z/OS Basic Skills Information Center

Data and storage management on z/OS

z/OS Basic Skills Information Center

Data and storage management on z/OS

Note
Before using this information and the product it supports, read the information in "Notices" on page 23.

This edition applies to z/OS (product number 5694-A01).

We appreciate your comments about this publication. Comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Send your comments through this Web site:

http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp?topic=/com.ibm.zcontact.doc/webqs.html

Contents

Introduction to data and storage management on z/OS

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.

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.

Business systems need to be available continuously, efficient, easy to use, and secure from unauthorized users. System outages can be costly because users cannot access the system and data loss might result. As businesses grow, the systems tend to become extremely complicated and difficult to maintain, which results in lost productivity and time spent troubleshooting problems.

The DFSMS™ software suite, together with IBM® hardware products, and your installation-specific requirements for data and resource management, comprise the key to system-managed storage in a z/OS® environment. The elements of DFSMS automate and centralize storage management, based on policies your installation defines for availability, performance, space, and security.

Chapter 1. The DFSMS software suite

DFSMS is a software suite that automatically manages data from creation to expiration. The base element of this suite, DFSMSdfp™, performs the essential data, storage, and device management functions of the system. Other DFSMS features in the suite complement DFSMSdfp to provide a fully integrated approach to data and storage management.

In a system-managed storage environment, the DFSMS suite automates and centralizes storage management based on the policies that your installation defines for availability, performance, space, and security. Storage management policies reduce the need for users to make many detailed decisions that are not related to their business objectives.

DFSMSdfp

Storage management
DFSMSdfp includes Interactive Storage Management Facility (ISMF), which lets you define and maintain policies to manage your storage resources. These policies help to improve the usage of storage devices and to increase levels of service for user data, with minimal effort required from users. Storage Management Subsystem (SMS) uses these policies to manage storage for the operating system. More specifically, the storage administrator uses SMS to define data classes, storage classes, management classes, storage groups, aggregate groups, copy pools, and automatic class selection routines. You can also use the NaviQuest tool under ISMF to migrate to SMS, maintain your SMS configuration, and perform testing, implementation, and reporting tasks in batch.

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.

Data management
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.

Device management
DFSMSdfp can be used when you define your input and output (I/O) devices to the system and in controlling those devices in the z/OS environment.

Distributed data access
Distributed data access allows all authorized systems and users in a network to use system-managed storage or automated storage management.DFSMSdfp uses the Distributed FileManager/MVS (DFM) or the z/OS Network File System to enable remote clients in a network to access data and storage resources on z/OS systems.

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

Advanced copy services
Advanced Copy Services 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.

DFSMSdss™

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.

DFSMShsm™

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

DFSMSrmm™

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.

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.

DFSMStvs

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. DFSMStvs is built on top of VSAM record-level sharing (RLS), which permits sharing of recoverable VSAM data sets at the record level.

Chapter 2. Data and storage management policies

To allow your business to grow efficiently and profitably, you want to find ways to control the growth of your information systems and use your current storage more effectively. In an SMS-managed storage environment, your enterprise establishes centralized policies for how to use your hardware resources. The Interactive Storage Management Facility (ISMF) provides the user interface for defining and maintaining these policies, while the Storage Management Subsystem (SMS) governs the system.

Data and storage management policies balance your available resources with your users' requirements for data availability, performance, space, and security. SMS implements these policies and manages most of your storage management tasks. This frees users from manually administering storage and makes more efficient use of your storage resources.

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

- What performance objectives are required at your site?
- 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?

To implement a policy for managing storage, your storage administrator defines classes of space management, performance, and availability requirements for data sets at your installation. 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 and applies them 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 of the data.

The ISMF panels make it easy to define SMS classes and groups, test and validate ACS routines, and perform other tasks to analyze and manage your storage.

Chapter 3. DFSMS tools for data and storage management

DFSMSdfp provides several types of tools for system programmers and database administrators to use to manage the organization and storage of data in the z/OS environment.

With DFSMS tools, you can use access methods with macro instructions to organize and process a data set or object; use access method services commands to manage data sets, volumes, and catalogs; use utilities to perform tasks such as copying or moving data. You also can use system commands to display and set SMS configuration parameters, callable services to write advanced application programs, and installation exits to customize DFSMS.

DFSMSdfp access methods

DFSMSdfp provides several access methods for formatting and accessing data. An access method defines the organization of the data in a data set and the technique by which the data is stored and retrieved. DFSMSdfp access methods have their own data set structures to organize data, macro instructions to process data sets, and utility programs to manipulate data sets.

Table 1 describes the access methods that DFSMSdfp uses.

Table 1. DFSMSdfp access methods

Access method	Description	Data set organization
Basic partitioned access method (BPAM)	Use BPAM to create and retrieve program and data libraries on DASD. BPAM arranges records as members of PDSs, PDSEs, or z/OS UNIX directories.	• PDS • PDSE • z/OS UNIX
Basic sequential access method (BSAM)	Use BSAM to process data sets sequentially. You organize the records into blocks for retrieval.	• Sequential data sets • Extended-format data sets • PDS members • PDSE members • z/OS UNIX files
Object access method (OAM)—OSREQ interface	Use OAM to store, back up, and retrieve objects on DASD, optical, and tape storage.	• Objects
Queued sequential access method (QSAM)	Use QSAM to process data sets sequentially. QSAM collects the records into blocks.	• Sequential data sets • Extended-format data sets • PDS members • PDSE members • z/OS UNIX files
Virtual storage access method (VSAM)	Use VSAM for direct or sequential processing of records on DASD. VSAM arranges records by an index key, by relative byte address, or by relative record number. VSAM catalogs data sets for easy retrieval.	• Entry-sequenced data sets • Key-sequenced data sets • Linear data sets • Relative record data sets • HFS files

DFSMS also supports the basic direct access method (BDAM) for coexistence with previous operating systems.

You can use assembler language macro instructions to create, maintain, and process all the data set types supported by the access methods described in Table 1 on page 7. Macro instructions control data set allocation, input and output, and data security.

Each compiler provides facilities to create, read, and write data sets. Your compiler documentation describes how to use the access method facilities.

The following are a few of the functions that the macro instructions perform:

- Control block macros generate information that the access method needs to process the data sets.
- Request macros retrieve, update, delete, or insert logical records into data sets.
- Checkpoint/restart functions establish checkpoints during a program and restart the job at a checkpoint or at the beginning of a job step.

Access Method Services (IDCAMS) commands

Access method services, also known as IDCAMS, creates and maintains VSAM data sets.

With access method services, you can perform the following tasks:

- Define VSAM data sets.
- Define and build alternate indexes.
- Back up and restore VSAM data sets.
- Copy data sets.
- Print the contents of data sets.
- Delete data sets.
- Collect information about data sets.
- Examine the structural consistency of VSAM key-sequenced data sets.
- Control DASD cache.
- List tape volume (VOLCAT) catalog entries.
- Diagnose catalog errors.
- Recover from catalog errors.
- Define system-managed libraries and volumes.
- Define extended addressability for an extended-format VSAM data set to support a data set size greater than 4 GB.
- Encrypt and decrypt data sets.

You also can define VSAM data sets using JCL or dynamic allocation macros.

DFSMSdfp callable services

User programs written in assembler language and in high-level languages can call the DFSMSdfp callable services.

IGWABWO
: Retrieves or sets data set indicators. For example, your program can determine if a data set can be backed up while it is open for update.

IGWARLS
Obtains information about the record-level sharing attributes for a VSAM data set.

IGWASMS
Determines if a data set is system-managed, and returns the SMS class names and data set type.

IGWASYS
Determines the version, release, and modification level of DFSMS and the status of the SMS subsystem.

IGWLSHR
Determines the DFSMSdfp share attributes currently in use on the system.

Data management utility programs

No specific set of characteristics define what constitutes a z/OS utility program today, but common usage includes only a limited number of z/OS-provided programs as utilities. The UNIX community, by contrast, considers many of the standard commands as utilities, including compilers, backup programs, filters, and many other types of programs. To the z/OS community these are applications or programs, not utilities.

z/OS utilities are usually submitted as batch programs that have similar JCL requirements, including four specific data definition (DD) statements:

- The `SYSPRINT DD` statement tells the system where to print the informational or error messages from the utility program.
- The `SYSUT1 DD` statement identifies the data set that the utility is to use for input.
- The `SYSUT2 DD` statement identifies the data set that the utility is to use for output.
- The `SYSIN DD` statement contains utility control statements, which identify a particular function to be performed by a utility program and, when required, to identify specific volumes or data sets to be processed. As an alternative to using the input stream (the SYSIN DD statement), you may place utility control statements in a sequential data set, in a member of a partitioned data set or PDSE, or in a z/OS UNIX System Services (z/OS UNIX) file such as a HFS file.

Although utilities are usually run as batch jobs, using ALLOC commands in the TSO foreground is an alternative to using JCL.

Considering the wide-ranging functions and abilities of z/OS, only a small number of system-provided utilities exist. Most z/OS users are familiar with the utilities IEFBR14, IEBGENER, and IEBCOPY. VSAM users must be familiar with IDCAMS, which is the program name for the access method services utility.

A large number of customer-written utility programs also exist– although most users refrain from naming them utilities– and many of these are widely shared by the user community. Independent software vendors also provide many similar products (for a fee). Some of these programs or products can be categorized as utilities; of these, some compete with IBM utilities, while many others provide functions not included with the IBM-provided utilities.

Guide to z/OS utility program functions

Although z/OS utilities provide functions that are better performed by newer applications, many customers continue to use these programs, and IBM continues

to ship them for compatibility with older supported system levels. This guide lists the tasks for which you can use these utilities, and identifies which utility is especially suited to perform each task.

You can use the DFSMS utility programs to perform a variety of tasks, as shown in Table 2. The "Task" column shows tasks that you might want to perform. The "Options" column more specifically defines the tasks. The "Primary Utility" column identifies the utility that is especially suited for the task. The "Secondary Utilities" column identifies other utilities that can be used to perform the task.

Table 2. Tasks and Utility Programs

Task	Options	Primary Utility	Secondary Utilities
Add	a member to a partitioned data set	IEBUPDATE, IEBGENER	IEBDG
	a password	IEHPROGM	
Alter in place	a load module	IEBCOPY	
Catalog	a data set in a catalog	IEHPROGM	
Change	data set organization	IEBUPDTE	IEBGENER, IEBPTPCH
	logical record length	IEBGENER	
Clear	checkpointed members from a PDSE	IEBCOPY	
Compare	z/OS UNIX System Services (z/OS UNIX) files such as HFS files	IEBCOMPR	
	partitioned data sets	IEBCOMPR	
	sequential data sets	IEBCOMPR	
	PDSEs	IEBCOMPR	
Compress	a partitioned data set	IEBCOPY	
Compress in place	a partitioned data set	IEBCOPY	
Convert to partitioned data set	an unloaded PDSE containing program objects cannot be loaded into a PDS. An unloaded PDSE containing data objects can be loaded into a PDS but all extended attributes will be lost.	IEBCOPY	
	sequential data sets	IEBGENER	IEBUPDTE
	a PDSE	IEBCOPY	
Convert to PDSE	a partitioned data set	IEBCOPY	
	an unloaded copy of a partitioned data set or PDSE	IEBCOPY	
	sequential data sets	IEBGENER	IEBUPDTE
Convert to sequential data set	a partitioned data set or PDSE	IEBGENER	IEBUPDTE

Table 2. Tasks and Utility Programs (continued)

Task	Options	Primary Utility	Secondary Utilities
Copy	a load module or load module library	IEBCOPY	
	a partitioned data set	IEBCOPY	IEHMOVE
	a volume of data sets (on tape or disk)	IEHMOVE	
	job steps	IEBEDIT	
	selected members of a partitioned data set	IEBCOPY	IEHMOVE
	sequential data sets	IEBGENER	IEHMOVE, IEBUPDTE, IEBPTPCH
	a PDSE	IEBCOPY	
	a group of PDSE members	IEBCOPY	
	selected members of a PDSE	IEBCOPY	
Create	a backup copy of a partitioned data set or PDSE	IEBCOPY	
	a character arrangement table module	IEBIMAGE	
	a copy modification module	IEBIMAGE	
	a 3800 or 4248 forms control buffer module	IEBIMAGE	
	a graphic character modification module	IEBIMAGE	
	a library character set module	IEBIMAGE	
	a library of partitioned members	IEBGENER	IEBUPDTE
	a member of a partitioned data set or PDSE	IEBGENER	IEBDG, IEBUPDTE
	a sequential output data set	IEBDG	IEBGENER, IEBPTPCH
	an indexed sequential data set	IEBDG	
	an output job stream	IEBEDIT	
Delete	a data set or member of a partitioned data set	IEHPROGM	
	password	IEHPROGM	
	catalog entries	IEHPROGM	
	records in a partitioned data set or PDSE member	IEBUPDTE	
Edit and convert to partitioned data set or PDSE	a sequential data set	IEBGENER	IEBUPDTE
Edit and copy	a job stream	IEBEDIT	
	a sequential data set	IEBGENER	IEBUPDTE, IEBPTPCH
Edit and list	error statistics by volume (ESV) records	IFHSTATR	
Edit and print	a sequential data set	IEBPTPCH	IEBGENER
Edit and punch	a sequential data set	IEBPTPCH	IEBGENER
Enter	a procedure into a procedure library	IEBUPDTE	
Exclude	a partitioned data set member from a copy operation	IEBCOPY	IEHMOVE
	a PDSE member from a copy operation	IEBCOPY	
Expand	a partitioned data set or PDSE	IEBCOPY	
	a sequential data set	IEBGENER	
Generate	test data	IEBDG	

Table 2. Tasks and Utility Programs (continued)

Task	Options	Primary Utility	Secondary Utilities
Include	changes to members or sequential data sets	IEBUPDTE	
	a partitioned data set member from a copy operation	IEBCOPY	IEHMOVE
	a PDSE member from a copy operation	IEBCOPY	
Indicate	double-byte character set string by supplying enclosing shift-out/shift-in characters	IEBGENER	IEBPTPCH
Insert records	into a partitioned data set or PDSE	IEBUPDTE	
Label	magnetic tape volumes	IEHINITT	
List	a password entry	IEHPROGM	
	a volume table of contents	IEHLIST	
	number of unused directory blocks and tracks	IEHLIST	IEBCOPY
	partitioned data set or PDSE directories	IEHLIST	IEHPROGM
	CVOL entries	IEHLIST	
Load	an unloaded partitioned data set to a partitioned data set	IEBCOPY	
	an unloaded data set	IEHMOVE	
	an unloaded partitioned data set to a PDSE (for non-load modules only)	IEBCOPY	
	an unloaded PDSE to a partitioned data set (for non-load modules only)	IEBCOPY	
	an unloaded PDSE to a PDSE	IEBCOPY	
Merge	partitioned data sets	IEBCOPY	IEHMOVE
	PDSEs	IEBCOPY	
	partitioned data sets and PDSEs	IEBCOPY	
Modify	a partitioned or sequential data set, or a PDSE	IEBUPDTE	
Move	a volume of data sets	IEHMOVE	
	partitioned data sets	IEHMOVE	
	sequential data sets	IEHMOVE	
Number records	in a new or old member of a partitioned data set or PDSE	IEBUPDTE	
Password protection	add a password	IEHPROGM	
	delete a password	IEHPROGM	
	list passwords	IEHPROGM	
	replace a password	IEHPROGM	
Print	sequential data sets	IEBPTPCH	IEBGENER, IEBUPDTE
	partitioned data sets or PDSEs	IEBPTPCH	
	selected records	IEBPTPCH	
	mixed strings of double-byte and single-byte character set data	IEBPTPCH	IEBGENER
	double-byte character set data	IEBPTPCH	IEBGENER

Table 2. Tasks and Utility Programs (continued)

Task	Options	Primary Utility	Secondary Utilities
Punch	a partitioned data set member	IEBPTPCH	
	a sequential data set	IEBPTPCH	
	selected records	IEBPTPCH	
	mixed strings of double-byte and single-byte character set data	IEBPTPCH	IEBGENER
	Double-byte character set data	IEBPTPCH	IEBGENER
Reblock	a load module	IEBCOPY	
	a partitioned data set or PDSE	IEBCOPY	
	a sequential data set	IEBGENER	IEBUPDTE
Re-create	a partitioned data set or PDSE	IEBCOPY	
Rename	member of a partitioned data set or PDSE	IEBCOPY	IEHPROGM
	a sequential or partitioned data set, or PDSE	IEHPROGM	
	moved or copied members of a partitioned data set	IEHMOVE	
Renumber	logical records	IEBUPDTE	
Remove	indication of a double-byte character set string by stripping off enclosing shift-out/shift-in characters	IEBGENER	
Replace	a password	IEHPROGM	
	logical records	IEBUPDTE	
	records in a member of a partitioned data set or PDSE	IEBUPDTE	
	selected members of a PDSE	IEBCOPY	IEBUPDTE
	selected members of a partitioned data set	IEBCOPY	IEBUPDTE, IEHMOVE
Scratch	data sets	IEHPROGM	
Uncatalog	data sets	IEHPROGM	
Unload	a partitioned data set	IEBCOPY	IEHMOVE
	a sequential data set	IEHMOVE	
	a PDSE	IEBCOPY	
Update in place	a partitioned data set or PDSE	IEBUPDTE	

Data set utilities

Data set utilities reorganize, change, or compare data at the data set or record level. These programs are widely used in batch jobs.

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. These data set utilities cannot be used with VSAM data sets.

IDCAMS: Use access method services for catalogs

Although it provides other functions, IDCAMS, which is the program name for access method services, is used primarily to define and manage VSAM data sets and integrated catalog facility catalogs.

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, system-provided programs (or macros) to define data sets, and utility programs to process data sets. VSAM (Virtual Sequential Access Method) is an access method used for more complex applications. VSAM arranges records by an index key, relative record number, or relative byte addressing.

Some users pronounce the name of this program as “id-cams” (two syllables) while others say “I-D-cams” (three syllables).

A typical example of a simple use of IDCAMS is as follows:

```
//VDFNDEL  JOB 1,LINDAJO,MSGCLASS=X
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD *
//DATAIN   DD DISP=OLD,DSN=LINDA.SORTOUT
//SYSIN    DD *
 DEFINE CLUSTER (NAME (LINDA.DATA.VSAM) -
 VOLUMES(WORK02) CYLINDERS(1 1) -
 RECORDSIZE (72 100) KEYS(9 8) INDEXED)
 REPRO INFILE(DATAIN) OUTDATASET(LINDA.DATA.VSAM) ELIMIT(200)
/*
//STEP2    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE LINDA.DATA.VSAM CLUSTER
/*
```

This sample job, named VDFNDEL, consists of two steps: One to define a VSAM data set; the other to delete it. The first step, STEP1 performs two functions:

1. Creates a VSAM data set through the DEFINE CLUSTER command. Note that IDCAMS uses dynamic allocation to create the necessary JCL for this new data set, so the sample does not include a DD statement for the new data set.
   - The DEFINE CLUSTER command is continued over three records; the continuation indicators are hyphens.
   - The VSAM data set is on volume WORK02, and uses one cylinder for primary space and one cylinder for secondary allocation. The average record size is 72 bytes and the maximum record size is 100 bytes. (VSAM data sets always use variable length records.) The primary key (for accessing records in the data set) is 8 bytes long and begins at an offset of 9 bytes into each record.
   - Records for loading a VSAM data set this way should already be sorted into key order.
   - The ELIMIT parameter specifies the number of error records that REPRO will ignore before terminating operation. An error record is usually due to a duplicate key value.
2. Loads the new data set through the REPRO command. The input loaded into the new data set comes from a sequential data set, which is identified through the DATAIN DD statement.

The second step, STEP2, deletes the data set that is created STEP1.

Many of IDCAMS functions can be entered as TSO commands. For example, DEFINE CLUSTER can be used as a TSO command. However, using IDCAMS in this manner is generally not recommended because these commands can be complex and the errors encountered can be complex. Entering the IDCAMS commands through a batch job allows the commands and resulting messages to be reviewed as often as necessary by using SDSF to view the output.

The IEBCOPY utility: Copy libraries (partitioned data sets)

IEBCOPY is a data set utility that is used to copy or merge members between one or more partitioned data sets, or partitioned data sets extended (PDSEs), in full or in part.

More specifically, this utility is commonly used for several purposes:

- To copy selected (or all) members from one partitioned data set to another.
- To copy a partitioned data set into a unique sequential format known as an unloaded partitioned data set. As a sequential data set it can be written on tape, sent by FTP, or manipulated as a simple sequential data set.
- To read an unloaded partitioned data set (which is a sequential file) and recreate the original partitioned data set. Optionally, only selected members might be used.
- To compress partitioned data sets (in place) to recover lost space.

Most z/OS software products are distributed as unloaded partitioned data sets. The ISPF copy options (option 3.3, among others) uses IEBCOPY "under the covers." Moving a PDS or PDSE from one volume to another is easily done with IEBCOPY. If there is a need to manipulate partitioned data sets in batch jobs, IEBCOPY is probably used. Equivalent manipulation under TSO (using ISPF) uses IEBCOPY indirectly.

A simple IEBCOPY job might be the following:

```
//COPYJOB5 JOB 1,WAYNE,MSGCLASS=X
//         EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DISP=SHR,DSN=WAYNE.LIB.SOURCE
//SYSUT2   DD DISP=(NEW,KEEP),UNIT=TAPE,DSN=WAYNES.SOURCE,
//   VOL=SER=123456
```

This job will unload WAYNE.LIB.SOURCE (which we assume is a partitioned data set) and write it on tape. (The name TAPE is assumed to be an esoteric name that the local installation associates with tape drives.) By default, IEBCOPY copies from the data set defined by the SYSUT1 DD statement to the data set defined by SYSUT2 DD. For most utilities, SYSUT1 is the name used for the DD statement that defines the input data set, and SYSUT2 for the DD statement that defines the output data set. Notice that the data set name on tape is not the same as the data set name used as input (the same name could be used, but there is no requirement to do so).

The following job could be used to restore the PDS on another volume:

```
//COPYJOB6 JOB 1,WAYNE,MSGCLASS=X
//         EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DISP=OLD,UNIT=TAPE,DSN=WAYNES.SOURCE,
//   VOL=SER=123456
//SYSUT2 DD DISP=(NEW,CATLG),DSN=P390Z.LIB.PGMS,UNIT=3390,
//   SPACE=(TRK,(10,10,20)),VOL=SER=333333
```

In this example, IEBCOPY will detect that the input data set is an unloaded partitioned data set, which we determined would fit in about 10 tracks and should have 20 directory blocks.

Instead of using the DUMMY parameter on the SYSIN DD statement, you could substitute this JCL:

```
//SYSIN DD *
  COPY OUTDD=SYSUT2,INDD=SYSUT1
  SELECT MEMBER=(PGM1,PGM2)
/*
```

The SELECT statement specifies the member names to be processed, with the OUTDD and INDD parameters specifying the DD names to be used for output and input, respectively. You would have to use this JCL if you used names other than SYSUT1 and SYSUT2 for the input and output DD statements.

Restoring a partitioned data set from an unloaded copy automatically compresses (recovers lost space) the data set.

The IEBDG utility: Generate test data

The IEBDG utility provides a pattern of test data to be used as a programming debugging aid. This pattern of data can then be analyzed quickly for predictable results.

You can either use one of the IBM-supplied patterns or you can specify your own pattern of data to be generated in a variety of fields in a data set. The fields can be changed for each record with ripple, wave, shift, roll, and other field permutations. IEBDG can accept input data records and overlay specified fields in the input with generated data.

The following is a simple example of IEGDB use:

```
//GENDATA7 JOB 1,CHRIS,MSGCLASS=X
//         EXEC PGM=IEBDG
//SYSPRINT DD SYSOUT=*
//OUTDS    DD DISP=(NEW,CATLG),DSN=CHRIS.TEST.DATA,UNIT=3390,
//   VOL=SER=WORK01,SPACE=(CYL,(10,1)),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
//SYSIN    DD *
  DSD OUTPUT=(OUT)
  FD NAME=FIELD1,LENGTH=30,FORMAT=AL,ACTION=RP
  FD NAME=FIELD2,LENGTH=10,PICTURE=10,'TEST DATA '
  FD NAME=FIELD3,LENGTH=10,FORMAT=RA
  CREATE QUANTITY=90000,NAME=(FIELD1,FIELD2,FIELD3)
  END
/*
```

This job creates a new data set, CHRIS.TEST.DATA, with 90,000 records. Each record is 80 bytes, as specified in the DCB parameters in the DD statement. The control statements specify three fields that occupy the first 50 bytes of each record. By default, IEBDG fills the remaining bytes with binary zeros. The three fields are:

- An alphabetic field ('ABCDEF...'), 30 bytes long. It is rippled (rotated left one byte) after each record is generated.
- The second field contains 10 bytes with the fixed constant 'TEST DATA '.
- The third field contains 10 bytes with random binary data.

The utility can generate more complex patterns, but this example is typical of simple usage. It also illustrates an estimate of the amount of disk space needed for data:

- A 3390 track holds about 57 K, less whatever space is lost to inter-record gaps.
- The data control block (DCB) parameters, which relate to the program that will use the test data, specify:
  - A logical record length (LRECL) of 80
  - A block size (BLKSIZE) of 8000, and

– A record format (RECFM) of fixed block (FB)

The space requirements (SPACE) for the data set are based on the following assumptions:

- Approximately six blocks of 8000 each will probably fit on one track.
- Each block contains 100 records of 80 bytes each, and each track contains 600 records.
- A cylinder contains 15 tracks, therefore a cylinder will hold 9000 of these records.
- Given these assumptions, 10 cylinders are needed to hold 90,000 records. So 10 cylinders is the primary space allocation specified in the JCL, with one cylinder as the secondary allocation increment. The secondary allocation is used only if the primary allocation is not sufficient space.

The IEBGENER utility: Generate (copy) a sequential data set

The IEBGENER utility is a copy program that has been part of the operating system since the first release of OS/360. One of its many uses is to copy a sequential data set, a member of a partitioned data set (PDS) or PDSE, or a z/OS UNIX System Services (z/OS UNIX) file such as a HFS file.

IEBGENER also can filter data; change a data set's logical record length (LRECL) and block size (BLKSIZE); and generate records.

The most common use is to simply copy data sets. A typical job looks like this:

```
//SMITH2   JOB 1,GEOFF,MSGCLASS=X
//         EXEC PGM=IEBGENER
//SYSIN    DD DUMMY
//SYSPRINT DD SYSOUT=X
//SYSUT1   DD DISP=SHR,DSN=SMITH.SEQ.DATA
//SYSUT2   DD DISP=(NEW,CATLG),DSN=SMITH.COPY.DATA,UNIT=3390,
//   VOL=SER=WORK02,SPACE=(TRK,3,3))
```

IEBGENER requires four data definition (DD) statements with the DD names shown in the example:

- The SYSIN DD statement is used to read control parameters; for simple uses, no control parameters are needed and a DD DUMMY can be used.
- The SYSPRINT statement is for messages from IEBGENER.
- The SYSUT1 statement is for input and the SYSUT2 statement is for output. This example reads an existing data set and copies it to a new data set.

The IEBUPDTE utility: Update data sets with fixed-length records

The IEBUPDTE utility creates multiple members in a partitioned data set, or updates records within a member. While it can be used for other types of records, its main use is to create or maintain JCL procedure libraries or assembler macro libraries.

Today, this utility is used mostly for z/OS licensed program distributions and maintenance. It is seldom used by TSO users.

This basic example uses IEBUPDTE to add two JCL procedures to the data set named MY.PROCLIB:

```
//ADDPROC1 JOB 1,SMCHUGH,MSGCLASS=X
//         EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,DSN=MY.PROCLIB
//SYSUT2   DD DISP=OLD,DSN=MY.PROCLIB
```

```
//SYSIN    DD DATA
./ ADD LIST=ALL,NAME=MYJOB1
//STEP1 EXEC=SUZNX1
//PRINT DD SYSOUT=A
// (more JCL for MYJOB1)
//SYSUDUMP DD SYSOUT=* (last JCL for MYJOB1)
./ REPL LIST=ALL,NAME=LASTJOB
//LIST EXEC PGM=SUZNLIST
// (more JCL for this procedure)
//* LAST JCL STATEMENT FOR LASTJOB
./ ENDUP
/*
```

This example requires a few comments:

- When a library is to be updated, then SYSUT1 and SYSUT2 both point to that library. (If they point to different libraries, the SYSUT1 library is copied to the SYSUT2 library and then updated.)
- The SYSIN DD DATA format indicates that the data in the input stream contains // (two slashes) in columns one and two. The information in the input stream should not be interpreted as JCL. The end of the input stream is indicated by /*.
- The IEBUPDTE utility uses control statements with the symbols ./ (a period and slash) in the first two columns.
- A member named MYJOB1 is added to MY.PROCLIB; this member should not already exist in the library.
- A member named LASTJOB is replaced with new contents.

The IEBUPDTE utility also can add or replace statements in a member based on the sequence numbers in the statements. This capability is one of the few remaining uses for sequence numbers in JCL or source statements.

Again, IEBUPDTE is typically used for program distribution and maintenance. For example, if a software vendor's product adds 25 JCL procedures to a customer's procedure library, the vendor might package the procedures as an IEBUPDTE job. One advantage is that all the material is in source format and the customer can easily review the contents before running the job.

The IEFBR14 utility: Do (almost) nothing

The utility program IEFBR14 performs no action other than return a completion code of 0; however, "running" this utility invokes other system components that perform useful tasks.

For example, submitting JCL to run IEFBR14 causes the z/OS job scheduler to check your JCL statements for syntax errors. If your JCL contains data definition (DD) statements, the z/OS initiator will allocate space for new data sets and perform disposition processing for all data sets.

The following JCL for the job named SMITH1 accomplishes several tasks, even though IEFBR14 does nothing but return 0:

```
//SMITH1 JOB 1,LEO,MSGCLASS=X
//       EXEC PGM=IEFBR14
//NEWDS  DD DSN=SMITH.LIB.CNTL,DISP=(NEW,CATLG),VOL=SER=WORK02,
//   UNIT=3390,SPACE=(CYL,(3,1,25)
//OLDDS  DD DSN=SMITH.OLD.DATA,DISP=(OLD,DELETE)
```

- The z/OS job scheduler to check your JCL statements for syntax errors.
- The initiator allocates the new data set defined by NEWDS (SMITH.LIB.CNTL) and keeps the data set when the job ends.

- The initiator also deletes an old data set defined by OLDDS (SMITH.OLD.DATA) at the end of the job.

The same functions to create one data set and delete another could be done through ISPF, for example, but these actions might be needed as part of a larger sequence of batch jobs.

Note: This explanation of the name IEFBR14 might help you remember what this utility does... One IBM group writing early OS/360 code used the prefix "IEF" for all their code modules. In assembly language, "BR" means Branch to the address in a register. Branching to the address in general register 14 is the standard way to end a program.

System utilities

System utility programs are used to list or change information that is 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.

The ICKDSF utility: Install and manage DASD volumes

The ICKDSF utility performs functions needed for the installation, use, and maintenance of IBM direct-access storage devices (DASD). You also can use it to perform service functions, error detection, and media maintenance.

The ICKDSF utility is used primarily to initialize disk volumes. At a minimum, this process involves creating the disk label record and the volume table of contents (VTOC). ICKDSF also can scan a volume to ensure that it is usable, can reformat all the tracks, can write home addresses, as well as other functions.

You can use the ICKDSF utility through two methods:

- Execute ICKDSF as a job or job step using job control language (JCL). ICKDSF commands are then entered as part of the SYSIN data for z/OS.
- Use Interactive Storage Management Facility (ISMF) panels to schedule ICKDSF jobs.

The IEHINITT utility: Initialize tape devices

IEHINITT is a system utility used to place standard volume label sets onto any number of magnetic tapes mounted on one or more tape units. Many larger z/OS installations do not allow unlabeled tapes to be brought into the installation.

The tape labels can be ISO/ANSI Version 3 or ISO/ANSI Version 4 volume label sets written in ASCII (American Standard Code for Information Interchange) or IBM standard labels written in EBCDIC.

IEHINITT is an APF-authorized program, which means that if another program calls IEHINITT, that program must also be APF-authorized. To protect system integrity, the calling program must follow the sytem integrity requirements described in *z/OS MVS™ Programming: Authorized Assembler Services Guide*.

Because IEHINITT can overwrite previously labeled tapes regardless of expiration date and security protection, IBM recommends that the security administrator use PROGRAM protection as noted in *z/OS DFSMSdfp Utilities*.

The IEHLIST utility: List system data

IEHLIST is a system utility used to list 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 (VTOC).

IEHLIST is not used often in most installations, but is needed in the rare cases where a VTOC is corrupted. It is sometimes used with the SUPERZAP program to patch or fix a broken VTOC.

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.

IEHLIST can be used to list, partially or completely, entries in a specified volume table of contents (VTOC), whether indexed or non-indexed. The program lists the contents of selected data set control blocks (DSCBs) in edited or unedited form.

The IEHPROGM utility: Manage catalogs and data sets

The IEHPROGM utility is used primarily to manage catalogs, rename data sets, and delete data sets. Most of the IEHPROGM functions are available through access method services (the IDCAMS utility), which is now the preferred utility for catalog and data set functions.

The IEHPROGM utility offers a programmatic alternative to using job control language to perform functions required during system installation or the installation of a major program product. These functions may involve dozens (or hundreds) of such catalog and data set actions. Having commands prepared beforehand (in a batch job with IEHPROGM) is much less error-prone than more dynamic approaches.

You can use IEHPROGM to perform the following tasks:

- Scratch (delete) a data set or a member of a partitioned data set.
- Rename a data set or a member of a partitioned data set.
- Maintain data set passwords.

You must have RACF® authority to use IEHPROGM.

SPZAP (a.k.a. Superzap): Dynamically update programs or data

The SPZAP service aid has several aliases, including AMASPZAP and Superzap, the latter being the most commonly used. SPZAP allows you to patch or fix volume table of contents (VTOCs), executable programs, or almost any other disk record. In practice, it is used most frequently to patch executable programs.

The functions of SPZAP provide many capabilities, including:

- Using the inspect and modify functions of SPZAP, you can fix programming errors that require only the replacement of instructions in a load module member of a PDS or a program object member of a PDSE without recompiling the program.
- Using the modify function of SPZAP, you can set traps in a program by inserting incorrect instructions. The incorrect instructions will force abnormal

ending; the dump of storage provided as a result of the abnormal ending is a valuable diagnostic tool, because it shows the contents of storage at a predictable point during processing.

- Using SPZAP to replace data directly on a direct access device, you can reconstruct VTOCs or data records that may have been destroyed as the result of an I/O error or a programming error.

As an example of the first function, modifying a programming error, suppose your company has just purchased a new release of product XXX. The new release may have been sent on tape to hundreds or thousands of customers. After shipping all these tapes the product developers may have discovered a minor bug that could be fixed by changing a few instructions. Instead of creating new distribution tapes and shipping them to all the customers (a massive and expensive undertaking for a major software product), the developers could create an SPZAP solution and mail, fax, or ftp it to their customers.

The SPZAP solution might look something like this:

```
//AMYS15   JOB  1,AMYS,MSGCLASS=X
//STEP1    EXEC PGM=AMASPZAP
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD DISP=OLD,DSN=LANDER.LIB.LOAD
//SYSIN    DD *
 NAME QSAM1
 VERIFY 004E 4780
 REP 004E 4700
/*
```

In the example:

- The EXEC statement identifies SPZAP as the program to run, using Superzap's program name (AMASPZAP).
- The SYSLIB DD statement points to the data set containing the load module to be modified.
- The SYSIN DD statement contains the control statements that tell SPZAP what actions to perform:
  - The NAME control statement identifies the executable module (which is the PDS member name) to be altered.
  - The VERIFY statement says to look at offset x'004E' in the module and verify that it contains x'4780'.
  - If the verify is correct then change the module to contain x'4700' at this same offset. This action changes a Branch Equal instruction to a No Operation and changes the logic of the program.

An SPZAP patch like this is easily constructed when you have an assembly listing of the program and can see the exact offset within the module containing the instruction you want to change. Creating a patch is more difficult without a listing, although it can be accomplished by reading hexadecimal storage dumps and reconstructing machine language operation from the dumps. Note that the format of executable programs on disk is complex and is not a simple image of the program when it is loaded into memory. (Relocation data, external symbols, and an optimized disk loading format form part of the complexity.) SPZAP understands this disk format and allows users to zap an executable program as if it were a memory image.

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 grant 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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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 illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This book documents information that is NOT intended to be used as Programming Interfaces of z/OS.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

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

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

Printed in USA