--	--
*LIST....	2	2	YES	NONE	NONE	--	NONE	NONE	--	--
LOAD*....	400	400	YES	15	BOTH	--	1	2	--	--
MACLIB...	400	400	YES	15	BOTH	--	1	2	400	60
MISC.....	400	400	YES	15	BOTH	--	2	2	400	60
NAMES....	.NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120
OBJ*.....	180	180	YES	7	BOTH	--	3	1	180	30
PLI.....	NOLIM	NOLIM	YES	15	BOTH	--	0	5	1100	120

Figure 5-28 Setting the LLQ standards

### Setting the low-level qualifier standards

The LLQ determines the contents and storage management processing of the data. You can use LLQs to identify data requirements for:

- ▶ Migration (data sets only)
- ▶ Backup (data sets and objects)
- ▶ Archiving (data sets)
- ▶ Retention or expiration (data sets and objects)
- ▶ Class transitions (objects only)
- ▶ Release of unused space (data sets only)

The retention and expiration of objects on tape volumes are determined on two levels. Tape volumes containing objects have a tape data set expiration date and an expiration date of when the last object on the tape is going to expire. For information on deleting expired objects on tape, see *z/OS DFSMSHsm Storage Administration Guide*, SC35-0421.

Mapping storage management requirements to data names is especially useful in a system-managed environment. In an environment without storage groups, data with differing requirements is often segregated onto separate volumes that are monitored and managed manually. LLQ data naming conventions allow data to be mixed together in a system-managed environment and still retain the separate management criteria.

Figure 5-28 shows examples of how you can use LLQ naming standards to indicate the storage management processing criteria.

The first column lists the LLQ of a data name. An asterisk indicates where a partial qualifier can be used. For example, LIST\* indicates that only the first four characters of the LLQ must be LIST; valid qualifiers include LIST1, LISTING, and LISTOUT. The remaining columns show the storage management processing information for the data listed.

## 5.29 Establishing installation standards

- ❑ Based on user needs
- ❑ Improve service to users
- ❑ Better transition to SMS-managed storage
- ❑ Use service-level agreement

*Figure 5-29 Establishing installation standards*

### **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.

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. Document negotiated policies in a service level agreement.

You can simplify storage management by limiting the number of data sets and volumes that cannot be system-managed.



## 5.30 Planning and defining data classes

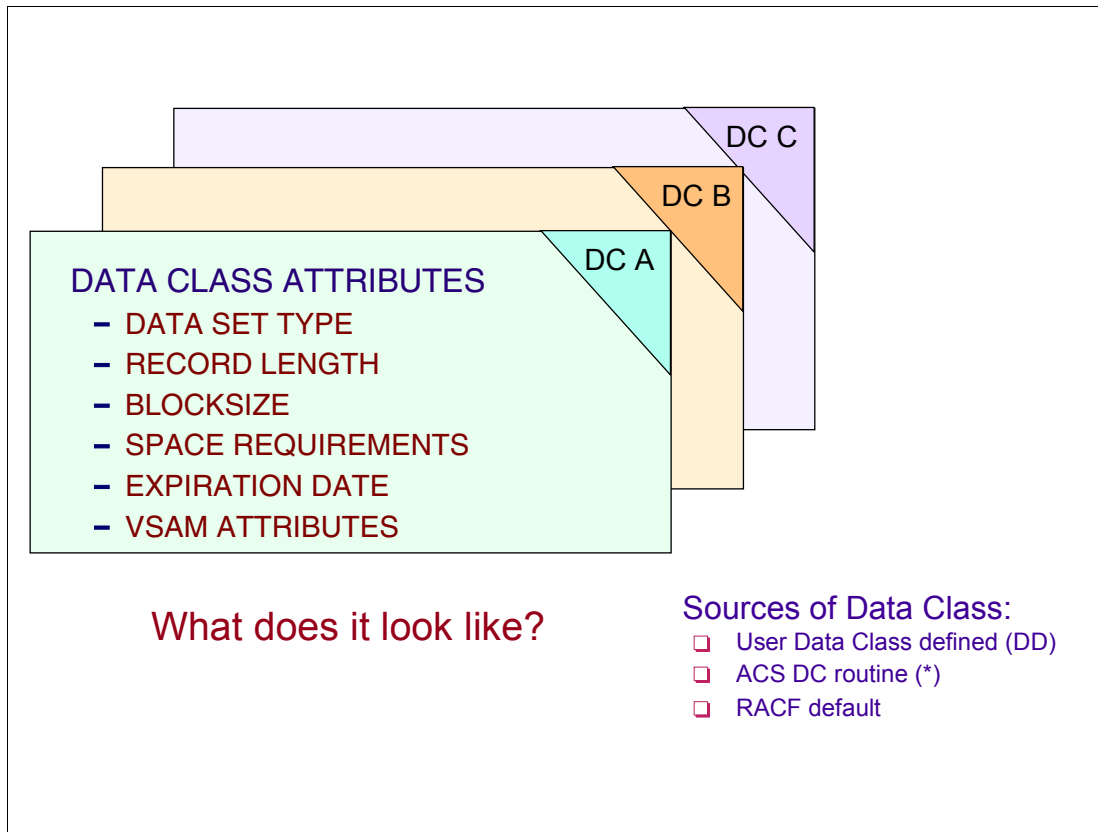


Figure 5-30 Planning and defining data class

### Planning and defining data class

After you establish your installation's standards, use your service level agreement (SLA) for reference when planning your data classes. SLAs identify users current allocation practices and their requirements. For example:

- ▶ Based on user requirements, you might create a data class to allocate standard control libraries.
- ▶ You can create a data class to supply the default value of a parameter, so users do not have to specify a value for that parameter in the JCL or dynamic allocation.

Data class names should indicate the type of data they are assigned to. This makes it easier for users to identify the template they need to use for allocation.

You define data classes using the ISMF data class application. Users can access the Data Class List panel to determine which data classes are available and the allocation values that each data class contains.

Figure 5-31 on page 242 contains information that can help in this task. For more information on planning and defining data classes, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

## 5.31 Data class attributes

- ❑ Data class name and data class description (DC)
- ❑ Data set organization (RECORG) and data set name type (DSNTYPE)
- ❑ Record format (RECFM) and logical record length (LRECL)
- ❑ Key length (KEYLEN) and offset (KEYOFF)
- ❑ Space attributes (AVGREC, AVE VALUE, PRIMARY, SECONDARY, DIRECTORY)
- ❑ Retention period or expiration date (RETPD or EXPDT)
- ❑ Number of volumes the data set can span (VOLUME COUNT)
- ❑ Allocation amount when extending VSAM extended data set
- ❑ VSAM index options (IMBED or REPLICATE)
- ❑ Control interval size for VSAM data components (CISIZE DATA)
- ❑ Percentage of control interval or control area free space (% FREESPACE)
- ❑ VSAM share options (SHAREOPTIONS)
- ❑ Compaction option for data sets (COMPACTION)
- ❑ Tape media (MEDIA TYPE)

Figure 5-31 Data class attributes

### Data class attributes

You can specify the data class space attributes to control DASD space waste. For example:

- ▶ The primary space value should 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 COMPACTION attribute specifies whether data is to be compressed on DASD if the data set is allocated in the extended format. The COMPACTION 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 COMPACTION attribute, the data set *must* be system-managed, since 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 on specifying data class attributes, see *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402.

## 5.32 Use data class ACS routine to enforce standards

### Examples of standards to be enforced:

- Prevent extended retention or expiration periods
- Prevent specific volume allocations, unless authorized
- You can control allocations to spare, system, database, or other volumes
- Require valid naming conventions for permanent data sets

Figure 5-32 Using data class (DC) ACS routine to enforce standards

### Using data class (DC) ACS routine to enforce standards

Once you started DFSMS with the minimal configuration, you can use data class ACS routine facilities to automate or simplify storage allocation standards *if you*:

- ▶ Use manual techniques to enforce standards
- ▶ Plan to enforce standards before implementing DFSMS
- ▶ Use DFSMSdfp or MVS installation exits to enforce storage allocation standards

The data class ACS routine provides an automatic method for enforcing standards, because it is called *for system-managed and non-system-managed 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

## 5.33 Simplifying JCL use

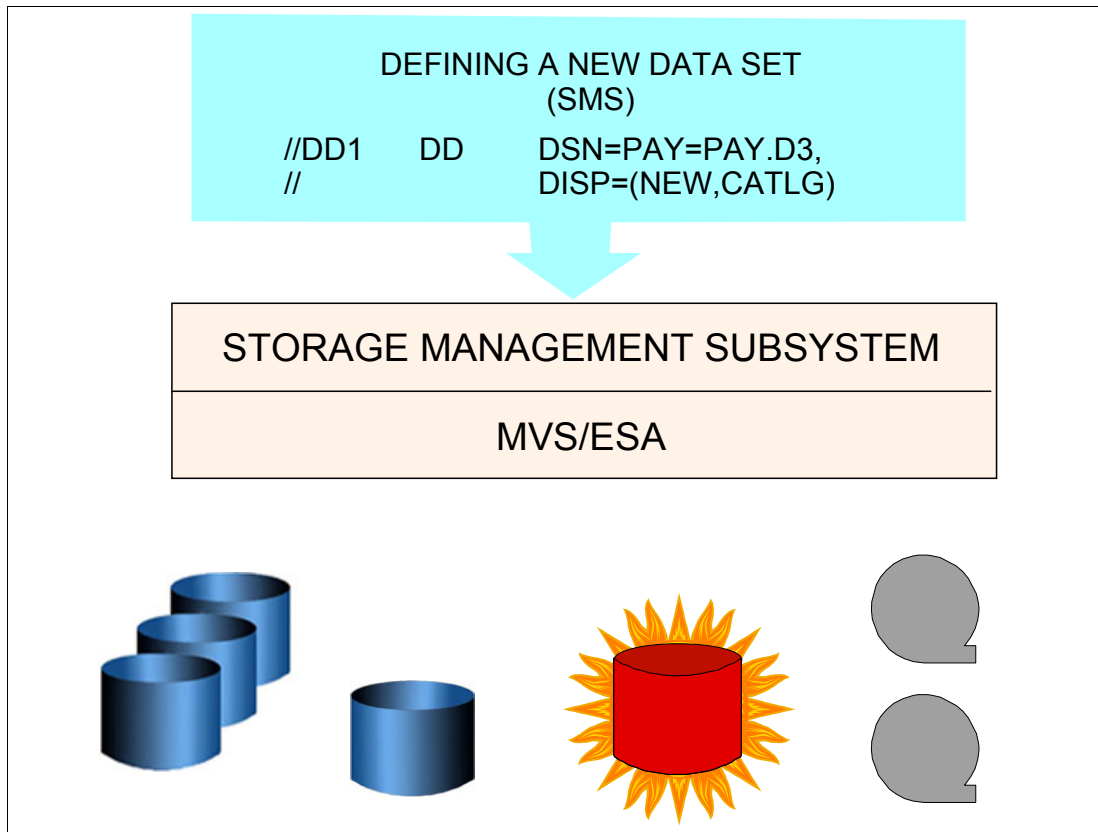


Figure 5-33 Using SMS capabilities to simplify JCL

### Use simplified JCL

Once you defined and start using data classes, several JCL keywords can help you simplify the task of creating data sets and also to make the allocation process more consistent. It is also possible to allocate VSAM data sets through JCL without IDCAMS assistance.

For example, with the use of data classes, you have less use for the JCL keywords: UNIT, DCB, and AMP. When you start using *system-managed data sets*, you do not need to use the JCL VOL keyword.

### JCL keywords used in the DFSMS environment

You can use JCL keywords to create VSAM and non-VSAM data sets. For a detailed description of the keywords and their use, see *z/OS MVS JCL User's Guide*, SA22-7598.

In the following pages, we present some sample jobs exemplifying the use of JCL keywords when:

- ▶ Creating a sequential data set
- ▶ Creating a VSAM cluster
- ▶ Specifying a retention period
- ▶ Specifying an expiration date

## 5.34 Allocating a data set

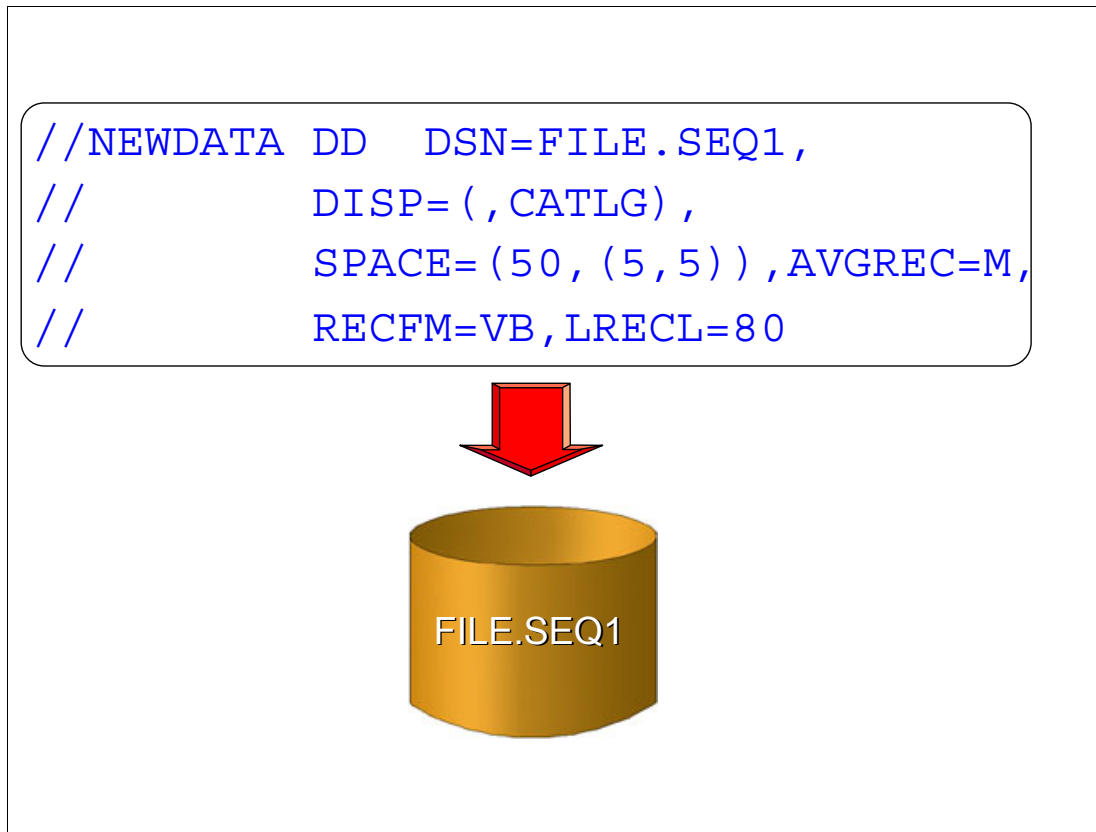


Figure 5-34 Allocating a sequential data set

### Creating and allocating data sets

Many times the words *create* and *allocate*, when applied to data sets, are used in MVS as synonyms. However, they are not.

- ▶ To *create* (in DASD) means to assign a space in VTOC to be used for a data set (sometimes *create* implies cataloging the data set). A data set is created in response to the DD card `DISP=NEW` in JCL.
- ▶ To *allocate* means to establish a logical relationship between the request for the use of the data set within the program (through the use of a DCB or ACB) and the data set itself in the device where it is located. Being more specific, allocation implies finding where the data set is (for an already existent data set) or where it will be (for a new one). Thinking in control block terms, the DCB/ACB is connected to the DD card through the DDNAME field. The DD card content forms a TIOT entry and at allocation time this entry points to the UCB where the data set exists.

Figure 5-34 shows an example of JCL used to create a data set in a system-managed environment.

These are some characteristics of the JCL in a system-managed environment:

- ▶ The LRECL and RECFM parameters are independent keywords. This makes it easier to override individual attributes that are assigned default values by the data class, selected by the ACS routines, that might not be appropriate for the data set being allocated.

- ▶ In this example, the SPACE parameter is coded with the average number of bytes per record (50), and the number of records required for the primary data set allocation (5 M) and secondary data set allocation (5 M). These are the values that the system uses to calculate the least number of tracks required for the space allocation. This also eliminates the need for device awareness, replacing the TRK or CYL unit specification. For variable-block data sets, the average number of bytes per record is not necessarily the same as the LRECL value. In the example, the average record length is 50, whereas the LRECL is 80.

Note that if you code the SPACE parameter on a DD statement that defines an existing data set, the SPACE value you specify temporarily overrides the SPACE value used to create the data set.

- ▶ The AVGREC attribute indicates the scale factor for the primary and secondary allocation values. In the example, an AVGREC value of M indicates that the primary and secondary values of 5 are each to be multiplied by 1 048 576.

The SPACE parameter would result in a primary allocation of 5 MB and a secondary allocation of 5 MB.

- ▶ For *system-managed data sets*, the device-dependent volume serial number and unit information is no longer required, because the volume is assigned within a storage group selected by the ACS routines. This eliminates the need for device awareness.

### Overriding data class attributes with JCL

In a DFSMS environment, the JCL to allocate a data set is simpler and has no device-dependent keywords. The data class can be:

- ▶ Specified in the DATACLAS parameter of the JCL DD statement
- ▶ Automatically assigned by data class ACS routine
- ▶ Set in the user RACF profile

Table 5-2 lists the attributes a user can override with JCL.

Table 5-2 Data class attributes that can be overridden by JCL

JCL DD statement keyword	Use for
RECORD,KEYLEN,KEYOFF	Only VSAM
RECFM	Sequential (PO or PS)
LRECL,SPACE,AVGREC, RETPD or EXPDT, VOLUME (volume count)	All data set types
DSNTYPE	PDS or PDSE

As previously mentioned, in order to use a data class, the data set does *not* have to be system-managed. An installation can take advantages of a minimal SMS configuration to simplify JCL use and manage data set allocation.

For information on managing data allocation, refer to *z/OS DFSMS Using Data Sets*, SC26-7410.

## 5.35 Creating a VSAM cluster

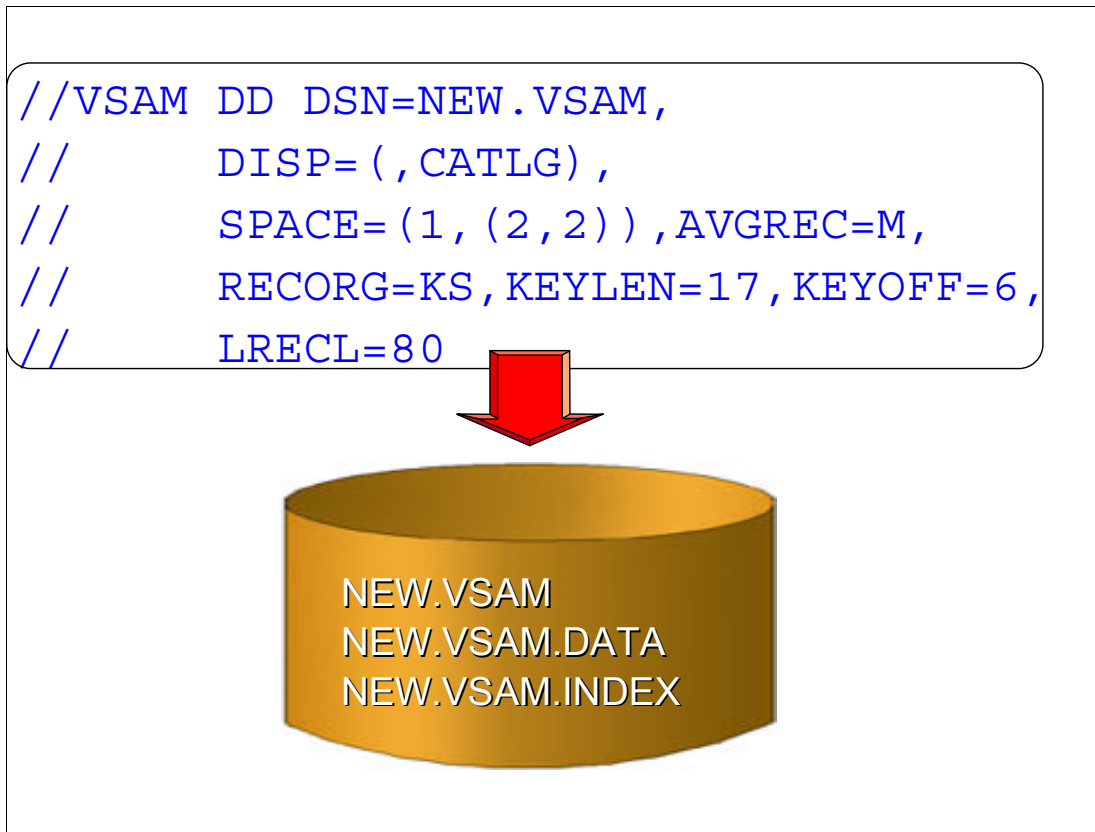


Figure 5-35 Creating a VSAM data class

### Creating a VSAM data set using JCL

In the DFSMS environment, you can create temporary and permanent VSAM data sets using JCL by using either of the following:

- ▶ The RECOrg parameter of the JCL DD statement
- ▶ A data class

You can use JCL DD statements parameters to override some data class attributes; refer to Table 5-2 on page 246 for those related to VSAM data sets.

**Attention:** Regarding DISP=(OLD,DELETE), in an SMS environment, the VSAM data set is deleted at unallocation. In a non-SMS environment, the VSAM data set is kept.

A data set with a disposition of MOD is treated as a NEW allocation if it does not already exist; otherwise, it is treated as an OLD allocation.

For a non-SMS environment, a VSAM cluster creation is only done through IDCAMS. In Figure 5-35, NEW.VSAM refers to a KSDS VSAM cluster.

## 5.36 Space allocation for a VSAM KSDS cluster

- Allocation specified at the cluster or alternate index level
- Allocation specified at the data level
- Allocation specified at both the data and index levels
- Secondary allocation specified at the data level

*Figure 5-36 Allocating VSAM data sets in JCL*

### **Considerations when specifying space for a KSDS**

The space allocation for a VSAM entity depends on the level of the entity being allocated:

- ▶ If allocation is specified at the cluster or alternate index level only, the amount needed for the index is subtracted from the specified amount. The remainder of the specified amount is assigned to data.
- ▶ If allocation is specified at the data level only, the specified amount is assigned to data. The amount needed for the index is in addition to the specified amount.
- ▶ If allocation is specified at both the data and index levels, the specified data amount is assigned to data and the specified index amount is assigned to the index.
- ▶ If secondary allocation is specified at the data level, secondary allocation must be specified at the index level or the cluster level.

You cannot use certain parameters in JCL when allocating VSAM data sets, although you can use them in the IDCAMS **DEFINE** command.



## 5.37 Retention period and expiration date

```
//RETAIN DD DSN=DEPTM86.RETPD.DATA,  
//          DISP=(,CATLG),RETPD=365
```

```
//RETAIN DD DSN=DEPTM86.EXPDT.DATA,  
// DISP=(,CATLG),EXPDT=99364
```

Figure 5-37 Retention period and expiration date

### Managing retention period and expiration date

The RETPD and EXPDT parameters specify retention period and expiration date. They apply alike to system-managed and non-system-managed data sets. They control the time during which a data set is protected from being deleted by the system. The first DD statement in Figure 5-37 protects the data set from deletion for 365 days. The second DD statement in Figure 5-37 protects the data set from deletion until December 30, 1999.

The VTOC entry for non-VSAM and VSAM data sets contains the expiration date as declared in the JCL, the TSO **ALLOCATE** command, or the IDCAMS **DEFINE** command, or can also come from the data class definition. The expiration date is placed in the VTOC either directly from the date specification, or after it is calculated from the retention period specification. The expiration date in the catalog entry exists for information purposes only. If you specify the current date or an earlier date, the data set is immediately eligible for replacement.

You can use *management class* to limit or ignore the RETPD and EXPDT parameters given by a user. If a user specifies values that exceed the maximum allowed by the management class definition, the retention period is reset to the allowed maximum. For an expiration date beyond year 1999 use the following format: YYYY/DDD. For more information on using management class to control retention period and expiration date, refer to *z/OS DFSMSHsm Storage Administration Guide*, SC35-0421.

**Attention:** EXPDT=99365, or 99366, or 1999/365 or 1999/3666 are special dates and they mean *never expires*.

## 5.38 SMS PDSE support

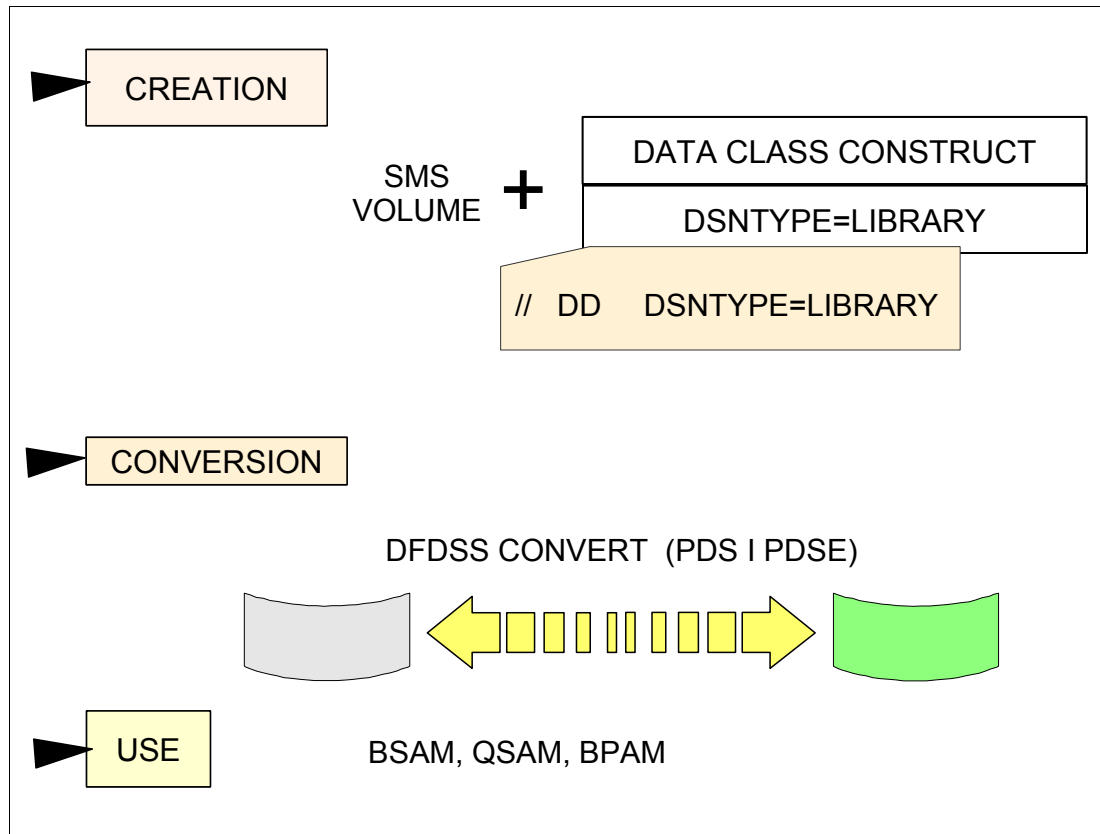


Figure 5-38 SMS PDSE support

### SMS PDSE support

Partitioned data set extended (PDSE) is a type of data set organization that improves the partition data set (PDS) organization. It has an improved indexed directory structure and a different member format.

With the minimal SMS configuration, you can exploit the use of PDSE data sets. A PDSE *does not have to be system-managed*. You can use them for source (programs and text) libraries, macros, and program object libraries. PDSE advantages, when compared with PDS, are:

- ▶ The size of a PDSE directory is flexible and can expand to accommodate the number of members stored in it (the size of a PDS directory is fixed at allocation time).
- ▶ PDSE members are indexed in the directory by member name. This eliminates the need for time-consuming sequential directory searches.
- ▶ The logical requirements of the data stored in a PDSE are separated from the physical (storage) requirements of that data, which simplifies data set allocation.
- ▶ PDSEs provide more efficient use of DASD space. For example, by moving or deleting a PDSE member, you free space that is immediately available for the allocation of a new member, without first having to compress the data set to consolidate the fragmented space for reuse. This makes PDSEs less susceptible to space-related abends than PDSs.
- ▶ The number of PDSE members stored in the library can be large or small without concern for performance or space considerations.

- ▶ The ability to update a member in place is possible with PDSs and PDSEs. But with PDSEs, you can extend the size of members and the integrity of the library is maintained while simultaneous changes are made to separate members within the library.
- ▶ The maximum number of extents of a PDSE is 123; the PDS is limited to 16.
- ▶ PDSEs are device-independent because they do not contain information that depends on location or device geometry.
- ▶ All members of a PDSE are reblockable.
- ▶ PDSEs can contain program objects built by the program management binder that cannot be stored in PDSs.

You can also share PDSEs within and across systems. With systems that support PDSEs (MVS/DFP 3.2.0 or higher level), multiple users are allowed to read PDSE members while the data set is open.

If you have DFSMS installed, you can extend the sharing to enable multiple users on multiple systems to concurrently create new PDSE members and read existing members.

Using the PDSESHARING keyword in the SYS1.PARMLIB member, IGDSMSxx, you can specify:

- ▶ **NORMAL.** This allows multiple users to read any member of a PDSE.
- ▶ **EXTENDED.** This allows multiple users to read any member or create new members of a PDSE

All systems sharing PDSEs need to be upgraded to DFSMS to use the extended PDSE sharing capability.

After updating the IGDSMSxx member of SYS1.PARMLIB, you need to issue the **SET SMS ID=xx** command for every system in the complex to activate the sharing capability.

For additional information on PDSEs, see *z/OS DFSMS Using Data Sets*, SC26-7410.

Although SMS supports PDSs, you should consider converting these to the PDSE format. The following sections describe this process.

## 5.39 PDSE conversion

Using DFSMSdss	Using IEBCOPY
<pre>COPY DATASET (INCLUDE               (MYTEST.** )               BY (DSORG = PDS) )               INDY (SMS001)               OUTDY (SMS002)               CONVERT (PDSE (**))               RENAMEU (MYTEST2)               DELETE</pre>	<pre>//INPDS DD DISP=SHR,           DSN=USER.PDS.LIBRARY //PDSE  DD DSN=USER.PDSE.LIBRARY,           DISP=OLD //SYSIN DD *           COPY OUTDD=OUTPDSE           INDD=INPDS           SELECT MEMBER=(A,B,C)</pre>

Figure 5-39 Converting PDS to PDSE

### Converting a PDS data set to a PDSE

You can use **IEBCOPY** or **DFSMSdss COPY** to convert partitioned data sets to PDSEs, as shown in Figure 5-39. We recommend using **DFSMSdss**.

You can convert the entire data set or individual members, and also back up and restore PDSEs. By using the **DFSMSdss COPY** function with the **CONVERT** and **PDS** keywords, you can convert a PDSE back to a PDS. This is especially useful if you need to prepare a PDSE for migration to a site that does not support PDSEs. When copying members from a partitioned data set load module library into a PDSE program library, or vice versa, the system invokes the program management binder.

Many types of libraries are candidates for conversion to PDSE:

- ▶ PDSs that are updated often, and that require frequent and regular reorganization
- ▶ Large PDSs that require specific device types because of the size of allocation

Converting PDSs to PDSEs is beneficial, but be aware that certain data sets are unsuitable for conversion to, or allocation as, PDSEs because the system does not retain the original block boundaries. Also, data sets requiring device dependency are inappropriate to convert or allocate because PDSEs are device-independent.

To reclaim unused space in those data sets that cannot be converted, use the **DFSMSdss COMPRESS** command to compress PDSs in place. This consolidates space that is no longer used within a PDS and makes it available at the end of the data set. For large or critical

PDSs, you might want to copy or back up the data sets before you compress them. This maintains data set availability should the compress fail. For more information on DFSMSdss, see *z/OS DFSMSdss Storage Administration Guide*, SC35-0423, and *z/OS DFSMSdss Storage Administration Reference*, SC35-0424.

## 5.40 DFSMS and program objects

- ❑ Functions to create, update, execute, and access program objects in PDSEs
- ❑ New load module format
- ❑ New Linkage Editor, the binder
- ❑ New program fetch
- ❑ DESERV internal interface function AMASPZAP
- ❑ Set of utilities such as IEWTPORT, which builds transportable programs from program objects and vice versa
- ❑ Coexistence between PDS and PDSE load module libraries in same system

Figure 5-40 DFSMS and program objects

### Problems for load modules in PDS

Load modules stored in a PDS present some constraints, such as:

- ▶ Maximum size for TXT is 16 MB.
- ▶ Maximum number of CESDs is 32 K.
- ▶ The PDS restrictions, such as:
  - It needs compression (IEBCOPY)
  - Unexpandable directory size
  - High directory search connect time (LLA and DASD cache relief)
  - Directory must be rewritten, when a member is added (sorted by collating sequence)
  - Authorization and serialization at data-set level only
  - Concurrent member creation by different tasks is an integrity exposure
- ▶ Software programs such as service aids and utilities must know the internal structures of module and directory entries.

The binder converts the output of language translators and compilers into an executable program unit that can either be read directly into virtual storage for execution or stored in a program library.

Most of the loading functions are transparent to the user. The loader knows whether the program being loaded is a load module or a program object by the source data set type:

- ▶ If the program is being loaded from a PDS, it calls IEWFETCH (now integrated as part of the loader) to do what it has always done.
- ▶ If the program is being loaded from a PDSE, a new routine is called to bring in the program using DIV. The loading is done using special loading techniques that can be influenced by externalized options.

A second directory service in support of PDSE directories, DESERV, was externalized in DFSMS 1.3. You may issue DESERV for either PDS or PDSE directory access, but you must pass the DCB address. It does not default to a predefined search order, as does BLDL. (Both BLDL and DESERV support "bypass-LLA.") DESERV returns an SMDE which, for PDSE directories, contains more information than is mapped by IHAPDS.

You create a transportable copy of the program object using IEWTPORT, then send the transportable copy to the system without program management services. A program on the target system can access the transportable copy using QSAM. If you want to load, bind, or execute a transportable program, you must first recreate the program object by executing IEWTPORT on a system with program management services installed. No programming interfaces exist to perform any of these operations on transportable programs. IEWTPORT does not support load modules, nor does it support program objects in overlay format.

## Program objects

Program objects are a new format of load modules. In this format, load modules are called program objects. This format is only allowed when stored in a PDSE program object library. A program object consists of *text* (executable code and data areas), *information* about load (relocating address constants) and *binding* (solve external references) in text. The format and content of the object program and directory entry are not externalized (encapsulation).

The constraints removed from program objects are:

- ▶ Module size up to 2 Gb (TXT up to 1 Gb)
- ▶ Virtually unlimited number of aliases and external names

DFSMS has:

- ▶ Functions to create, update, execute, and access load modules (program objects) in PDSEs.
- ▶ A new load module format named the program object library.
- ▶ A new linkage editor, called the binder.
- ▶ A new program fetch, called the loader, and five new load modes. For fetching load modules, IEWFETCH is invoked by the loader.
- ▶ The DESERV internal interface function, to access, add, or replace directories entries in a program library (PDS or PDSE), used by:
  - Binder
  - Loader
  - LLA
  - AMASPZAP
- ▶ A set of utilities including:
  - IEWTPORT, which builds transportable programs from program objects, and vice versa.
- ▶ Coexistence between PDS and PDSE load module libraries in the same system.

Binder (DFSMS) replaces the linkage editor and loader. It executes all functions of load module linkage and editing done by the linkage editor/loader. It supports the new PDSE load module format program object, and also supports the old PDS load module format.

The program management loader is the MVS support to load program objects from PDSEs:

- ▶ Relocates all the address constants in the program to point to the appropriated areas in VS.
- ▶ Supports 24-bit or 31-bit addressing.
- ▶ Program objects may have different load modes, based on the module characteristics and parameters specified to the binder when the object program was created (FETCHOPT). Among these load modes are the following:
  - Relocate and pre-load in virtual storage before execution
  - Relocate and load into real storage (with a virtual storage address) for execution as a result of a page fault)



## 5.41 Selecting data sets to allocate as PDSEs

### The &DSNTYPE ACS read-only variable controls the allocation:

- ❑ &DSNTYPE = 'LIBRARY' for PDSEs.
- ❑ &DSNTYPE = 'PDS' for PDSs.
- ❑ &DSNTYPE is not specified
  - This indicates that the allocation request is provided by the user through JCL, the TSO/E ALLOCATE command, or dynamic allocation.

Figure 5-41 Selecting a data set to allocate as PDSE

### Selecting a data set to allocate as PDSE

As a storage administrator, you can code appropriate ACS routines to select data sets to allocate as PDSEs and prevent inappropriate PDSs from being allocated or converted to PDSEs.

By using the &DSNTYPE read-only variable in the ACS routine for data-class selection, you can control which PDSs are to be allocated as PDSEs. The following values are valid for DSNTYPE in the data class ACS routines:

```
&DSNTYPE = 'LIBRARY' for PDSEs.  
&DSNTYPE = 'PDS' for PDSs.  
&DSNTYPE is not specified. This indicates that the allocation request  
is provided by the user through JCL, the TSO/E ALLOCATE command, or  
dynamic allocation.
```

If you specify a DSNTYPE value in the JCL, and a different DSNTYPE value is also specified in the data class selected by ACS routines for the allocation, the value specified in the data class is ignored.

## 5.42 Allocating new PDSEs

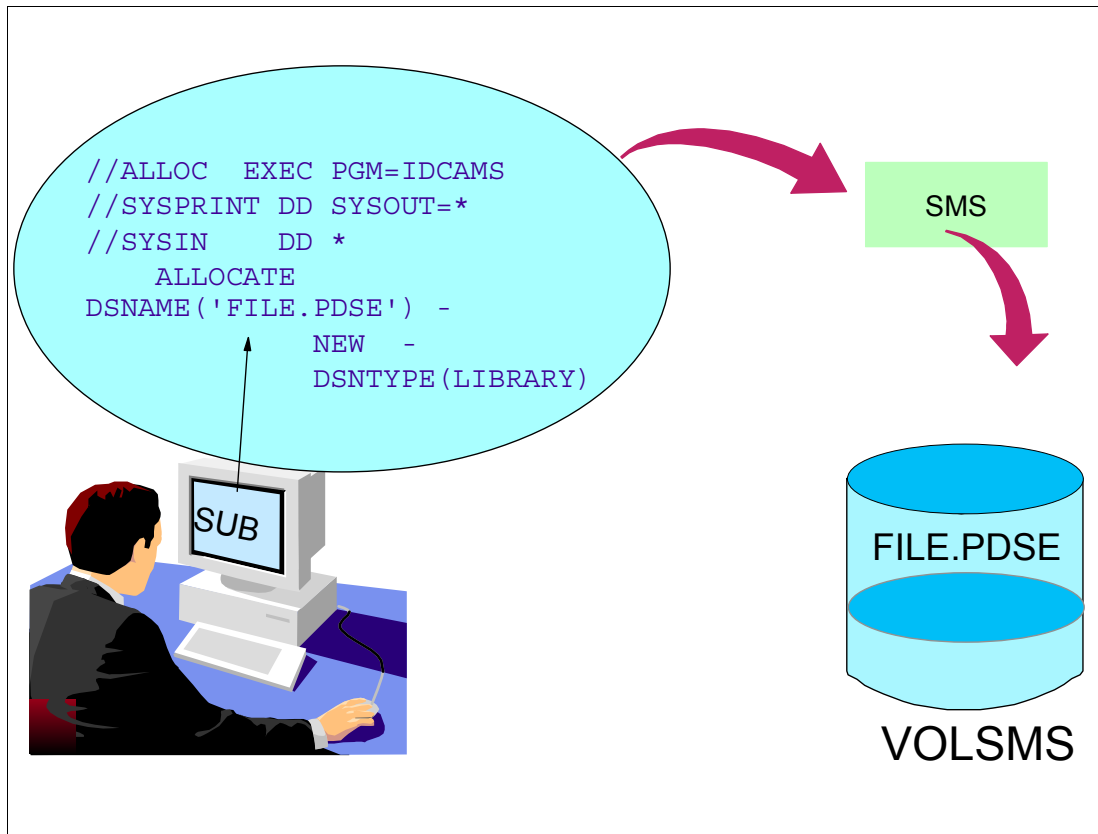


Figure 5-42 Allocating a PDSE data set

### Allocating new PDSEs

You can allocate PDSEs only in an SMS-managed environment. The PDSE data set does not have to be system-managed. To create a PDSE, use:

- ▶ DSNTYPE keyword in the JCL, TSO or IDCAMS **ALLOCATE** command
- ▶ A data class with LIBRARY in the Data Set Name Type field

You use DSNTYPE(LIBRARY) to allocate a PDSE, or DSNTYPE(PDS) to allocate a PDS. Figure 5-42 shows IDCAMS **ALLOCATE** used with the DSNTYPE(LIBRARY) keyword to allocate a PDSE.

A PDS and a PDSE can be concatenated in JCL DD statements, or by using dynamic allocation, such as the TSO **ALLOCATE** command.

## 5.43 Identifying PDSEs

```

Using PDF, option 3.4:
DSLISL - Data Sets Matching MHLRES2
Command ==> █
Row 92 of 96
Scroll ==> CSR

Command - Enter "/" to select action
-----
MHLRES2.TESTS.JCL          PO    FB      80  27920
MHLRES2.TESTS.SYSOUTS     PO-E  FBA     133  32718
MHLRES2.TST.GTFNEW        PO    U        0  13030
MHLRES2.TXCOMP.TEXT       PS    FB      80  6160
MHLRES2.XMIT.GTF          PS    FB      80  3120
***** End of Data Set list *****

Using ISMF, option 1:
DATA SET LIST
Command ==>
Enter Line Operators below:
Entries 89-96 of 96
Data Columns 30-32 of 39

LINE OPERATOR  DATA SET NAME  DATA SET NAME TYPE  NUM OF STRIPES  ENTRY TYPE
--- (1) ---    --- (2) ---      --- (30) ---        --- (31) ---    --- (32) ---
MHLRES2.SMS.LIST  OTHERS          --                --              NONVSAM
MHLRES2.SRCHFOR.LIST  OTHERS          --                --              NONVSAM
MHLRES2.SUPERC.LIST  OTHERS          --                --              NONVSAM
MHLRES2.TESTS.JCL   OTHERS          --                --              NONVSAM
MHLRES2.TESTS.SYSOUTS  LIBRARY         --                --              NONVSAM
MHLRES2.TST.GTFNEW  OTHERS          --                --              NONVSAM
MHLRES2.TXCOMP.TEXT  OTHERS          --                --              NONVSAM
MHLRES2.XMIT.GTF    OTHERS          --                --              NONVSAM
-----
                        BOTTOM OF DATA
  
```

Figure 5-43 Identifying PDSEs

### Identifying PDSEs

You can use ISMF to display information associated with data set name type (DSNTYPE). Figure 5-43 shows a sample data set list obtained through a catalog. Navigating in the right direction, you find the Data Set Name Type column. LIBRARY indicates the data set is a PDSE, and OTHERS indicates that the data set is a PDS.

When you obtain the list through a catalog and the Data Set Name Type column contains nulls "-----" the data set name type is neither LIBRARY (PDSE) nor a PDS.

A saved data set list from a release prior to DFSMS can still be used, and the values of data set name type matches those under DFSMS. For example, PDSs are indicated as OTHERS in the Data Set Name Type column.

When defining a data class, the valid values you use for the Data Set Name Type field in the ISMF data set application are:

- ▶ EXTENDED, for extended format sequential data sets
- ▶ HFS, for hierarchical file system data sets
- ▶ LIBRARY, for PDSEs
- ▶ Or leave the field blank (for data sets not created in extended, HFS, or PDSE format)

## 5.44 System-managed data types

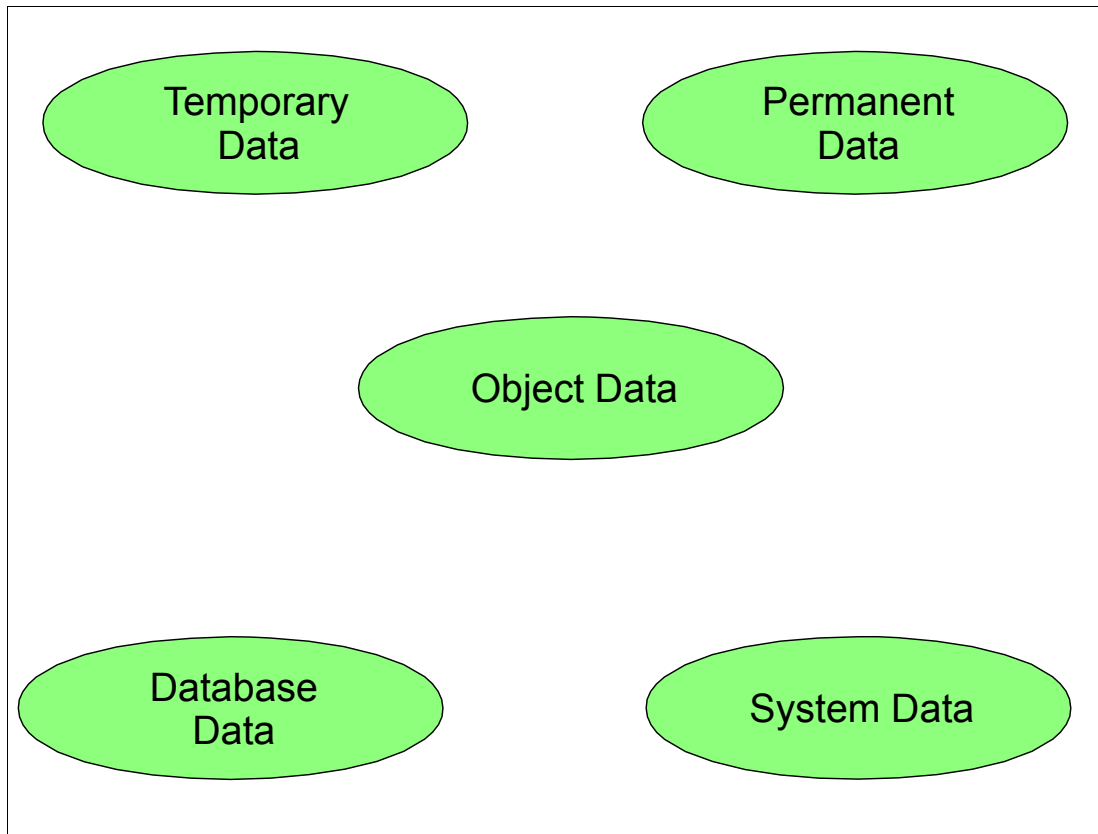


Figure 5-44 System-managed data types

### Data set types that can be system-managed

Now that you have experience with SMS using the minimal SMS configuration, you can plan system-managed data sets implementation. First you need to know which data sets can be SMS-managed and which data sets cannot be SMS-managed.

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

- |                       |   |
|-----------------------|---|
| <b>Temporary data</b> | Data sets used only for the duration of a job, job step, or terminal session, and then deleted. These data sets can be cataloged or uncataloged, and can range in size from small to very large.  |
| <b>Permanent data</b> | Data sets consisting of: <ul style="list-style-type: none"><li>• Interactive data</li><li>• TSO user data sets</li><li>• ISPF/PDF libraries you use during a terminal session</li></ul> Data sets classified in this category are typically small, and are frequently accessed and updated. |
| <b>Batch data</b>     | Data that is classified as either online-initiated, production, or test. <ul style="list-style-type: none"><li>• Data accessed as online-initiated are background jobs that an online facility (such as TSO) generates.</li></ul>   |

	<ul style="list-style-type: none"> <li>• Production batch refers to data created by specialized applications (such as payroll), that could be critical to the continued operation of your business or enterprise.</li> <li>• Test batch refers to data created for testing purposes.</li> </ul>
<b>VSAM data</b>	Data organized with VSAM, including VSAM data sets that are part of an existing database.
<b>Large data</b>	For most installations, large data sets occupy more than 10 percent of a single DASD volume. Note, however, that what constitutes a large data set is installation-dependent.
<b>Multivolume data</b>	Data sets that span more than one volume.
<b>Database data</b>	Data types usually having varied requirements for performance, availability, space, and security. To accommodate special needs, database products have specialized utilities to manage backup, recovery, and space usage. Examples include DB2, IMS, and CICS data.
<b>System data</b>	<p>Data used by MVS to keep the operating system running smoothly. In a typical installation, 30-to-50 percent of these data sets are high performance and are used for cataloging, error recording, and other system functions.</p> <p>Because these critical data sets contain information required to find and access other data, they are read and updated frequently, often by more than one system in an installation. Performance and availability requirements are unique for system data. The performance of the system depends heavily upon the speed at which system data sets can be accessed. If a system data set such as a master catalog is unavailable, the availability of data across the entire system and across other systems can be affected.</p> <p>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 which are allocated at MVS system initialization are not system-managed, because the SMS address space is not active at initialization time.</p>
<b>Object data</b>	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, once written, the data is not changed or updated. However, the data can be referenced many times during its lifetime.

## 5.45 Data types that cannot be system-managed

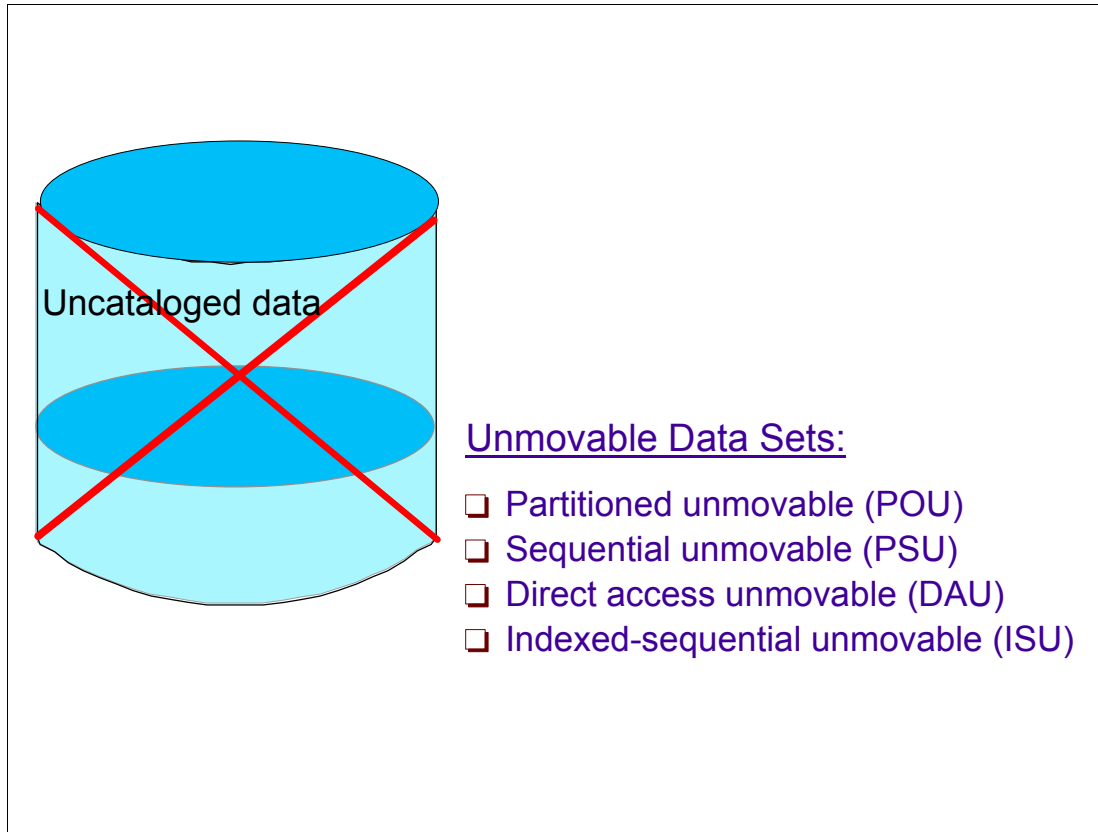


Figure 5-45 Data types that cannot be system-managed

### Data types that cannot be system-managed

All permanent DASD data under the control of SMS *must* be cataloged in integrated catalog facility (ICF) catalogs using the standard search order. The catalogs contain the information required for locating and managing system-managed data sets.

When data sets are cataloged, users do not need to know which volumes the data sets reside on when they reference them; they do not need to specify unit type or volume serial number. This is essential in an environment with storage groups, where users do not have private volumes.

Some data cannot be system-managed, as described here:

**Uncataloged data** Objects, stored in groups called *collections*, must have their collections cataloged in ICF catalogs because they, and the objects they contain, are system-managed data. The object access method (OAM) identifies an object by its collection name and the object's own name.

An object is described only by an entry in a DB2 object directory. An object collection is described by a collection name catalog entry and a corresponding OAM collection identifier table entry. Therefore, an object is accessed by using the object's collection name and the catalog entry.

When objects are written to tape, they are treated as tape data sets and OAM assigns two tape data set names to the objects. Objects in an object storage group being written to tape are stored as a tape data set named OAM.PRIMARY.DATA. Objects in an object backup storage group being written to tape are stored as a tape data set named OAM.BACKUP.DATA. Each tape containing objects has only one tape data set, and that data set has one of the two previous names. Because the same data set name can be used on multiple object-containing tape volumes, the object tape data sets are *not* cataloged.

If you do not already have a policy for cataloging all permanent data, it is a good idea to establish one now. For example, you can enforce standards by deleting uncataloged data sets.

### **Uncataloged data sets**

The system locates these with JOBCAT or STEPCAT statements. Data set LOCATEs using JOBCATs or STEPCATs are not permitted for system-managed data sets. You must identify the owning catalogs before you migrate these data sets to system management. The ISPF/PDF SUPER utility is valuable for scanning your JCL and identifying any dependencies on JOBCATs or STEPCATs.

### **Unmovable data sets**

Unmovable data sets cannot be system-managed. These data sets include:

- Data sets identified by the following data set organizations (DSORGs):
  - Partitioned unmovable (POU)
  - Sequential unmovable (PSU)
  - Direct access unmovable (DAU)
  - Indexed-sequential unmovable (ISU)
- Data sets with user-written access methods
- Data sets containing processing control information on the device or volume on which they reside, including:
  - Absolute track data that is allocated in absolute DASD tracks or on split cylinders
  - Location-dependent direct data sets

All unmovable data sets must be identified and converted for use in a system-managed environment. For information on identifying and converting unmovable data sets, see *z/OS DFSMSdss Storage Administration Guide*, SC35-0423.

## 5.46 Introduction to ISMF

```
■ Panel Help
-----
ISMF PRIMARY OPTION MENU - z/OS DFSMS V1 R6
Enter Selection or Command ===> _____

Select one of the following options and press Enter:
0 ISMF Profile - Specify ISMF User Profile
1 Data Set - Perform Functions Against Data Sets
2 Volume - Perform Functions Against Volumes
3 Management Class - Specify Data Set Backup and Migration Criteria
4 Data Class - Specify Data Set Allocation Parameters
5 Storage Class - Specify Data Set Performance and Availability
6 Storage Group - Specify Volume Names and Free Space Thresholds
7 Automatic Class Selection - Specify ACS Routines and Test Criteria
8 Control Data Set - Specify System Names and Default Criteria
9 Aggregate Group - Specify Data Set Recovery Parameters
10 Library Management - Specify Library and Drive Configurations
11 Enhanced ACS Management - Perform Enhanced Test/Configuration Management
C Data Collection - Process Data Collection Function
L List - Perform Functions Against Saved ISMF Lists
P Copy Pool - Specify Pool Storage Groups for Copies
R Removable Media Manager - Perform Functions Against Removable Media
X Exit - Terminate ISMF
Use HELP Command for Help; Use END Command or X to Exit.
```

Figure 5-46 ISMF Primary Option Menu panel

### Using ISMF

The Interactive Storage Management Facility (ISMF) helps you analyze and manage data and storage interactively. ISMF is an Interactive System Productivity Facility (ISPF) application. Figure 5-46 shows the first ISMF panel, the Primary Option Menu.

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 operates as an Interactive System Productivity Facility (ISPF) application. It is menu-driven with fast paths for many of its functions. ISMF uses the ISPF data-tag language (DTL) to give its functional panels on workstations the look of common user access (CUA) panels and a graphical user interface (GUI).



## 5.47 ISMF product relationships

- ❑ ISPF/PDF
- ❑ TSO/Extensions (TSO/E), TSO CLISTs, commands
- ❑ DFSMS
- ❑ Data Facility SORT (DFSORT)
- ❑ Resource Authorization Control Facility (RACF)
- ❑ Device Support Facilities (ICKDSF)
- ❑ IBM NaviQuest for MVS (NaviQuest), 5655-ACS

Figure 5-47 ISMF product relationships

### ISMF product relationships

ISMF works with the following products, which you should be familiar with:

- ▶ Interactive System Productivity Facility/Program Development Facility (ISPF/PDF), which provides the edit, browse, Data Set and Library utility functions.
- ▶ TSO/Extensions (TSO/E), TSO CLISTs and commands.
- ▶ DFSMS, which consists of four functional components: DFSMSdftp, DFSMSshsm, DFSMSdss, and DFSMSrmm. ISMF is designed to use the space management and availability management (backup/recovery) functions provided by those products.
- ▶ Data Facility SORT (DFSORT), which provides the record-level functions.
- ▶ Resource Authorization Control Facility (RACF), which provides the access control function for data and services.
- ▶ Device Support Facilities (ICKDSF) to provide the storage device support and analysis functions.
- ▶ IBM NaviQuest for MVS 5655-ACS.

ISMF also works with NaviQuest, which is a new product from IBM Storage System Division Software Products. NaviQuest is a testing and reporting tool that speeds and simplifies the tasks associated with DFSMS initial implementation and ongoing ACS routine and configuration maintenance. NaviQuest assists storage administrators by allowing more

automation of storage management tasks. More information on NaviQuest can be found in the NaviQuest User's Guide.

NaviQuest provides:

- ▶ A familiar ISPF panel interface to functions
- ▶ Fast, easy, bulk test-case creation
- ▶ ACS routine and DFSMS configuration-testing automation
- ▶ Storage reporting assistance
- ▶ Additional tools to aid with storage administration tasks
- ▶ Batch creation of data set and volume listings
- ▶ Printing of ISMF LISTs
- ▶ Batch ACS routine translation
- ▶ Batch ACS routine validation

## 5.48 What you can do with ISMF

- Edit, browse, and sort data set records
- Delete data sets and backup copies
- Protect data sets by limiting their access
- Copy data sets to another migration level
- Back up data sets and copy entire volumes, mountable optical volumes, or mountable tape volumes
- Recall data sets that have been migrated

Figure 5-48 What you can do with ISMF

### What you can do with ISMF

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

- ▶ 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 find out 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. Once 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:

- ▶ 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

Every site can control who can use certain functions described in this book. Your organization might require you to have authorization to use certain functions. Your security administrator can explain any restrictions your site has established.

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.

## 5.49 Accessing ISMF

```
Panel Help
-----
                ISMF PRIMARY OPTION MENU - z/OS DFSMS V1 R6
Enter Selection or Command ==> _____

Select one of the following options and press Enter:
0 ISMF Profile           - Specify ISMF User Profile
1 Data Set               - Perform Functions Against Data Sets
2 Volume                 - Perform Functions Against Volumes
3 Management Class      - Specify Data Set Backup and Migration Criteria
4 Data Class             - Specify Data Set Allocation Parameters
5 Storage Class          - Specify Data Set Performance and Availability
6 Storage Group          - Specify Volume Names and Free Space Thresholds
7 Automatic Class Selection - Specify ACS Routines and Test Criteria
8 Control Data Set      - Specify System Names and Default Criteria
9 Aggregate Group        - Specify Data Set Recovery Parameters
10 Library Management    - Specify Library and Drive Configurations
11 Enhanced ACS Management - Perform Enhanced Test/Configuration Management
C Data Collection        - Process Data Collection Function
L List                   - Perform Functions Against Saved ISMF Lists
P Copy Pool              - Specify Pool Storage Groups for Copies
R Removable Media Manager - Perform Functions Against Removable Media
X Exit                   - Terminate ISMF
Use HELP Command for Help; Use END Command or X to Exit.
```

Figure 5-49 ISMF Primary Option Menu panel for storage administrators

### 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 starts an ISMF session from the ISPF/PDF Primary Option Menu.
- ▶ To access ISMF directly from TSO, use the command:

```
ISPSTART PGM (DGT FMD001) NEWAPPL(DGT)
```

There are two Primary Option Menus, one for storage administrators, and another for end users:

1. Figure 5-49 shows the menu available to storage administrators; it includes additional applications not available to end users.
2. Figure 5-51 on page 271 shows the ISMF Primary Option Menu for end users.

Option 0 controls the user mode or the type of Primary Option Menu to be displayed. Refer to “ISMF Profile option” on page 270 for information on how to change the user mode.

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, refer to *DFSORT Installation and Customization Release 14*, SC33-4034.

## 5.50 ISMF Profile option

```
Panel Help
-----
                                ISMF PROFILE OPTION MENU
Enter Selection or Command ==> _____

Select one of the following options and Press Enter:

 0 User Mode Selection
 1 Logging and Abend Control
 2 ISMF Job Statement
 3 DFSMSdss Execute Statement
 4 ICKDSF Execute Statement
 5 Data Set Print Execute Statement
 6 IDCAMS Execute Statement
 X Exit

Use HELP command for Help; Use END command or X to Exit.
```

Figure 5-50 ISMF PROFILE OPTION MENU panel

### Setting the ISMF profile

Figure 5-50 shows the ISMF Profile Option Menu panel, Option 0 from the ISMF Primary Menu. Use this menu to control the way ISMF runs during the session. You can:

- ▶ Change the user mode from end user to storage administrator, or from storage administrator to end user
- ▶ Control ISMF error logging and recovery from abends
- ▶ Define statements for ISMF to use in processing your jobs, such as:
  - JOB statements,
  - DFSMSdss
  - Device Support Facilities (ICKDSF)
  - Access Method Services (IDCAMS)
  - PRINT execute statements in your profile

You can select ISMF or Interactive System Productivity Facility (ISPF) JCL statements for processing batch jobs.

## 5.51 Navigating through ISMF

```
■ Panel Help
-----
                ISMF PRIMARY OPTION MENU - z/OS DFSMS V1 R6
Enter Selection or Command ===> _____

Select one of the following options and press Enter:

0 ISMF Profile           - Change ISMF User Profile
1 Data Set              - Perform Functions Against Data Sets
2 Volume                - Perform Functions Against Volumes
3 Management Class     - Specify Data Set Backup and Migration Criteria
4 Data Class           - Specify Data Set Allocation Parameters
5 Storage Class        - Specify Data Set Performance and Availability
9 Aggregate Group      - Specify Data Set Recovery Parameters
L List                 - Perform Functions Against Saved ISMF Lists
R Removable Media Manager - Perform Functions Against Removable Media
X Exit                 - Terminate ISMF

Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-51 End user ISMF Primary Option Menu

### Navigating through ISMF

ISMF provides an action bar-driven interface that exploits many of the usability features of Common User Access (CUA) interfaces. The panels look different than in previous releases: all screens will be mixed case and most will have action bars at the top.

### Navigating without using the action bar

You can still navigate through ISMF using the standard method of typing in a selection number and pressing Enter.

### Using the action bar

Most ISMF panels have action bars at the top. The choices display in white (by default). The action bar gives you another way to move through ISMF. If the cursor is located somewhere on the panel, there are several ways to move the cursor to the action bar:

- ▶ Using the Tab key
- ▶ Using the mouse button
- ▶ Using the cursor manually

After you have chosen an action, press Enter to open the menu. Figure 5-54 on page 274 shows the List pull-down menu for the Data Set List panel. Notice the input field in the upper left corner. There, type the number of the action you want, then press Enter.

Figure 5-51 shows the options available for end users.

## 5.52 Obtaining information about a panel field

```
HELP----- DATA SET LIST LINE OPERATORS -----HELP
COMMAND ===>

  Use ENTER to see the line operator descriptions in sequence or choose them
  by number:

   1  LINE OPERATOR  10  DUMP                20  HRECOVER
      Overview      11  EDIT                21  MESSAGE
   2  ALTER          12  ERASE                22  RELEASE
   3  BROWSE         13  HALTERDS            23  RESTORE
   4  CATLIST        14  HBACKDS            24  SECURITY
   5  CLIST          15  HBDELETE           25  SORTREC
   6  COMPRESS       16  HDELETE            26  TSO Commands and CLISTS
   7  CONDENSE       17  HIDE                27  VSAMREC
   8  COPY           18  HMIGRATE           28  VTOCLIST
   9  DELETE         19  HRECALL

  Use ENTER to continue, END to exit Help.
```

Figure 5-52 Obtaining information using Help command

### Using the Help Program Function Key (PFK)

On any ISMF panel, you can use ISMF Help Program Function Key (PFK) to obtain information about the panel you are using and the panel fields. By positioning the cursor in a specific field, you can obtain detailed information related to that field.

Figure 5-52 shows the panel you reach when you press the Help PFK with the cursor in the Line Operator field of the panel shown in Figure 5-54 on page 274. The panel shows the commands available to enter in that field. If you want an explanation about a specific command, type the option corresponding to the desired command and a panel is displayed showing information about the command function.

You can exploit the Help PFK, when defining classes, to obtain information about what you have to enter in the fields. Place the cursor in the field and press the Help PFK.

To see and change the assigned functions to the PFKs, enter the **KEYS** command in the Command field.



## 5.53 Data Set option

```
Panel Defaults Utilities Scroll Help
-----
DATA SET SELECTION ENTRY PANEL Page 1 of 5
Command ==> _____

For a Data Set List, Select Source of Generated List . . 2 (1 or 2)

1 Generate from a Saved List          Query Name To
  List Name . . _____          Save or Retrieve _____

2 Generate a new list from criteria below
  Data Set Name . . 'MHLRES2.**'
  Enter "/" to select option - Generate Exclusive list
  Specify Source of the new list . . 2 (1 - VTOC, 2 - Catalog)
  1 Generate list from VTOC
    Volume Serial Number . . . _____ (fully or partially specified)
  2 Generate list from Catalog
    Catalog Name . . . _____
    Volume Serial Number . . . _____ (fully or partially specified)
    Acquire Data from Volume . . . . . Y (Y or N)
    Acquire Data if DFSMShsm Migrated . . N (Y or N)

Use ENTER to Perform Selection; Use DOWN Command to View next Selection Panel;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-53 ISMF Data Set Selection Entry Panel

### Data Set Selection Entry panel

Figure 5-53 shows the panel that appears when you select option 1 (Data Set) from the ISMF Primary Option Menu.

The Data Set Application constructs a list of data sets, using the filters provided in the panel. Figure 5-54 on page 274 shows the Data Set List generated, in our environment, for the selection criteria used in Figure 5-53.

## 5.54 Obtaining a data set list

Panel List Dataset Utilities Scroll Help		DATA SET LIST				
-----						
Comman	1. Clear					Scroll ==> CSR
	2. Reshow					Entries 1-22 of 96
	3. Fold					Data Columns 3-6 of 39
Enter	4. Refresh					
	5. Filter...					
L	6. Filter Clear					
0	7. Format View...					
--	8. Sort...					
	9. Print...					
		NAME	ALLOC SPACE	ALLOC USED	% NOT USED	COMPRESSED FORMAT
		-----	--(3)--	--(4)--	-(5)-	---(6)---
			-----	-----	---	---
		NT	277	55	80	---
		MHLRES2.BCDS.PRINT1	277	55	80	---
		MHLRES2.BCDS.PRINT2	277	55	80	---
		MHLRES2.BCDS.PRINT3	277	0	100	---
		MHLRES2.BCDS.PRINT4	277	55	80	---
		MHLRES2.BCDS.PRINT5	277	111	59	---
		MHLRES2.BROADCAST	55	55	0	---
		MHLRES2.CLIST.CONVTOD	55	55	0	---
		MHLRES2.CNTL.JCL	4980	3818	23	---
		MHLRES2.DISPLAY.H.OUT	1107	55	95	---
		MHLRES2.DITPROF	55	55	0	---
		MHLRES2.FIXCDS.CB	1107	55	95	---
		MHLRES2.FIXCDS.CB1	1107	55	95	---
		MHLRES2.FIXCDS.CB2	1107	55	95	---
		MHLRES2.FIXCDS.CP1	1107	55	95	---
		MHLRES2.FRJCL.UNLOAD	221	55	75	---
		MHLRES2.FSR	830	55	93	---
		MHLRES2.FSR.FSRSTAT	55	55	0	---

Figure 5-54 Data Set List panel using action bars

### Data Set List panel

You can use line operators to execute tasks with individual data sets. Use list commands to execute tasks with a group of data sets. These tasks include editing, browsing, recovering unused space, copying, migrating, deleting, backing up, and restoring data sets.

TSO commands and CLISTs can also be used as line operators or list commands. You can save a copy of a data set list and reuse it later.

If ISMF is unable to get certain information required to check if a data set meets the selection criteria specified, that data set is also to be included in the list. This is indicated by dashes on the corresponding column. For example, if ISMF is unable to check if a data set meets the specified volume serial number criteria, that data set still appears in the list with dashes in the corresponding Volume Serial Number field.

Figure 5-54 shows a data set list. The Data Fields field shows how many fields you have in the list. You can navigate throughout these fields using Right and Left PF keys. The figure also shows the use of the actions bar; in this case the cursor was placed in the List action, the Enter key was pressed, and the list options were shown.

## 5.55 Volume Option

```

Panel Help
-----
                                VOLUME LIST SELECTION MENU 1
Enter Selection or Command ==> _____

1  DASD                        - Generate a List of DASD Volumes
2  Mountable Optical           - Generate a List of Mountable Optical Volumes
3  Mountable Tape              - Generate a List of Mountable Tape Volumes
-----

Panel Defaults Utilities Scroll Help
-----
                                VOLUME SELECTION ENTRY PANEL 2
Command ==> _____ Page 1 of 3

Select Source to Generate Volume List . . . 2 (1 - Saved list, 2 - New list)
1  Generate from a Saved List          Query Name To
   List Name . . . _____         Save or Retrieve _____
2  Generate a New List from Criteria Below
   Specify Source of the New List . . . 1 (1 - Physical, 2 - SMS)
   Optionally Specify One or More:
   Enter "/" to select option - Generate Exclusive list
   Type of Volume List . . . 1         (1-Online,2-Not Online,3-Either)
   Volume Serial Number . . . SBOX*   (fully or partially specified)
   Device Type . . . _____        (fully or partially specified)
   Device Number . . . _____      (fully specified)
   To Device Number . . . _____   (for range of devices)
   Acquire Physical Data . . . Y       (Y or N)
   Acquire Space Data . . . Y         (Y or N)
   Storage Group Name . . . _____ (fully or partially specified)
   CDS Name . . . _____          (fully specified or 'Active')

Use ENTER to Perform Selection; Use DOWN Command to View next Selection Panel;
Use HELP Command for Help; Use END Command to Exit.

```

Figure 5-55 Volume Section Entry panel

### Volume option

Selecting option 2 (Volume) from the ISMF Primary Menu takes you to the Volume List Selection Menu Panel, shown in the first part (1) of Figure 5-55. Here you can choose the volume type from DASD, mountable optical or mountable tape.

Selecting option 1 (DASD) displays the Volume Selection Entry Panel, shown in the second part (2) of Figure 5-55.

Using filters, you can select a Volume List Panel, shown in Figure 5-56 on page 276.

## 5.56 Obtaining a volume list

```

Panel List Utilities Scroll Help
-----
                                VOLUME LIST
Command ==>                               Scroll ==> HALF
                                           Entries 1-21 of 178
Enter Line Operators below:                Data Columns 3-8 of 40

  LINE   VOLUME  FREE   %   ALLOC   FRAG   LARGEST   FREE
 OPERATOR SERIAL SPACE  FREE SPACE INDEX  EXTENT  EXTENTS
---(1)--- --(2)-- --(3)--- (4) - --(5)--- --(6)-- --(7)--- --(8)---
      SBOXAA   190522    7   2580978    52   168996    4
      SBOXAB   1301612   47   1469888    0   1301501    2
      SBOXAC   627952   23   2143548    0   627952    1
      SBOXAD   262016    9   2509484   10   258751    5
      SBOXAE   1519359   55   1252141   164   459122    6
      SBOXAF   1878102   68   893398   102   1211691    6
      SBOXA0    166     0   2771334    1    166     1
      SBOXA1    221     0   2771279    0    221     1
      SBOXA2   1081596   39   1689904   53   909723    9
      SBOXA3    221     0   2771279    0    221     1
      SBOXA4   1113912   40   1657588   121   678972   16
      SBOXA5   1065217   38   1706283   119   583351    5
      SBOXA6   128490    5   2643010   90   103755    5
      SBOXA7   253937    9   2517563   326   124506   99
      SBOXA8   32095    1   2739405   812   1328   100
      SBOXA9   13502    0   2757998   921    830   83
      SBOXBA   1254300   45   1517200    0   1254189    2
      SBOXBB   1967746   71    803754    0   1967193    2
      SBOXBC   308277   11   2463223   92   232245    4
      SBOXBD   142767    5   2628733    0   142767    1
      SBOXBE   604490   22   2167010   52   516893    6

```

Figure 5-56 Volume List panel

### Volume List panel

The volume application constructs a list of DASD volumes, mountable optical volumes, or mountable tape volumes. Use line operators to do tasks with an individual volume. These tasks include consolidating or recovering unused space, copying, backing up, and restoring volumes. TSO commands and CLISTs can also be line operators or list commands.

You can save a copy of a volume list and reuse it later. With the list of mountable optical volumes or mountable tape volumes, you can only browse the list.

For information about when to select the Volume option and tasks you can do using the Volume Application, you can:

- ▶ Type the **HELP** command in the Command field.
- ▶ Press the Help PFK.
- ▶ Refer to *z/OS DFSMS: Using the Interactive Storage Management Facility*, SC26-7411.

## 5.57 Management Class option

```
Panel Utilities Scroll Help
MANAGEMENT CLASS APPLICATION SELECTION Page 1 of 2
Command ==> _____
To perform Management Class Operations, Specify:
CDS Name . . . . . 'ACTIVE'
Management Class Name . . . * _____ (1 to 44 character data set name or 'Active' )
                                         (For Management Class List, fully or
                                         partially specified or * for all)

Select one of the following options :
1 1. List - Generate a list of Management Classes
2 2. Display - Display a Management Class
3 3. Define - Define a Management Class
4 4. Alter - Alter a Management Class

If List Option is chosen,
Enter "/" to select option - Respecify View Criteria
                           - Respecify Sort Criteria

Use ENTER to Perform Selection; Use DOWN Command to View next Selection Panel;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-57 Management Class Selection Menu panel

### Management Class Application Selection panel

Figure 5-57 shows the panel displayed when you select option 3 (Management Class) from the ISMF Primary Menu. Use this option to display, modify, and define options for the management classes of the SMS. It also constructs a list of the available management classes.

Figure 5-58 on page 278 shows the management class list generated by the filters chosen in this panel, using option 2 (Display).

## 5.58 Management Class List panel

```

Panel List Utilities Scroll Help
-----
MANAGEMENT CLASS LIST
Command ==>
CDS Name : ACTIVE
Enter Line Operators below:
Scroll ==> HALF
Entries 1-18 of 32
Data Columns 3-7 of 40

LINE OPERATOR  MGMTCLAS  EXPIRE  EXPIRE  RET  PARTIAL  PRIMARY
  (1)         (2)         (3)     (4)     (5)   (6)      (7)
  (1)         (2)         (3)     (4)     (5)   (6)      (7)
ABTEST        CATALOG    NOLIMIT  NOLIMIT  NOLIMIT  NO        2
D98090        D98090     NOLIMIT  NOLIMIT  NOLIMIT  NO        2
D99000        D99000     NOLIMIT  NOLIMIT  NOLIMIT  NO        2
FKSMF         FKSMF      NOLIMIT  NOLIMIT  NOLIMIT  YES_IMMED 7
HFS           HFS        NOLIMIT  NOLIMIT  NOLIMIT  NO        90
HGMC          HGMC       NOLIMIT  NOLIMIT  NOLIMIT  NO        2
HSMFR        HSMFR      NOLIMIT  NOLIMIT  NOLIMIT  NO        ---
MCABARS      MCABARS    NOLIMIT  NOLIMIT  NOLIMIT  NO        2
MCDB20       MCDB20     NOLIMIT  NOLIMIT  NOLIMIT  NO        ---
MCDB21       MCDB21     NOLIMIT  NOLIMIT  NOLIMIT  NO        7
MCDB22       MCDB22     NOLIMIT  NOLIMIT  NOLIMIT  NO        7
MCGOML1      MCGOML1    NOLIMIT  NOLIMIT  NOLIMIT  NO        0
MCGOML2      MCGOML2    NOLIMIT  NOLIMIT  NOLIMIT  NO        0
MCN0ACT      MCN0ACT    NOLIMIT  NOLIMIT  NOLIMIT  NO        ---
MCN0MIG      MCN0MIG    NOLIMIT  NOLIMIT  NOLIMIT  NO        0
MCRMM        MCRMM      NOLIMIT  NOLIMIT  NOLIMIT  NO        ---
MC365        MC365      NOLIMIT  NOLIMIT  NOLIMIT  YES       10
  
```

Figure 5-58 Listing the management class defined in the SMS configuration

### Management Class List panel

Figure 5-58 shows the partial contents of the ISMF Management Class List panel. Note how many data columns are available. You can navigate through them using right and left PFKs.

To view the commands, you can use in the Line Operator field (marked with a circle in the figure), place the cursor in the field and press the Help PFK.

To see the PFKs, type the **KEYS** command in the Command panel field and press Enter.

## 5.59 Data Class option

```
Panel Utilities Help
-----
DATA CLASS APPLICATION SELECTION
Command ==> _____

To perform Data Class Operations, Specify:
CDS Name . . . . . 'ACTIVE'
Data Class Name . . . . . KEY* (1 to 44 character data set name or 'Active' )
                               (For Data Class List, fully or partially
                               specified or * for all)

Select one of the following options :
1. List - Generate a list of Data Classes
2. Display - Display a Data Class
3. Define - Define a Data Class
4. Alter - Alter a Data Class

If List Option is chosen,
Enter "/" to select option - Respecify View Criteria
                          - Respecify Sort Criteria

Use ENTER to Perform Selection;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-59 Data Class Application Selection panel

### Displaying information about data classes

Figure 5-59 shows the panel displayed when you choose option 4 (Data Class) from the ISMF Primary Menu. Use this option to define the way data sets are allocated in your installation.

Data class attributes are assigned to a data set when the data set is created. They apply to *both* DFSMS-managed and non-DFSMS-managed data sets. Attributes specified in JCL or equivalent allocation statements override those specified in a data class. Individual attributes in a data class can be overridden by JCL, TSO, IDCAMS, and dynamic allocation statements.

Figure 5-60 on page 280 shows the Data Class List generated by the filters located in this panel, when using the option 1 Display.

## 5.60 Obtaining a data class listing

```
Panel List Utilities Scroll Help
-----
Command ==>                                DATA CLASS LIST
                                           Scroll ==> HALF
                                           Entries 1-3 of 3
                                           Data Columns 3-9 of 44

CDS Name : ACTIVE

Enter Line Operators below:

  LINE      DATACLAS
  OPERATOR  NAME      RECORG  RECFM  LRECL  KEYLEN  KEYOFF  AVGREC  AVG
  (1)----- (2)---- (3)--- (4)--- (5)--- (6)--- (7)--- (8)--- (9)---
  KEYDEXG   KS        ----   300   8      0 K     300
  KEYDEXT   KS        ----   300   8      0 K     300
  KEYEXT2   KS        ----   80    8      0 K     30
  -----
                BOTTOM OF DATA
```

Figure 5-60 Data Class List panel

### Data Class List panel

Figure 5-60 shows the data class listing obtained using the filters shown in Figure 5-59 on page 279.

Typing **DISPLAY** line command in the Line Operator field, in front a data class name, shows the informations about that data class, with out have to navigate throughout the use of Right and Left PFks, as shown in Figure 5-61 on page 281.



## 5.61 Displaying data class information

```
Panel Utilities Scroll Help
-----
                                DATA CLASS DISPLAY
                                Page 1 of 4
Command ==> _____

CDS Name . . . . : ACTIVE
Data Class Name : KEYEDEXG

Description : USING GUARANTEED SPACE

Recfm . . . . . :
Lrecl . . . . . : 300
Space Avgrec . . . . . : K
  Avg Value . . . . . : 300
  Primary . . . . . : 600
  Secondary . . . . . : 100
  Directory . . . . . :
Retpd Or Expdt . . . . . :
Volume Count . . . . . : 4
  Add'l Volume Amount . . :

Use DOWN Command to View next Panel;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-61 Displaying information about a data class

### Data Class Display panel

To see all the attributes specified to the data class, navigate through this application by using the up and down PFKs.

## 5.62 Storage Class option

```
Panel Utilities Help
-----
                STORAGE CLASS APPLICATION SELECTION
Command ==> _____
To perform Storage Class Operations, Specify:
CDS Name . . . . . 'ACTIVE'
                (1 to 44 character data set name or 'Active' )
Storage Class Name . . *_____ (For Storage Class List, fully or
                                partially specified or * for all)
Select one of the following options :
 1 1. List           - Generate a list of Storage Classes
 2 2. Display        - Display a Storage Class
 3 3. Define         - Define a Storage Class
 4 4. Alter          - Alter a Storage Class
 5 5. Cache Display - Display Storage Classes/Cache Sets
If List Option is chosen,
Enter "/" to select option  - Respecify View Criteria
                             - Respecify Sort Criteria

If Cache Display is Chosen, Specify Cache Structure Name . . _____

Use ENTER to Perform Selection;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-62 Storage Class Application Selection panel

### Storage Class Application Selection panel

Figure 5-62 shows the Storage Class Application Selection panel displayed when you select option 5 (Storage Class) of the ISMF Primary Menu.

The Storage Class Application Selection panel lets the storage administrator specify performance objectives and availability attributes that characterize a collection of data sets.

For objects, the storage administrator can define the performance attribute Initial Access Response Seconds. A data set or object must be assigned to a storage class in order to be managed by DFSMS.

Figure 5-63 on page 283 shows the storage class list generated by the filters located in Figure 5-62 with the option 1 Display.

## 5.63 Storage Class List panel

```

Panel List Utilities Scroll Help
-----
                                STORAGE CLASS LIST
Command ==>
                                Scroll ==> HALF
                                Entries 1-18 of 52
                                Data Columns 3-7 of 20
CDS Name : ACTIVE
Enter Line Operators below:

  LINE   STORCLAS  DIR RESP  DIR   SEQ RESP  SEQ
OPERATOR NAME      (MSEC)  BIAS  (MSEC)  BIAS  AVAILABILITY
---(1)--- --(2)--- --(3)--- (4)-  --(5)--- (6)-  -----(7)-----
      DB8B          -         -         -         -         NOPREF
      DB8BLOG1     -         -         -         -         NOPREF
      DB8BLOG2     -         -         -         -         NOPREF
      DB8BMISC     -         -         -         -         NOPREF
      DB8XA        -         -         -         -         NOPREF
      DB8XC        -         -         -         -         NOPREF
      DB8XD        -         -         -         -         NOPREF
      DB8XL        -         -         -         -         NOPREF
      DJLSTP2      -         -         -         -         NOPREF
      DJLVSAM      -         -         -         -         NOPREF
      FKSMF        -         -         -         -         NOPREF
      GSPACE       -         -         -         -         NOPREF
      GSSTD        -         -         -         -         STANDARD
      HSMFR        -         -         -         -         NOPREF
      NONSMS       5 R         5 R         -         -         NOPREF
      OPENMVS      -         -         -         -         NOPREF
      PAVNOPRF     -         -         -         -         NOPREF
      PAVPREF      -         -         -         -         NOPREF

```

Figure 5-63 Storage Class List panel

### Storage Class List panel

You can specify the DISPLAY line operator next to any class name on a class list to generate a panel that displays values associated with that particular class. This information can help you decide whether you need to assign a new DFSMS class to your data set or object.

If you determine that a data set you own should be associated with a different management class or storage class, and if you have authorization, you can use the ALTER line operator against a data set list entry to specify another storage class or management class.

## 5.64 List option

```
Panel List Utilities Scroll Help
-----
                                SAVED ISMF LISTS
Command ===>                                Scroll ===> PAGE
                                           Entries 1-1 of 1
Enter Line Operators below:                Data Columns 3-7 of 8

  LINE      LIST      LIST      LAST DATE   LAST TIME   LAST MOD   LIST ROW
  OPERATOR  NAME       TYPE      MODIFIED    MODIFIED    USERID    COUNT
  --- (1) --- -- (2) --- -- (3) --- --- (4) --- --- (5) --- -- (6) --- -- (7) ---
                AUGLIST  STORCLAS  2004/08/20  19:18      VALERIA    52
  -----
                                BOTTOM OF DATA
```

Figure 5-64 Saved ISMF Lists panel

### Saving ISMF lists

After obtaining a list (data set, data class, and storage class), you can save the list by typing **SAVE listname** in the Command panel field. To see the saved lists, use the option **L** (List) in the ISMF Primary Option Menu.

The List Application panel displays a list of all lists saved from ISMF applications. Each entry in the list represents a list that was saved. If there are no saved lists to be found, the ISMF Primary Option Menu panel is redisplayed with the message that the list is empty.

You can reuse and delete saved lists. From the List Application, you can reuse lists as if they were created from the corresponding application. You can then use line operators and commands to tailor and manage the information in the saved lists.

For more about the ISMF panel, refer to *z/OS DFSMS: Using the Interactive Storage Management Facility*, SC26-7411.

## 5.65 Removable Media Manager option

```
Panel Help
-----
EDG@PRIM          REMOVABLE MEDIA MANAGER (DFSMSrmm)
Option ==>

0  OPTIONS        - Specify dialog options and defaults
1  USER          - General user facilities
2  LIBRARIAN      - Librarian functions
3  ADMINISTRATOR  - Administrator functions
4  SUPPORT        - System support facilities
5  COMMANDS       - Full DFSMSrmm structured dialog
6  LOCAL         - Installation defined dialog
X  EXIT          - Exit DFSMSrmm Dialog

Enter selected option or END command.  For more info., enter HELP or PF1.

5695-DF1 (C) COPYRIGHT IBM CORPORATION 1993
```

Figure 5-65 Removable Media Manager option

### Removable Media Manager primary option menu

Figure 5-65 shows option **R** (Removable Media Manager) of the ISMF Primary Option Menu. This option displays the Primary Option Menu of the Removable Media Manager application.

Under normal circumstances, the DFSMSrmm subsystem starts automatically through IPL procedures, either standard or as modified by your installation. In exceptional cases, such as after recovery of the DFSMSrmm control data set, you might need to restart the subsystem.

Data Facility Removable Media Manager for MVS/DFP Version 3 (DFRMM) Program Offering provides support for non-system-managed tape libraries. When you use DFRMM and DFSMSrmm together, or multiple DFSMSrmm systems, they can share the same control data set. When both DFRMM and DFSMSrmm share the control data set, you can use the DFRMM ISPF dialog and RMM TSO subcommands to display all information that has been recorded in the control data set. There are some restrictions on using the RMM TSO subcommands from DFRMM, and from DFSMSrmm on a non-system-managed tape system, to add and change information in the control data set.



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 288. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *VSAM Demystified*, SG24-6105
- ▶ *DFSMSStvs Overview and Planning Guide*, SG24-6971
- ▶ *DFSMSStvs Presentation Guide*, SG24-6973
- ▶ *z/OS DFSMS V1R3 and V1R5 Technical Guide*, SG24-6979

## Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS DFSMSStvs Administration Guide*, GC26-7483
- ▶ *Device Support Facilities User's Guide and Reference Release 17*, GC35-0033
- ▶ *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
- ▶ *DFSMS Optimizer User's Guide and Reference*, SC26-7047
- ▶ *z/OS DFSMSStvs Planning and Operating Guide*, SC26-7348
- ▶ *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394
- ▶ *z/OS DFSMSsdfp Storage Administration Reference*, SC26-7402
- ▶ *z/OS DFSMSrmm Guide and Reference*, SC26-7404
- ▶ *z/OS DFSMSrmm Implementation and Customization Guide*, SC26-7405
- ▶ *z/OS DFSMS Implementing System-Managed Storage*, SC26-7407
- ▶ *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 DFSMSsdfp Utilities*, SC26-7414
- ▶ *z/OS Network File System Customization and Operation*, SC26-7417
- ▶ *z/OS Network File System User's Guide*, SC26-7419
- ▶ *z/OS DFSORT: Getting Started*, SC26-7527
- ▶ *DFSORT Installation and Customization Release 14*, SC33-4034
- ▶ *z/OS DFSMSShsm Storage Administration Guide*, SC35-0421
- ▶ *z/OS DFSMSShsm Storage Administration Reference*, SC35-0422

- ▶ *z/OS DFSMSdss Storage Administration Guide*, SC35-0423
- ▶ *z/OS DFSMSdss Storage Administration Reference*, SC35-0424
- ▶ *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
- ▶ *Tivoli Decision Support for OS/390 System Performance Feature Reference Volume I*, SH19-6819

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

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

## Help from IBM

IBM Support and downloads

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

IBM Global Services

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





# ABCS of z/OS System Programming Volume 3

(0.5" spine)  
0.475" x 0.873"  
250 x 459 pages







# ABCs of z/OS System Programming Volume 3



**DFSMS, data set basics**

**Storage management hardware and software**

**System-managed storage, ISMF**

The ABCs of z/OS System Programming is an eleven-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.

If you would like to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection will serve as a powerful technical tool.

Volume 3 describes an introduction to DFSMS, data set basics, storage management hardware and software, system-managed storage, and ISMF.

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:  
[ibm.com/redbooks](http://ibm.com/redbooks)**

SG24-6983-00

ISBN 0738491403