z/OS

DFSMSdss Storage Administration Guide

SC35-0423-06

z/OS

DFSMSdss Storage Administration Guide

SC35-0423-06

Note
Before using this information and the product it supports, be sure to read the general information under “Notices” on page 221.

Eighth Edition, April 2006

This edition applies to Version 1 Release 7 (Refresh Version) of z/OS® (5694-A01), Version 1 Release 7 (Refresh Version) of z/OS.e (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC35-0423-05.

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

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

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

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

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

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

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

Contents

Figures

Tables

About This Document

This document describes how to use the DFSMSdss™ component of DFSMS to perform various storage management tasks. It is intended primarily for storage administrators and system programmers.

This guide also includes diagnostic information intended to help you diagnose DFSMSdss problems. Before you begin diagnostic procedures follow these steps to verify that the problem is not the result of incorrect command usage:

1. Use the *z/OS DFSMSdss Storage Administration Reference* and verify that all of the parameters specified for each command are used correctly
2. Correct any errors you may have found and resubmit the JCL
3. Use this document to build a set of keywords that describes the error and, if all parameters appear to be correctly specified, contact IBM for assistance.

Related reading:

- For information about the syntax of DFSMSdss commands, see the *z/OS DFSMSdss Storage Administration Reference.*
- For information about DFSMSdss messages, see *z/OS MVS System Messages, Vol 1 (ABA-AOM).*

For information about the accessibility features of z/OS®, for users who have a physical disability, see Appendix E, "Accessibility," on page 219.

Required product knowledge

To use this manual you need some knowledge of DFSMSdfp™, DFSMShsm™, RACF®; (a component of the Security Server for z/OS), and Job Control Language (JCL).

Referenced documents

The following publications are referenced in this book:

Publication Title	Order Number
z/OS DFSMS Introduction	SC26-7397
z/OS DFSMSdss Storage Administration Reference	SC35-0424
z/OS DFSMS Installation Exits	SC26-7396
z/OS DFSMS Using the Interactive Storage Management Facility	SC26-7411
z/OS MVS System Messages, Vol 1 (ABA-AOM)	SA22-7631
z/OS DFSMSdfp Storage Administration Reference	SC26-7402
z/OS DFSMS Implementing System-Managed Storage	SC26-7407
MVS/ESA SML: Managing Data	SC26-3124
z/OS Security Server RACF Security Administrator's Guide	SA22-7683
z/OS DFSMShsm Storage Administration Guide	SC35-0421
z/OS DFSMS Managing Catalogs	SC26-7409
z/OS DFSMS Macro Instructions for Data Sets	SC26-7408

Publication Title	Order Number
z/OS MVS Planning: Global Resource Serialization	SA22-7600
CICS and VSAM Record Level Sharing: Implementation Guide	SG24-4766

Accessing z/OS DFSMS documents on the Internet

In addition to making softcopy documents available on CD-ROM, IBM provides access to unlicensed z/OS softcopy documents on the Internet. To view, search, and print z/OS documents, go to the z/OS Internet Library:

`www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Using LookAt to look up message explanations

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

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

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

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

Summary of Changes

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

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

Summary of Changes for SC35-0423-06 z/OS Version 1 Release 7

This document contains information previously presented in *z/OS Version 1 Release 7 DFSMSdss Storage Administration Guide* (SC35-0423-05).

The following sections summarize the changes to that information.

New Information

This edition includes information about the following features:

- New section added for "Backing Up and Restoring Volumes with Incremental FlashCopy" on page 48.
- New section added for "Backing Up Volumes with FlashCopy Consistency Group" on page 120.

Changed Information

This edition includes information about the following features:

- Updated section for "Freeing Subsystem Resources" on page 99.
- Updated section for "Consolidating Free Space on Volumes" on page 141.

Summary of Changes for SC35-0423-05 z/OS Version 1 Release 7

This document contains information previously presented in *z/OS Version 1 Release 6 DFSMSdss Storage Administration Guide* (SC35-0423-04).

The following sections summarize the changes to that information.

New Information

This edition includes information about the following features:

- DFSMSdss now supports large format data sets in z/OS V1R7. PTFs will be available for down level systems to prevent DFSMSdss from attempting to process large format data sets on systems which are not capable of processing them correctly.
- New to this book is the diagnostic information which is included in chapters 10, 11, and 12 and appendices C and D. This information has been moved here from the DFSMSdss Diagnostic Guide which is discontinued and will not be published for z/OS V1R7 or later releases.

Changed Information

This edition includes information about the following features:

- DFSMSdss no longer supports Indexed Sequential data sets.

Deleted Information

This edition deletes information about the following features:

- Most references to Indexed Sequential data sets have been removed
- References to IEBISAM have been removed
- The STEPCAT DD and JOBCAT DD statements are no longer allowed in z/OS v1r7. Users of DFSMSdss will no longer be able to request DFSMSdss to process data sets that are cataloged in a catalog where the catalog is outside of the standard order of search.

Summary of Changes for SC35-0423-04 z/OS Version 1 Release 6

This document contains information previously presented in *z/OS Version 1 Release 5 DFSMSdss Storage Administration Guide* (SC35-0423-03).

The following sections summarize the changes to that information.

New Information

This edition includes information about the following features:

- This document has been enabled for the following z/OS Library Center advanced searches: Examples.

Changed Information

This edition includes information about the following features:

- Changed various examples to update references from OS/390 R10 and OS/390 2.10

Chapter 1. Introduction to the DFSMSdss Component of DFSMS

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 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 storage management subsystem (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

Stand-Alone Restore Program of DFSMSdss

The DFSMSdss stand-alone restore function is a single-purpose program. It is designed to allow the system programmer to restore vital system packs during disaster recovery without relying on an MVS environment.

You can restore the following from a physical dump:

- A full volume or ranges of tracks
- Your system residence (SYSRES) volume, if your operating system fails to IPL

Related reading: For additional information about the DFSMSdss Stand-Alone Services function, see the *z/OS DFSMSdss Storage Administration Reference*.

Understanding the Role of DFSMSdss

The role that DFSMSdss plays at your site depends on what other DFSMS components you use. The way you use DFSMSdss depends upon whether you also use the DFSMShsm component. To understand the role of DFSMSdss in an SMS environment, you need a basic understanding of SMS.

Managing User Data with SMS

SMS allows you to match users' data characteristics (like data set organization, size, and format) to the characteristics of storage devices, without requiring users to know or to understand the hardware configuration at your site. With SMS, end users can store and retrieve data without being aware of space limitations, device characteristics, or volume serial numbers.

Using SMS, you can define allocation management criteria for the different types of data at your site. The values you specify identify your users' requirements for

space, availability, and performance. You define these values to SMS as:

Data Class	A named list of data set allocation attributes that SMS assigns to a data set when it is created.
Storage Class	A named list of data set storage service attributes that identify performance and availability requirements. SMS uses these attributes to control data placement.
Management Class	A named list of management attributes that SMS uses to control DFSMShsm action for data set retention, migration, backup, and release of allocated but unused space.
Storage Group	A named list of DASD volumes used for allocation of new SMS-managed data sets or for a dummy storage group.

Automatic class selection (ACS) is the SMS mechanism for assigning SMS classes and storage groups (also known as constructs). Depending on the DFSMSdss command you are using, SMS invokes some or all of the ACS routines in the following order:

1. Storage class ACS routine
2. Management class ACS routine
3. Storage group ACS routine

SMS uses the assigned constructs to automatically place and manage data and storage. For example, you can use a storage class to keep performance-sensitive data on high-speed storage devices and use management classes to control the movement of less active data to tape.

If there are WRITE statements in the SMS ACS routines, these are only displayed in the DFSMSdss output when the ACS routines return a nonzero return code. If DFSMSdss processes a data set successfully, then no WRITE messages are displayed.

Related reading: For a additional information about SMS and how to use it, see the ACS routine information found in *z/OS DFSMSdfp Storage Administration Reference*.

Sequential Data Striping

Extended-format sequential data sets and extended-format VSAM data sets, which must reside on SMS volumes, can be striped. Striping is a subtype of the basic record organizations: sequential and VSAM. With striping, the data is written across multiple volumes, with consecutive “loading units” being striped (applied) to different volumes. The “loading unit” for extended-format sequential data sets is a track. The “loading unit” for striped extended-format VSAM data sets is a control interval (CI).

Striping can reduce the processing time for batch jobs that process large data sets sequentially.

DFSMSdss can dump, restore, copy, or release space from a striped data set.

Notes:

1. No new keywords or commands are required to support striped data sets.

2. DFSMSdss treats a striped data set in the same way as it does other multivolume SMS data sets.
3. DFSMSdss can convert a striped extended-format VSAM data set to extended-format during RESTORE processing. DFSMSdss can convert an extended-format sequential data set to sequential during RESTORE or COPY processing.

Related reading: For additional information about how striped data sets are processed, see *z/OS DFSMS Implementing System-Managed Storage*.

Record Counting

DFSMSdss provides a means for verifying results of certain operations:

- **Sequential extended-data sets**—DFSMSdss performs and reports byte counting for logical data set COPY, DUMP, and RESTORE operations. The byte counts are reported in message ADR902I for copy, ADR903I for dump, and ADR906I for restore.
- **Indexed VSAM data sets**—DFSMSdss performs and reports record counting for logical data set DUMP and RESTORE operations if the VALIDATE support is used during the dump processing. VALIDATE processing is the default for dump.

  During dump processing, the record count is reported in message ADR788I. In restore processing, message ADR788I is issued if the restore record count matches the dump count. Message ADR789W is issued if the dump and restore record counts differ and both the dump and restore record counts are provided.

Related reading: For additional information about the messages, see *z/OS MVS System Messages, Vol 1 (ABA-AOM)*.

Installation Exit Routines

You can customize DFSMSdss by coding exit routines. The following installation exit routines are supplied with DFSMSdss:

Authorization installation exit routine (ADRUPSWD)
Forces authorization checking of protected data sets

Enqueue installation exit routine (ADRUENQ)
Forces enqueuing of the volume table of contents (VTOC)

Options installation exit routine (ADRUIXIT)
Can override any default or user-specified command options in the input stream

Reblock installation exit routine (ADRREBLK)
Allows DFSMSdss, during a data set copy or data set restore operation, to use the block size it selects for the target data set

Related reading: For additional information about these exit routines, see *z/OS DFSMS Installation Exits*.

Authorization Checking

Related reading: For information on authorization checking, see the *z/OS DFSMSdss Storage Administration Reference*.

Managing Availability with DFSMSdss

DFSMSdss availability management consists of backing up data on DASD to tape and restoring from the backup if the original is lost, damaged, or inadvertently changed.

There are two general forms of backup:

Data set backup	Protects against the loss of individual data sets
Volume backup	Protects against the loss of a volume

For data set backup, you can perform incremental backups to help reduce processing time while still meeting your backup requirements. Incremental backup means that data sets are backed up only if they have changed since their last backup.

Volume backups are used to protect against media failure. You can use volume backups in conjunction with incremental data set backups to recover a volume. As a result, you need not do volume backups as frequently. Incremental data set backups should be done daily and volume backups weekly. If a volume is lost for some reason, you can restore from the most recent volume backup and apply incremental data set backups to the volume to bring it back to its most current status.

Backing Up and Restoring Volumes and Data Sets

You use the DFSMSdss DUMP command to back up volumes and data sets, and you use the RESTORE command to recover them. You can make incremental backups of your data sets by specifying a data set DUMP command with RESET and filtering on the data-set-changed flag.

In an SMS environment, DFSMSdss saves the class names for the data sets it dumps. When you restore the data set to an SMS-managed volume, DFSMSdss invokes ACS and passes it the class names saved with the data set. Based upon this and other input from DFSMSdss (for example, class names specified with the STORCLAS or MGMTCLAS keywords), ACS assigns SMS constructs to each data set.

Because DFSMSdss restore invokes ACS, you can restore the data sets to SMS-managed volumes. Conversely, data sets backed up as SMS-managed data sets can be restored as non-SMS-managed data sets.

In addition to providing for routine backup requirements, you can use DFSMSdss to back up application data for disaster recovery and vital records purposes. You can back up all the data sets (including data that resides only on primary DASD; you cannot use DFSMSdss to process migrated data sets) that are associated with a particular application for disaster recovery or vital records by using DFSMSdss logical data set dump, and filtering on data set names. If you do not want to perform a separate dump operation for disaster recovery, you can specify more than one OUTDDNAME to create up to 255 separate backup copies when you do your routine backup. These extra copies can then be used for disaster recovery or vital records purposes. The DUMP command can also be used to archive data sets that have not been accessed for long periods of time.

Using DFSMShsm for Backup

The DFSMShsm component of DFSMS provides automated incremental backup, interactive recovery, and inventory of what it backs up. If DFSMShsm is being used, you should use DFSMSdss for volume backup of data sets not supported by DFSMShsm and for dumping SYSRES and special volumes such as the one containing the master catalog. If DFSMShsm is not installed, you can use DFSMSdss for all volume and data set backups.

Using Concurrent Copy

Many online data bases must be available at all times. If a backup is made while the data is being updated, the backup could be unusable or could require that a log be applied to the restored version to synchronize the data. The alternative is to synchronize all parts of the data base and stop all update activity during the backup.

The concurrent copy (CC) function of DFSMSdss is a hardware and software solution that allows you to back up a data base or any collection of data at a point in time and with minimum down time for the data base. The data base is unavailable only long enough for DFSMSdss to initialize a CC session for the data, which is a very small fraction of the time that the complete backup will take. The copy that is made will not include any of the update activity; it will be as if the backup were made instantaneously when it was requested. After initialization, DFSMSdss releases all the serialization it holds on the data, informs the user that the initialization is complete so that update activity may resume, and begins reading the data.

Be aware, however, that CC does not remove all data integrity exposures. For example, a DFSMSdss full-volume dump serializes the VTOC of the source volume, but does not serialize the data sets on the volume. This ensures that the existing data sets are not deleted or extended, and new data sets are not allocated. However, there is an exposure in that the data in the existing data sets can be changed. Without CC, this exposure exists for the entire duration of the dump. With CC, the exposure exists only during initialization.

Notes:

1. If you are using CC on VM-format volumes, DFSMSdss does not serialize VM data in any way.
2. VM minivolumes are supported if you are using RAMAC®® Virtual Array (RVA) devices to the extent that they are supported by IBM Extended Facilities Product (IXFP) device reporting.

If a dump requestor does not stop all updating of the data sets during the CC session initialization, the backup data integrity is compromised.

If a CC operation fails after signaling that the CC initialization was complete (and update activity on the data has resumed), it is not possible to recover the data at the point-in-time at which the CC operation was started. This is because the data may have been updated while the copy operation was progressing.

Virtual Concurrent Copy

Virtual concurrent copy support uses SnapShot to provide a concurrent copy-like function when the source device supports SnapShot, but does not support concurrent copy.

During virtual concurrent copy, data is “snapped” from the source location to an intermediate location, and the data is gradually copied to the target location using

normal I/O methods. The operation is logically complete after the source data is "snapped" to the intermediate location and physically complete after the data is moved to the target media.

Managing Data Movement with DFSMSdss

DFSMSdss can help you move data to replace devices, add capacity, and meet performance requirements. The three general types of data movement are data set, volume, and track movement.

Related reading: For additional information about moving data, see Chapter 6, "Managing Availability with DFSMSdss," on page 29.

Moving Data

Using the DFSMSdss COPY command, you can perform data set, volume, and track movement. The COPY command with DELETE causes DFSMSdss to delete the source data set after it successfully copies the data set.

The full-volume COPY command is useful for moving data between like devices. If you are moving volumes to like devices of larger capacity, generally you need a larger VTOC because the larger device can hold more data sets. DFSMSdss rebuilds indexed VTOCs and recognizes larger VTOCs on target volumes (as long as the target VTOC is outside the range of the source volume) when the VTOCs are moved to a like device of larger capacity.

For moving data between unlike devices, you must use the logical data set COPY command for all the data sets on the volume. DFSMSdss fills the tracks as completely as possible instead of just copying track for track. In addition, if the reblockable indicator is set on, the data set is reblocked to a system-determined block size efficient for the device.

Moving Data in an SMS-Managed Environment

In an SMS-managed environment, ACS routines and VTOC/Data Set Services (VDSS) determine the target volume in an SMS-managed environment.

DFSMSdss moves data sets to different volumes if their storage groups change. However, even if their storage groups do not change, DFSMSdss might move the data sets to a different location on the same volume or to different volumes. Target volumes selected by the user may not be honored.

If a new, empty volume is added to a storage group, data sets moved into that storage group are likely to be placed on that volume.

If a data set's storage class has the guaranteed-space attribute and the user specifies output volumes, the data set is placed on the SMS volumes specified in the volume list if:

- All SMS-managed volumes specified with the OUTDDNAME or OUTDYNAM keyword belong to the same storage group.
- The ACS storage group routine assigns the data set to the storage group that contains the specified SMS volumes.

Note: SMS-managed data sets must be cataloged in the standard order of search.

For a more complete discussion of SMS and how to use it, refer to *z/OS DFSMSdfp Storage Administration Reference*.

Moving Data with Concurrent Copy

Concurrent copy and virtual concurrent copy can be used during copy as well as dump.

Related reading: For additional information about SMS and how to use it, see the *z/OS DFSMSdfp Storage Administration Reference*.

Moving Data with FlashCopy

FlashCopy® is faster than traditional methods of data movement, especially for moving large amounts of data. DFSMSdss can use the FlashCopy feature of the Enterprise Storage Server® (ESS) to quickly move the data from the source location to the target location. The source devices and the target devices must be in the same ESS, and the data to be moved must not need manipulation.

Related reading: For additional information about moving data using FlashCopy, see Chapter 10, “Managing Data Movement with DFSMSdss,” on page 89.

Moving Data with SnapShot

DFSMSdss can use SnapShot to quickly move the data from the source location to the target location. The source and target devices must be in the RAMAC Virtual Array (RVA) and the data must not need to be manipulated. SnapShot is much faster than traditional methods, especially when large amounts of data are moved.

Related reading: For additional information about moving data using SnapShot, see Chapter 10, “Managing Data Movement with DFSMSdss,” on page 89.

Converting Data to and from SMS Management

DFSMSdss is the primary tool for converting data to and from SMS management. There are two ways of converting data:

- Conversion of data sets with data movement
- Conversion of volumes without data movement

The following sections briefly describe these two kinds of conversion.

Converting Data Sets with Data Movement

To convert data sets by data movement, use the DFSMSdss COPY or DUMP/RESTORE command. When moving data sets from non-SMS-managed volumes to SMS-managed volumes, DFSMSdss invokes ACS, which may assign class names to the data sets. Alternatively, you can specify the BYPASSACS and STORCLAS keywords with the COPY or RESTORE command to force the data sets to be SMS-managed.

When moving data sets out of SMS management, specify the BYPASSACS and NULLSTORCLAS keywords with the COPY or RESTORE command. DFSMSdss then bypasses ACS and drops the data set’s class names. ACS can also make data sets non-SMS-managed.

Related reading: For additional information about converting data sets, see Chapter 11, “Converting Data to and from SMS Management,” on page 127.

Converting Volumes without Data Movement

To convert volumes to and from SMS management without data movement, you can use the DFSMSdss CONVERTV command. This command lets you:

- **Prepare a volume for conversion.** Using the PREPARE keyword, you can stop new allocations and data set extensions to another volume while still allowing access to the data on the volume.
- **Convert a volume to SMS management.** Using the SMS keyword, you can convert a volume and all its data sets to SMS management.
- **Convert a volume from SMS management.** Using the NONSMS keyword, you can remove a volume and its data sets from SMS management.
- **Simulate conversion.** Using the TEST keyword, you can verify that the volume and its data sets are eligible for conversion and see what class names ACS would assign to the data sets.

Related reading: For additional information about converting volumes, see Chapter 11, "Converting Data to and from SMS Management," on page 127.

Managing Space with DFSMSdss

DFSMSdss has four functions to help you manage your DASD space:

COMPRESS Compresses your partitioned data sets by taking unused space and consolidating it at the end of the data set. To make the unused space available for other data sets, you must use the RELEASE command. This does not apply to PDSEs.

RELEASE Releases the unused space in sequential, partitioned, and extended-format VSAM data sets for use by other data sets.

DEFRAG Consolidates the free space on a volume to help prevent out-of-space abends on new allocations.

DUMP/RESTORE
Deletes unwanted data sets and combines data set extents. (The COPY command can also be used to combine data set extents.)

Chapter 2. Requirements for Running DFSMSdss

This chapter describes the requirements for running DFSMSdss.

Understanding the Operating Environment

DFSMSdss is exclusive to z/OS and is available only as a component of z/OS.

You can use the Stand-Alone restore program of DFSMSdss outside a system environment. Alternatively, you can run the Stand-Alone restore program on an IBM System/370™; (S/370™™) in MVS/ESA™; mode, MVS/XA™; mode, and S/370 mode. It can also be run on an IBM System/390®; in S/390®; mode or S/370 mode. The available modes are dependent on your CPU type and model. Additionally, the Stand-Alone restore program can be run in a virtual machine under VM in either 370 mode or XA mode.

Storage Requirements

In most cases, you can let DFSMSdss determine the amount of storage it uses for an operation. Sometimes, however, you might want closer control of the amount of storage DFSMSdss uses. Use the storage estimates in this section as a starting point for determining minimum region sizes in which DFSMSdss can run. Table 1 and Table 2 on page 10 show the minimum storage requirements, in bytes, to run each DFSMSdss operation. Table 3 on page 11 shows the minimum storage requirements, in bytes, to restore partitioned and VSAM data sets to unlike devices. The legend for Table 1 on page 10, Table 2 on page 10, and Table 3 on page 11 is displayed beneath Table 3 on page 11. The values include the storage required to load the DFSMSdss program into the region.

Storage requirements depend on your operating system configuration and your device and data set characteristics. The storage requirement estimates shown for the COPY, DUMP, and RESTORE commands are only for the full-volume copy, dump, and restore operations; they might vary for a data set operation.

If DFSMSdss determines that the storage requirements are greater than the storage available during processing, DFSMSdss issues error message ADR376E to indicate this. The out-of-storage condition might cause abend 80A during DFSMSdss postprocessing.

If you use buffers above 16 megabytes virtual storage, the buffer size is allocated independently of the region size.

If you use the PARALLEL command to run two or more DFSMSdss tasks concurrently, the total storage required is the sum of the storage required for all functions to be run in parallel. However, because DFSMSdss is reentrant, the DFSMSdss code is not duplicated in storage. Therefore, do not include the DFSMSdss load module size more than once.

Table 1. Minimum Storage Requirements for DFSMSdss Operations with I/O Buffers below 16-Megabyte Virtual

DFSMSdss Command	Storage Requirements
COMPRESS	dsssize + (2 * trksize of largest device) + (number of additional volumes, up to five * copysize)
CONVERTV	dsssize + (buffersize * 5)
COPY (FULL)	dsssize + (trksize * 5)
COPYDUMP	dsssize + (buffersize * 5)
DEFRAG	dsssize + (trksize * 5) + (16KB if VSAM present) + (1KB * number of non-VSAM entries in VVDS)
DUMP (FULL) OPT(1)	dsssize + (buffersize * 7)
DUMP (FULL) OPT(2)	dsssize + (buffersize * 6)
DUMP (FULL) OPT(3)	dsssize + (buffersize * 15)
DUMP (FULL) OPT(4)	dsssize + (buffersize * (3 * trk/cyl))
PRINT	dsssize + (3 * trksize)
RELEASE	dsssize + trksize
RESTORE (FULL)	dsssize + copysize + (buffersize * 6)

Table 2. Minimum Storage Requirements for DFSMSdss Operations with I/O Buffers above 16-Megabyte Virtual

DFSMSdss Command	Storage Below 16MB Virtual	Storage Above 16MB Virtual
COMPRESS	dsssize + (number of additional volumes, up to five * copysize)	2 * trksize of largest device
CONVERTV	dsssize	buffersize * 5
COPY (FULL)	dsssize	trksize * 5
COPYDUMP	dsssize	buffersize * 5
DEFRAG	dsssize + (16KB if VSAM present) + (1KB * number of non-VSAM entries in VVDS)	(trksize * 5) + (152 * number of data set extents) + (144 * number of VSAM components) + (48 * number of non-VSAM data sets)
DUMP (FULL) OPT(1)	dsssize	buffersize * 7
DUMP (FULL) OPT(2)	dsssize	buffersize * 6
DUMP (FULL) OPT(3)	dsssize	buffersize * 15
DUMP (FULL) OPT(4)	dsssize	buffersize * (3 * trk/cyl)
PRINT	dsssize	3 * trksize
RELEASE	dsssize	trksize
RESTORE (FULL)	dsssize	buffersize * 6

Table 3. Minimum Storage Requirements for a Restore to an Unlike Device

Type of Data Set	Storage Requirements
Partitioned Data Sets	dsssize + (((trksize + 64) * 5) + 8KB)
VSAM Data Sets	dsssize + (((trksize + 64) * 5) + (2 * maximum record size) + (3 * buffspace))

Legend: This legend applies to Table 1 on page 10, Table 2 on page 10, and Table 3.

KB 1024 bytes.

dsssize DFSMSdss load module size, 1750KB.

copysize IEBCOPY load module size + IEBCOPY storage requirements below the 16MB line (that is, 1MB minimum, or 2MB if the data set being compressed has more than 1000 members).

buffersize 64KB for output to tape unless DFSMSdss is forced to a smaller blocksize. The maximum of the largest trksize used and a blocksize of 32KB.

buffspace Buffer space specified in the DEFINE command when the data set was allocated.

trksize The size, in bytes, of a track on your DASD volume.

trks/cyl The number of tracks per cylinder on your DASD volume.

Hardware Requirements

You can use DFSMSdss with all IBM DASD, magnetic tape devices, system consoles, printers, and card readers that are supported by DFSMS.

Notes:

1. VSAM-extended addressability requires a cached storage subsystem that has concurrent copy-capable licensed internal code.
2. DFSMSdss does not support virtual input/output (VIO) devices.
3. When running DFSMSdss operations against storage devices that do not support 64–bit real addressing, you must tell DFSMSdss not to use the I/O buffers that can be backed above the 2–gigabyte bar. This can be done by either specifying ZBUFF64R=OFF on the EXEC statement in your JCL or by turning off the UFPZB64R bit in ADRUFO block with the Options Installation Exit Routine, ADRUIXT.

Volume Formats

You can use DFSMSdss with the following DASD volume formats:

- Volumes with indexed VTOCs
- Volumes with nonindexed VTOCs
- OS/VS minivolumes in a VM environment
- VM-formatted volumes (full or mini) with an OS-compatible VTOC beginning on track zero, record five.

All DASD volumes used by DFSMSdss must be initialized by Device Support Facilities (ICKDSF) and be mounted and online.

Note: You cannot use concurrent copy on minivolumes of any format unless they are on an RVA. However, you can use concurrent copy on full VM-format

volumes that contain minivolumes to the extent that they are supported by IBM Extended Facilities Product (IXFP) device reporting.

Data Set Organizations

DFSMSdss can copy, dump, and restore data sets of the following types:

- DATABASE 2™ (DB2®;)
- Direct access
- EXCP (execute channel program)
- Partitioned, including:
  - PDS (partitioned data set)
  - PDSE (partitioned data set extended)
  - HFS (hierarchical file system) data set
- Sequential, including extended-format data sets and Large Format data sets
- VSAM data sets that are cataloged in an ICF catalog, including:
  - ESDS (entry-sequenced data set)
  - KSDS (key-sequenced data set)
  - KSDS with key ranges
  - LDS (linear data set)
  - RRDS (relative record data set)
  - VRRDS (variable relative record data set)
  - Extended-format ESDS, KSDS, LDS, RRDS, and VRRDS, including striped ESDS, KSDS, LDS, RRDS, and VRRDS
  - Extended-addressable VSAM ESDS, KSDS, LDS, RRDS, and VRRDS, including striped ESDS, KSDS, LDS, RRDS, and VRRDS
  - zFS (zSeries® file system) data set
- Unmovable data set types (PSU, POU, DAU, ABSTR, ISU, and direct with OPTCD=A).

Notes:

1. DFSMSdss does not provide conversion between non-extended-format VSAM and extended-format VSAM.
2. DFSMSdss cannot be used to process migrated data sets.
3. DFSMSdss cannot be used to process VSAM data sets that are cataloged in a catalog that is outside of the standard order of search.

Temporary Data Set Names

DFSMSdss must allocate the following temporary data sets to perform certain functions such as copy and restore. The high-level qualifiers of those data set names can be protected, and your installation must ensure that these temporary data sets can be allocated.

Message data set—
Allocated by DFSMSdss to store messages. This data set lets DFSMSdss print out messages by task rather than intermixing them. This data set is deleted when DFSMSdss completes the operation. System-generated temporary names are used.

Special DEFRAG data set—
Allocated by DFSMSdss to contain information about the DASD extents that are being moved. The data set name is in the following format:

```
SYS1.DFDSS.DEFRAG.xxxxxxxx.volser.DUMMY
```

where *xxxxxxxx* represents 8 bytes of X'FF', and *volser* is the volume serial number of the volume being defragmented. The data set is deleted when

the DEFRAG operation completes successfully. If the DEFRAG operation is interrupted (for example, when DFSMSdss is canceled), this data set is left on the volume. You must perform the following operations:

- You must run a new DEFRAG operation.
- You might need to convert an index VTOC (IXFORMAT) volume to non-indexed (OSFORMAT) before rerunning the DEFRAG operation. Otherwise, the volume free-space values may be incorrect.
- You need to use the hexadecimal qualifier to prevent the deletion of this data.

Temporary copied data sets

Allocated by DFSMSdss when a copy is performed and deleted when the copy is completed.

The format of the temporary name depends on the number of qualifiers of the data set being copied:

Number of qualifiers (*n*)	Temporary name
1	dsnhlq.Atidasid.*chmmsstt*
2	First 2 qualifiers.Atidasid.*chmmsstt*
>2	First 3 qualifiers.Atidasid.*chmmsstt*

The next to last qualifier, Atidasid, is a combination of a fixed ″A″ character followed by a task id (tid) and an address space id (asid).

The last qualifier is chmmsstt where *c* is:

T Target cluster name
D Target data component name
I Target index component name
U Source cluster name
E Source data component name
J Source index component name
P Source path name
Q Target path name

and *hmmsstt* is the time stamp information in low-order hours digits (*h*), minutes (*mm*), seconds (*ss*), and hundredths of a second (*tt*).

Note: In the course of copying data sets, DFSMSdss renames the source data set using the above conventions. Whenever DFSMSdss renames a data set that is protected by RACF, a component of the Security Server for z/OS, to a temporary name a RACF profile must exist for the temporary data set name.

Temporary copied catalogs

Allocated by DFSMSdss when it copies a catalog. When DFSMSdss copies a catalog, two temporary data sets are used.

First, DFSMSdss allocates a temporary data set into which records are temporarily exported. The export data set name format is:

CATHLQ.EXPORT.T*hmmsstt*

where,

CATHLQ The first three high-level qualifiers of the catalog that is being copied.

hmmsstt The time-stamp information in low-order hours digits (*h*), minutes (*mm*), seconds (*ss*), and hundredths of a second (*tt*).

Second, DFSMSdss allocates a temporary catalog. The temporary catalog name format is:

`CATHLQ.T`*`hmmsstt`*

where,

CATHLQ The first four high-level qualifiers of the catalog being copied.

hmmsstt The time-stamp information in low-order hours digits (*h*), minutes (*mm*), seconds (*ss*), and hundredths of a second (*tt*).

Dummy data set—

Allocated by DFSMSdss when copying or restoring volumes and an indexed-VTOC needs rebuilding or the volume free-space values need recalculating. The data set name is in the following format:

`SYS1.VTOCIX.DSS.TEMP.`*`volser`*

where *volser* is the volume serial number of the restored volume. Allocation of this data set is never successful because DFSMSdss uses dummy allocation values.

Chapter 3. Logical and Physical Processing and Data Set Filtering

Before you begin using DFSMSdss, you should understand the difference between logical and physical processing and how to use data set filtering to select data sets for processing. The following sections describe these two aspects of DFSMSdss.

Defining Logical 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.

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 operations.

DFSMSdss performs logical processing if:

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

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

Note: To copy or dump entire multivolume data sets, you do not need to specify all the volumes in the LOGINDDNAME or LOGINDYNAM volume list. However, you must specify the SELECTMULTI keyword with either the FIRST or ANY subkeywords.

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 was 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 cataloged as part of DFSMSdss processing.
- Data sets are deleted from the source volume after a successful dump or copy operation.
- Non-VSAM and VSAM data sets are renamed after a successful copy or restore operation.
- You want to control the percentage of space allocated on each of the output volumes for copy and restore operations.
- You want to copy and convert a PDS to a PDSE or vice versa.
- You want to copy or restore a data set with an undefined DSORG to an unlike device.
- You want to keep together all parts of a VSAM sphere.

Physical Processing

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

Notes:

1. VSAM data sets are *not* cataloged during physical processing within SMS or non-SMS environments. The CATALOG keyword is ignored for VSAM data sets during physical restore. Use IDCAMS DEFINE RECATALOG to catalog the data sets after the physical restore.
2. The RENAME and RENAMEUNCONDITIONAL keywords are ignored for VSAM data sets during physical restore.

DFSMSdss performs physical processing when the following conditions exist:

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

  Attention: Take care 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 DATASET 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 the following conditions exist:

- 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.

Data Integrity Considerations

In some circumstances, DFSMSdss can detect and correct inconsistencies while processing data. For example, DFSMSdss checks to verify the reliability of a partitioned data set (PDS) directory before it uses the PDS. You can also use the CHECKVTOC keyword to instruct DFSMSdss to perform additional consistency checking on the VTOC before data processing begins on that volume.

If you are creating backups as part of disaster recovery preparedness, you may want to take additional steps to ensure the validity of that data. You can establish validity before you invoke DFSMSdss, or as part of the DFSMSdss invocation.

Note: Periodically running the Access Method Services DIAGNOSE function to reorganize VSAM data sets establishes validity before you invoke DFSMSdss. However, *not* specifying the NOPACK keyword establishes validity as part of the DFSMSdss invocation. In this case, DFSMSdss verifies the PDS directory. If you specify the CHECKVTOC keyword, DFSMSdss performs consistency checking on the VTOC.

The choice between logical and physical processing depends on the expected type of abnormal condition (if any). Neither processing mode provides a significantly higher level of data integrity. Logical processing and physical processing are

simply different views of the same data. One mode could detect a condition that the other mode would miss. For example, a frequent PDS abnormal condition that does not cause problems during physical processing might cause problems during logical processing. On average, the selected DFSMSdss processing mode should closely mirror the mode in which you typically access data. Generally, logical processing is the most applicable choice.

Broken Data Set Considerations

Broken data sets are data sets that do not comply with defined IBM data set standards. These include data sets for which catalog entries, VTOC entries, or VSAM volume data set (VVDS) entries are either missing or invalid. DFSMSdss may not properly select broken data sets for processing because DFSMSdss relies on the validity of these structures during filtering.

Choosing Data Sets for Processing—Filtering

You can select data sets for DFSMSdss processing by filtering on specified criteria. DFSMSdss can filter on fully qualified or partially qualified data set names (by using the INCLUDE or EXCLUDE keyword) and on various data set characteristics (by using the BY keyword).

You can filter data sets with any of the following commands:

- Logical dump
- Logical restore
- Physical data set dump
- Physical data set restore
- Data set copy
- COMPRESS
- RELEASE

At least one of the INCLUDE, EXCLUDE, or BY parameters must be specified with the above commands.

Note: DFSMSdss cannot serialize all of the data sets being considered during filter processing. It is possible that, between the time when DFSMSdss does the filtering and builds the list of data sets to process and the time when DFSMSdss actually processes the data sets, some or all of the data sets may be moved, deleted, or migrated. The status of the moved, deleted, or migrated data sets will therefore have changed by the time they are processed, which may in turn cause the DFSMSdss operation to fail.

The following sections briefly describe what can be filtered and how to use the available criteria.

Filtering by Data Set Names

Using the INCLUDE or EXCLUDE keyword, you can filter on fully qualified or partially qualified data set names. A fully qualified data set name is one in which all qualifiers are completely spelled out. For example:

```
(INCLUDE(SYS1.UTIL3.LOAD))
```

A partially qualified data set name is one in which the qualifiers are not completely spelled out. Using asterisks (*) and percent signs (%), you can select data sets without specifying their fully qualified names.

The single asterisk (*) is used in place of one qualifier. For example:

```
(INCLUDE(ABC.*.LOAD))
```

This partially qualified name matches ABC.DEF.LOAD and ABC.XYZ.LOAD. The single * is also used to indicate that only part of a qualifier has been specified. For example, if you want to filter using only the first three characters of the first qualifier of a name, specify it as follows:

```
(INCLUDE(SYS*.**))
```

This partially qualified name matches data sets whose first qualifier was SYS1 and SYS1A. The other qualifiers in the data set name are ignored.

When used with other qualifiers, the double asterisk (**) indicates that one or more leading, trailing, or middle qualifiers do not exist or they do not play a role in the selection process. For example:

```
(INCLUDE(**.LOAD))
```

This partially qualified name selects any data set with LOAD as its last qualifier (such as data sets named LOAD, ABC.LOAD, and ABC.DEF.LOAD).

The percent sign (%) is used as an ignore character. Each % sign represents one character in the name being filtered, and any character in that position is ignored. One or more % signs can be specified in any qualifier. For example:

```
(INCLUDE(SYS1.A%%B))
```

This partially qualified name matches SYS1.AZZB and SYS1.AXYB, but not SYS1.AXXXB.

Filtering by Data Set Characteristics

The BY parameter can filter for the following data set characteristics:

Keyword	Criteria
ALLOC	Allocation type (cylinder, track, block, absolute track, or movable)
CATLG	Whether a data set is cataloged or not (using the standard catalog search order)
CREDT	Creation date (absolute or relative)
DATACLAS	Data class for SMS
DSCHA	Whether the data-set-changed flag is on or off
DSORG	Data set organization (SAM, PAM, PDS, PDSE, BDAM, EXCP, HFS, ISAM, VSAM, or zFS)
EXPDT	Expiration date (absolute or relative)
EXTNT	Number of extents

FSIZE	Data set size (number of allocated or used tracks)
MGMTCLAS	Management class for SMS
MULTI	Whether the VTOC shows that the data set is single-volume or multivolume (allocated single-volume data sets that have never been opened and are not cataloged may be selected as multivolume).
REFDT	Last-referenced date (absolute or relative)
STORCLAS	Storage class for SMS

You can use any of the following operators with the BY keyword:

Operator	**Meaning**
EQ or **=**	Equal to
LT or **<**	Less than
LE or **<=**	Less than or equal to
GT or **>**	Greater than
GE or **>=**	Greater than or equal to
NE or **¬=**	Not equal to

When you specify multiple arguments for an NE operation, DFSMSdss selects only those data sets not matching any of the arguments. When you specify multiple arguments for an EQ operation, DFSMSdss selects those data sets matching any of the arguments.

Some Examples of Filtering by Data Set Characteristics

If you use the following specification of the BY keyword, DFSMSdss selects all data sets allocated in cylinders:

```
BY( ALLOC,EQ,CYL )
```

You can specify more than one criterion with the BY keyword. The following example selects all data sets allocated in cylinders and whose management class is MCNAME1:

```
BY(( ALLOC,EQ,CYL ) ( MGMTCLAS,EQ,MCNAME1 ))
```

You can specify multiple arguments for any of the filtering criteria. The following example selects all data sets that have a data class of DCNAME1 or DCNAME2:

```
BY( DATACLAS,EQ,(DCNAME1,DCNAME2) )
```

The FILTERDD Keyword

The FILTERDD keyword must be used if you have more than 255 entries in the INCLUDE, EXCLUDE, or BY filtering lists. The FILTERDD keyword specifies the name of the DD statement that identifies the sequential data set or member of a

partitioned data set that contains the filtering criteria to be used. This is in the form of card-image records, in DFSMSdss command syntax, that contain the INCLUDE, EXCLUDE, and BY keywords.

Uses of Filtering

You will make the best use of filtering by data set names if you use meaningful naming conventions. Your naming conventions should allow you to identify large groups of data sets that can be treated similarly. With such conventions, you can use data set name filtering to select large groups of data sets against which you can run DFSMSdss functions.

Suppose you are a storage administrator and you want to do a daily backup of all payroll data sets that have changed since they were last backed up. If the data sets you want to back up have some identifying qualifiers (for example, PAYROLL.FEDTAX), you can select them by coding:

```
//VRPAY     JOB  Accounting Information,MORGAN
//STEP1     EXEC PGM=ADRDSSU,REGION=4000K
//SYSPRINT  DD   SYSOUT=*
//DROUT     DD   DSN=PAYROLL.DAY1,DISP=(NEW,CATLG),UNIT=3480,LABEL=(1,SL)
//SYSIN     DD   *
     DUMP DATASET(INCLUDE(PAYROLL.FEDTAX.**)  -
                  BY((DSCHA,EQ,YES) (MGMTCLAS,EQ,DAILY)))  -
     OUTDD(DROUT)
/*
```

Filtering by data set characteristics also lets you process large groups of data sets. You can use BY criteria to:

- Filter on the data-set-changed-flag to back up only those data sets that have not been backed up since they were last updated.
- Filter to select uncataloged data sets for deletion as a means of enforcing cataloging.
- Filter to select data sets whose expiration date passed for deletion.
- Filter on the last referenced date to archive or delete data sets that have not been referenced for a long period of time (for example, 18 months).
- Filter on data set size to ensure that when you use the COMPRESS and RELEASE commands, you compress and release space only in data sets where the savings may be significant.
- Filter on management class to perform space management (if in an SMS-managed environment).

It is possible to pass DFSMSdss filtering criteria in a data set by using the FILTERDD keyword. If you do this, the data set should have the following characteristics:

- RECFM=F or FB
- LRECL=80
- BLKSIZE=80 for F (or a multiple of 80 for FB).

Related reading: For additional information about setting up data set naming conventions, see *MVS/ESA SML: Managing Data.*

Chapter 4. Invoking DFSMSdss

You can use the following methods to invoke DFSMSdss:
- Interactive Storage Management Facility (ISMF)
- Job control language (JCL)
- The application interface

Invoking DFSMSdss with ISMF

You can use the menu-driven panels of ISMF to build job streams for many DFSMSdss space management and backup functions. ISMF supports the DFSMSdss commands COMPRESS, CONVERTV, COPY, DEFRAG, DUMP, RELEASE, and RESTORE.

The information you supply on ISMF panels is used to build and submit job streams like those you generate using JCL and DFSMSdss commands. Using ISMF panels, you do not have to remember DFSMSdss keywords and syntax. Simply fill in the values you want on the panels, and ISMF generates the job stream. You can then either submit the job or save the job stream for later use.

Using ISMF panels, you can build a list of data sets or volumes according to criteria that you provide. The list provides information about each volume or data set (for example, allocated space and percent of unused space). You can use the list to analyze and manage your data and storage more efficiently.

How to Invoke ISMF

You invoke ISMF by logging on to TSO. If ISMF is installed as an option on the ISPF Master Application Menu or as an option on the ISPF/PDF Primary Option Menu, specify the selection option that corresponds to ISMF. You can use ISMF to perform DFSMSdss functions against one or more data sets or volumes on a list you create. Extensive help screens are available for all the DFSMSdss functions supported by ISMF. Refer to *z/OS DFSMS Using the Interactive Storage Management Facility* for more details.

Invoking DFSMSdss with JCL

DFSMSdss is controlled by JCL statements and DFSMSdss commands. You can use the JCL statements to invoke DFSMSdss and to define the data sets used and created by it. The JCL defines the DFSMSdss commands that specify and control tasks. Refer to the chapter about specifying DFSMSdss commands in *z/OS DFSMSdss Storage Administration Reference* for JCL information and examples.

Invoking DFSMSdss with the Application Interface

This section documents General-Use Programming Interface and Associated Guidance Information provided by DFSMSdss.

You can invoke DFSMSdss from an application program by using the application interface. This allows you, for example, to gather statistical or auditing information and to specify control variables.

The application interface allows you to:

- Fully utilize the invocation capabilities of DFSMSdss when the ATTACH, LINK, or CALL system macro is specified in your application program.
- Optionally, specify a list of parameters to be used by DFSMSdss during the processing caused by that invocation.
- Optionally, interact with DFSMSdss during processing of user installation options after the installation options exit has been called.
- Optionally, interact with DFSMSdss during the processing at convenient points where input/output (I/O) operations are being performed.

Note: DFSMSdss runs as an authorized problem program (nonsupervisor state); any program invoking DFSMSdss must also be authorized and in nonsupervisor state.

For more information on the application interface, refer to *z/OS DFSMSdss Storage Administration Reference*.

User Interaction Module Exit Functions

When DFSMSdss is invoked from an application program, you can use the user interaction module (UIM) to interact with DFSMSdss at points where I/O operations are being performed. UIM exit functions can be used to:

- Replace, insert, delete, or modify a SYSIN record after DFSMSdss has read it or a SYSPRINT record when DFSMSdss is ready to print it.
- Replace, insert, delete, or modify a write-to-operator message before DFSMSdss writes it.
- Insert a statistics record during a logical dump operation.
- Modify the installation options specified in the ADRUFO control block to override the specified options.
- Bypass password and expiration-date checking, or reject the tape volume and request a scratch tape, when DFSMSdss is ready to open a tape.
- Request a specific volume serial when a nonspecific tape is passed to DFSMSdss.
- Get information about the data set being allocated.
- End a task or processing of individual data sets.
- Bypass authority checking for individual data sets. This includes both RACF and password authorization.
- Bypass serialization checking of individual data sets.
- Show the status of the concurrent copy initialization.

For more information and examples of UIM exit functions, refer to *z/OS DFSMSdss Storage Administration Reference*.

Chapter 5. Protecting DFSMSdss Functions

This chapter discusses the functions of DFSMSdss to which you can control access by using the Resource Access Control Facility (RACF), an element of the z/OS Security Server. You can protect DFSMSdss/ISMF functions and some DFSMSdss keywords.

Protecting DFSMSdss/ISMF Functions with the z/OS Security Server RACF Element

You can set authorization levels for the following ISMF elements by using the program control feature of the z/OS Security Server RACF component:

- ISMF itself
- Each of the ISMF applications
- The individual line operators and commands

The RACF report process and logging process for each ISMF function that you identify also includes the RACF element for authorization checking. You can also use standard RACF authorization checking to limit access to individual data sets, volumes, or catalogs. Used in conjunction with program control, authorization checking ensures that the appropriate ISMF data and functions are available to users when they need them.

ISMF Functions You Might Want to Protect

Program control allows you to determine the ISMF functions to which users have access. The authorization scheme you set up can apply to both individual users and user groups. The ISMF functions you can protect fall into two general categories: line operators and commands.

With program control, you can set up authorization levels for each category. You can also vary the level within a category to suit the needs of your site. Before you set up an authorization structure, consider the following:

- Do you want all users at your site to have access to ISMF?
- Do you want all users to have access to the data set, volume, or profile applications?
- Are there line operators or commands to which you want to limit access?

Setting Up the Authorization Structure

RACF program control checks authorization before allowing access to an ISMF function. Protection for each function is based on the authorization level of the load module that contains the function. A user is allowed to execute an ISMF function (for example, the RESTORE list command) when one of the following is true:

- The user is authorized to execute the load module corresponding to the function requested. Authorization is defined as READ level access or greater.
- The user's RACF profile has the OPERATIONS attribute.
- The user's group is authorized to execute the load module.
- The universal access authority (UACC) for the load module is READ or greater. This makes the load module available to anyone who can access ISMF.

Finding the DFSMSdss/ISMF Module Names

Programming Interface information

The names of the load modules for DFSMSdss/ISMF are stored in command tables in both the panel library, DGTPLIB, and the load library, DGTLLIB. The load module names are listed in Table 4 and Table 5. The module names are found in the DGTSMMD1 member of the panel library.

Table 4 lists the names for the corresponding line operators. The module names for line operators are found in the DGTTLPD3 member of the load library. Table 5 lists the names for commands. These names are in the DGTTCTD2 member of the load library.

Table 4. Module Names for DFSMSdss/ISMF Line Operators

Line Operator	Data Set Application Module Name	Volume Application Module Name
& CGCREATE	—	DGTFCG01
COMPRESS	DGTFCM01	DGTFCS01
CONVERTV	—	DGTFCN01
COPY	DGTFCY01	DGTFCV01
DEFRAG	—	DGTFDF01
DUMP	DGTFDP01	DGTFDM01
RELEASE	DGTFRL01	DGTFRV01
RESTORE	DGTFRT01	DGTFRO01

Table 5. Module Names for DFSMSdss/ISMF Data Set Application Commands

Command	Module Name
COMPRESS	DGTFCP01
COPY	DGTFCO01
DUMP	DGTFDU01
RELEASE	DGTFRE01
RESTORE	DGTFRR00

To view the command table, you need to know the data set names that your site uses for the panel library and the load library. The installation of DFSMSdss/ISMF puts the panel library in SYS1.DGTPLIB and the load library in SYS1.DGTLLIB. However, your site's postinstallation procedures might involve moving the DFSMSdss/ISMF libraries. If they were moved, you can determine the data set name by issuing the TSO LISTALC command and scanning the low-level qualifiers for DGTPLIB and DGTLLIB.

End of Programming Interface information

Note: DFSMSdss does not have special support for name hiding. You can prevent the disclosure of names by DFSMSdss by moving DFSMSdss to a protected library that only authorized users can access.

Protecting DFSMSdss/ISMF Modules

The steps used to protect DFSMSdss/ISMF modules are listed below:

1. To define the modules you want to protect, use the RDEFINE command or the ISPF RACF entry panels. When you define the modules to RACF, supply the name of the load module you want to protect, the name of the data set that contains the module, and the volume serial number of the volume that contains the data set. Each module you identify is added to the profile for the PROGRAM general resource class. You have several options when you define modules:
   - If you want to define several modules at the same time, you can use asterisk notation. For example, DGT* means all the modules beginning with the letters DGT.
   - You can add an access list with user IDs, group names with their associated access authority to the profile, or both.
   - You can define the UACC to give default access to all users or to none.
   - You can use the AUDIT parameter to set up RACF logging or to bypass it.
2. To allow users to execute an application, line operator, or command, use the PERMIT command.

Related reading: For additional information about how to perform these steps and the options you have using program control, see the *z/OS Security Server RACF Security Administrator's Guide*.

Protecting DFSMSdss Keywords with RACF

In addition to protecting DFSMSdss/ISMF functions, you can also protect certain DFSMSdss keywords. This is done by defining facility class resource profiles and restricting access to those profiles. Table 6 lists these keywords and their associated RACF class profiles. For a given command or parameter, protection occurs when both of the following conditions are met:

- RACF facility class is active
- The indicated facility class profile has been defined

When facility class is active and one of the profiles that is listed below is defined, you must have READ access authority in order to use the indicated command or keyword. Otherwise, anybody can use the indicated command or keyword. If facility class checking is not set up for these keywords, any DFSMSdss user can use them.

Table 6. RACF Facility Class Profile Names for DFSMSdss Keywords

Keyword	Profile Name
BYPASSACS with COPY	STGADMIN.ADR.COPY.BYPASSACS
BYPASSACS with RESTORE	STGADMIN.ADR.RESTORE.BYPASSACS
& CGCREATE	STGADMIN.ADR.CGCREATE
CONCURRENT with COPY	STGADMIN.ADR.COPY.CNCURRNT
CONCURRENT with DUMP	STGADMIN.ADR.DUMP.CNCURRNT
CONVERTV	STGADMIN.ADR.CONVERTV
DEFRAG	STGADMIN.ADR.DEFRAG
DELETECATALOGENTRY with RESTORE	STGADMIN.ADR.RESTORE.DELCATE

Table 6. RACF Facility Class Profile Names for DFSMSdss Keywords (continued)

Keyword	Profile Name
& FCCGFREEZE with COPY	STGADMIN.ADR.COPY.FCFREEZE
FCTOPPRCPRIMARY with COPY	STGADMIN.ADR.COPY.FCTOPPRCP
FCTOPPRCPRIMARY with DEFRAG	STGADMIN.ADR.DEFRAG.FCTOPPRCP
IMPORT with RESTORE	STGADMIN.ADR.RESTORE.IMPORT
INCAT(catname) with COPY	STGADMIN.ADR.COPY.INCAT
INCAT(catname) with DUMP	STGADMIN.ADR.DUMP.INCAT
INCAT(catname) with RELEASE	STGADMIN.ADR.RELEASE.INCAT
PROCESS(SYS1) with COPY	STGADMIN.ADR.COPY.PROCESS.SYS
PROCESS(SYS1) with DUMP	STGADMIN.ADR.DUMP.PROCESS.SYS
PROCESS(SYS1) with RELEASE	STGADMIN.ADR.RELEASE.PROCESS.SYS
TOLERATE(ENQF) with COPY	STGADMIN.ADR.COPY.TOLERATE.ENQF
TOLERATE(ENQF) with DUMP	STGADMIN.ADR.DUMP.TOLERATE.ENQF
TOLERATE(ENQF) with RESTORE	STGADMIN.ADR.RESTORE.TOLERATE.ENQF

You can bypass this type of FACILITY class checking with the DFSMSdss installation options exit routine that your installation may be using.

Related Reading:

- For additional information about the installation options exit routine, see *z/OS DFSMS Installation Exits*.
- For additional information about RACF class profiles, see *z/OS Security Server RACF Security Administrator's Guide*.

Name-Hiding

DFSMSdss has no special support for the name-hiding function. Your installation is responsible for protecting DFSMSdss functions and resources from unauthorized users. You can use the existing procedures to limit the use of DFSMSdss function by authorized users. For example, you can prevent disclosing names by placing DFSMSdss in a protected library that only authorized users can use.

Related reading: For additional information how to protect a library, see "Protecting DFSMSdss/ISMF Modules" on page 27.

Chapter 6. Managing Availability with DFSMSdss

One of the major functions of DFSMSdss is the backup and recovery of data. Using the DUMP and RESTORE commands, you can backup and recover data sets and volumes. You can also use the DUMP and RESTORE commands on ranges of tracks. However, this is usually done as a means of diagnosing I/O errors rather than as a means of backing up and recovering data.

Planning an Availability Strategy

In planning your overall availability strategy, you should consider the following types of backup:

Backup of volumes and data sets—The general type of backup to guard against users accidentally losing or incorrectly changing their data sets and against losing volumes because of hardware failures.

Disaster recovery backup—Backup to protect against the loss of all your data in a major disaster at your site. These backups are stored off site and, in the event of a major disaster, are recovered at another site.

Vital records backup—Backup copies of data sets kept to meet externally imposed retention requirements, such as tax records.

Archival—Backup of data that is unused for a long period of time. You remove the data from DASD and retain it on tape in case it is needed again.

DFSMSdss is a flexible backup and recovery tool. You can use DFSMSdss by itself to perform all backups listed above or to complement other backup and recovery tools.

Backup and Recovery

General backup should be done at both the data set and the volume level. To protect against users accidentally deleting or changing their data sets, it is usually more efficient to do incremental backup (logical backup of those data sets that changed since they were last backed up). Incremental backups minimize processing time because you are not backing up every data set. Logical backup lets you restore data sets to unlike devices.

Data Set Backup

For data set backup, you need to consider the frequency of backup and the number of versions you want to keep. A number of factors can influence this decision, such as:

- The rate at which the data changes.
- The ease or difficulty of rebuilding the data (for example, it is easier to rebuild an object library than a source library).
- The importance of the data. For data that is extremely important to your business, you might want to keep extra backup versions.

A more complete discussion of the things you must consider when determining frequency of backup and number of versions can be found in *MVS/ESA SML: Managing Data*.

Volume Backup

Volume backup is necessary to guard against losing a volume, but it need not be done often if you are doing incremental backup on a regular basis. If you lose a

volume, you can recover from the latest volume backup, and then recover data sets from incremental backups to return the volume to its status before the failure. This form of recovery is sometimes referred to as forward recovery. In order to perform it, though, you must have a record of all of your backups. The DFSMShsm component keeps its own inventory of the data sets it backs up and can perform forward recovery using that inventory. DFSMSdss prints the names of the data sets it dumps and the serial number and data set sequence number of the tape volumes on which the dump begins and ends. You must use this printed record to perform forward recovery with DFSMSdss.

Backup and Recovery in an SMS-Managed Environment

Two kinds of data exist in an SMS-managed environment: SMS-managed and non-SMS-managed data. DFSMSdss can help you fulfill your availability requirements for both kinds of data.

SMS-Managed Data: The DFSMShsm component can perform automatic volume backup (by invoking DFSMSdss) and incremental backup on SMS-managed data. Each data set is assigned a management class that indicates how often DFSMShsm should back it up and how many versions of the backup to keep. Using DFSMShsm this way lets you manage availability at the data set level.

If you do not have the DFSMShsm component installed, you can use DFSMSdss to back up and recover data sets and volumes. By filtering on management class name and the data-set-changed flag, you can perform incremental backup on all the data sets belonging to a particular management class. To facilitate this backup procedure, you can set up a DFSMSdss job to run periodically.

For more details on planning for backup of SMS-managed data, refer to *MVS/ESA SML: Managing Data.*

Non-SMS-Managed Data: Typically, non-SMS-managed data is data that SMS does not support or data that is in transition from non-SMS to SMS management. If it is data that SMS does not support, you can probably still use DFSMSdss to back it up and recover it, because DFSMSdss supports many kinds of data that SMS does not. If it is data in transition to SMS management, you can use DFSMShsm or DFSMSdss to back up and recover it until it is placed under SMS management.

Backup and Recovery in a Non-SMS-Managed Environment

If SMS is not active, you are in a non-SMS-managed environment. For availability management, the data in this environment can be treated much the same as the non-SMS-managed data in an SMS-managed environment. DFSMSdss can be used to back up and recover it at the data set and volume level.

Disaster Recovery

Disaster recovery backups are made specifically for recovering data and applications following a disaster. Never rely on your regular backup data sets (for instance, DFSMShsm or DFSMSdss incremental backups) for disaster recovery. Disaster recovery backups require some very special considerations that normally do not apply to other types of backups.

Storing at a Remote Site

A basic difference between regular backups and disaster recovery backups is that disaster recovery backups must be transported to a different site. The remoteness of the recovery site depends upon the type of disaster you are preparing for (in the case of a fire, the recovery site can be around the corner; in the case of an

earthquake or flood, it should be many miles away). The fact that the backups must be taken to another site means that they must be on a portable media: tape.

Note: You can also automatically transmit backups to another site.

Using Logical Data Set Dump

Because the environment at the remote site might differ from your environment, you should ensure that your disaster recovery backups can be restored in a different environment. In general, it is recommended that you use the logical data set DUMP command and filter on the data set name to make disaster recovery backups. Logical data set dump processing allows you to back up only your critical data sets and to restore to unlike devices.

Making logical data set dumps for disaster recovery backup requires a naming convention or some other method to identify your critical data sets. If, for example, you establish the convention of having the letters CRIT as the first four characters in the first qualifier of critical data sets, you can back them up for disaster recovery as follows:

```
DUMP  -
    DATASET(INCLUDE(CRIT*.**) -
            BY(MGMTCLAS,EQ,MCNAME))  -
    OUTDDNAME(TAPE)  -
    COMPRESS
```

If for some reason you must do volume dumps for disaster recovery, you should do logical volume dumps instead of physical volume dumps. That way, you can restore the backups to unlike devices. You can perform logical volume dumps by using DATASET (INCLUDE(**)) and either the LOGINDDNAME or LOGINDYNAM keyword with the DUMP command.

Back Up Only Critical Data Sets

You should back up only data sets that are critical to your operation. For example:

- Critical application data sets
- RACF inventory data sets
- System data sets
- Catalogs

Because you normally back up only critical data sets for disaster recovery, the amount of data you have to back up is only a small percentage of all your data. To identify those data sets that you want backed up for disaster recovery, you should create a unique naming convention.

If you have DFSMShsm installed on your system, the recommended method of disaster backup is to use aggregate backup and recovery support (ABARS).

To maintain versions of your disaster recovery backups, you can use generation data group (GDG) dump data sets.

When recovering after a disaster, you may need to use the DELETECATALOGENTRY or IMPORT keywords or both.

Related reading

- For additional information about using the DELETECATALOGENTRY and IMPORT keywords, see “Logical Restore of Data Sets with Phantom Catalog Entries” on page 83.
- For additional information about ABARS, see the *z/OS DFSMShsm Storage Administration Guide*.
- For additional information about disaster recovery, see *MVS/ESA SML: Managing Data*.

Maintaining Vital Records

Vital records are maintained to meet external retention requirements (such as legal requirements).

Like disaster recovery backups, vital records are kept at a remote site and therefore should reside on tape. Vital records are usually an even smaller percentage of all data than disaster recovery backups. Unlike disaster recovery backups, vital records are rarely necessary for normal processing.

Vital records are usually kept for long periods of time. The device they originally resided on may no longer be in use at the time of recovery, and you may need to restore them to unlike devices. Therefore, vital records should be dumped logically so they can be restored to unlike devices. As with disaster recovery, using logical data set DUMP processing requires a naming convention or some other method to identify data sets for dumping.

If, for example, you establish the convention of having the letters VR as the first two characters in the first qualifier of data sets to be backed up for vital records purposes, you can dump them as follows:

```
DUMP  -
    DATASET(INCLUDE(VR*.**) -
            BY(MGMTCLAS,EQ,MCNAME))  -
    OUTDDNAME(TAPE)  -
    COMPRESS
```

For more details on vital records, refer to *MVS/ESA SML: Managing Data*.

Archiving

Archived data sets are data sets created to remove data from active status. This data is placed on alternate storage media because it is not currently being used but may be used in the future. Archived data sets are usually used for long-term retention.

You can use DFSMSdss to archive data sets by periodically filtering on last-referenced date and then dumping and deleting data sets that have not been referenced for long periods of time. This frees space for data that is being accessed more frequently and requires the faster access time of DASD. Because archived data sets might not be recovered for a long time, they should be dumped logically so they can be restored to unlike devices.

For example, the following logical DUMP command results in the archiving of all

data sets in management class MCNAME1 that have not been referred to since April 10, 1999:

```
DUMP  -
     DATASET(BY((REFDT LT 99100)(MGMTCLAS EQ MCNAME1))) -
     OUTDDNAME(TAPE1)  -
     DELETE  -
     COMPRESS  -
     PURGE
```

Related reading: For additional information about archiving, see *MVS/ESA SML: Managing Data*.

Backing Up Data Sets

With the DUMP command, you can dump DASD data to a sequential data set, which can be a generation in a generation data group (GDG). The storage medium for the sequential data set can be tape or DASD. The output data set must be a standard format sequential data set and cannot use any extended-format features, such as compression. If the output resides on DASD, it may be a Large Format data set.

DFSMSdss can dump data sets both logically and physically. Data sets are located by searching either the catalog or the VTOC.

You can select data sets for dump processing based on data set names and numerous data attributes, as discussed in "Choosing Data Sets for Processing—Filtering" on page 18. If you want to perform incremental backups with DFSMSdss, you can filter with BY(DSCHA,EQ,1) to dump only data sets that have changed since the last dump was taken. If you also code the RESET keyword, DFSMSdss changes the data-set-changed (DSCHA) indicator after successfully dumping the data set.

Notes:

1. If you are using DFSMSdss on data sets that DFSMShsm is also backing up, you should not use the RESET keyword because it might cause confusion as to which backup is the most current.
2. DFSMSdss does not permanently record the names of candidate volumes during dump processing.
3. If a Large Format data set is used as the output of a DUMP command, it will not be able to be used as the input to a RESTORE command on a pre-z/OS V1R7 system.

The data-set-changed indicator and the last-referenced date (REFDT) are supported for VSAM and non-VSAM data sets.

Temporary data sets might be included in the data set list at the beginning of a DFSMSdss job. These data sets are created and deleted by other jobs that are running while DFSMSdss is running. Because they are temporary, these data sets can disappear before DFSMSdss finishes. DFSMSdss can issue a message informing the user what happened only at the time DFSMSdss tries to access the data sets. To hold all the data sets in a volume for the entire DFSMSdss execution, write an enqueue installation exit to enqueue the volume for the entire job.

When you create backups of data sets with the DUMP command, you can make multiple (up to 255) dump copies with a single DUMP command. This is done by

specifying multiple ddnames on the OUTDDNAME parameter. To specify multiple ddnames on the OUTDDNAME parameter, you could code:

```
DUMP  -
    DATASET(INCLUDE(**)  -
            BY(MGMTCLAS,EQ,MCNAME1))  -
    OUTDDNAME(TAPE1,TAPE2,TAPE3)  -
    COMPRESS
```

This technique can be helpful if you want to create several backup copies to be used for different purposes.

Unless overridden by the installation options exit routine, DFSMSdss continues dumping while at least one output copy does not have an output error. In the event of an abend, however, DFSMSdss ends without completing any backups.

Related reading

- For more information about using the RESET keyword, see "Backup with Concurrent Copy" on page 36.
- For additional information about the data-set-changed indicator and REFDT, see *z/OS DFSMS Installation Exits*.

Logical Data Set Dump

If you specify the DATASET keyword with the DUMP command and do not specify input volumes, DFSMSdss performs a logical data set dump using information in the catalogs to select data sets. For example, the following DUMP command results in a logical data set dump:

```
DUMP  -
    DATASET(INCLUDE(**)  -
            BY(DSCHA,EQ,YES))  -
    OUTDDNAME(TAPE1)  -
    COMPRESS
```

If you specify the DATASET keyword with the LOGINDDNAME, LOGINDYNAM, or STORGRP keywords, DFSMSdss performs a logical data set dump by using information in the VTOCs to select data sets. For example, the following DUMP command results in a logical data set dump of all the single-volume data sets on volume 338001:

```
DUMP  -
    DATASET(INCLUDE(**))  -
    LOGINDYNAM(338001)  -
    OUTDDNAME(TAPE)  -
    COMPRESS
```

The following data sets cannot be processed by logical data set dump or restore operations:

- VSAM data sets not cataloged in an integrated catalog facility catalog
- Page, swap, and SYS1.STGINDEX data sets
- VSAM Volume Data Sets (VVDS)

- Partitioned data sets containing location-dependent information that does not reside in note lists or in the directory

Note: DFSMSdss cannot be used to dump data sets with a volume serial of MIGRAT. The recommended method of dumping migrated data sets is to use ABARS.

Physical Data Set Dump

If you specify DATASET and INDDNAME or INDYNAM, DFSMSdss performs a physical data set dump. For instance, the following DUMP command results in a physical data set dump:

```
DUMP  -
     INDDNAME(DASD1) OUTDDNAME(TAPE)  -
     DATASET(INCLUDE(**))  -
     COMPRESS -
     OPTIMIZE(4)
```

When multiple input volumes are specified for a physical data set dump operation, multiple logical files (logical volumes) are created for each physical DASD source volume.

DFSMSdss facilitates backup and recovery procedures for physical data set dumps by printing the names of data sets dumped, and the serial and data set sequence numbers of the backup tape volumes on which the dump of a DASD volume begins and ends.

A physical data set dump or restore operation cannot process the following data sets:

- KSDSs with key ranges. Logical processing should be used for this type of data set.
- Extended-format VSAM data sets, including extended-addressable VSAM data sets. Use logical processing for these types of data sets.
- VSAM data sets not cataloged in an integrated catalog facility catalog.
- Page, swap, and SYS1.STGINDEX data sets.

Note: When dumping multivolume data sets, take care to ensure that all volumes where the data set resides are dumped at the same time and restored at the same time. Dumping parts of a multivolume data set and then restoring them may leave the entire data set or those parts unusable. In particular, keyed VSAM data sets are easily damaged by such an operation.

Backup with Concurrent Copy

Programming Interface information

DFSMSdss provides the concurrent copy (CC) function to let you backup data but minimize the time that the data is unavailable. The data base or application determines an appropriate time to start a backup (for example, when the data is in a known state and update activity is stopped). DFSMSdss is invoked directly or via the DFSMSdss application program interface (API) to do a CC of the entire data base. After initialization is complete, DFSMSdss releases any serialization it held on the data sets and prints a message to SYSPRINT and the console that the CC operation is logically complete. If DFSMSdss was invoked via the API, DFSMSdss

informs the caller through a new UIM exit option, Eioption 24. Refer to *z/OS DFSMSdss Storage Administration Reference* for details. The application can resume normal operation at this time.

End of Programming Interface information

If for any reason data cannot be processed with CC, (for example, the hardware being used does not support CC) DFSMSdss uses normal backup methods and does not release the serialization until the backup is completed.

Notes:

1. The CONCURRENT keyword applies to all the data being dumped or copied by the function under which it is specified. It cannot be applied to a subset of the data being processed.
2. To improve data integrity, do not update the data during a CC initialization.
3. If a CC operation fails after signaling that the CC initialization was complete (and update activity on the data has resumed), it is not possible to recover the data at the point-in-time at which the CC operation was started. This is because the data may have been updated while the copy operation was progressing.
4. The CONCURRENT keyword cannot be used with the DELETE or UNCATALOG keywords.
5. The RESET keyword is ignored when CONCURRENT is also specified unless you use a patch to allow it.
6. VM minivolumes are supported if you are using RVA devices to the extent that they are supported by IBM Extended Facilities Product (IXFP) device reporting.

If the source device supports SnapShot, but does not support CC, DFSMSdss will use the SnapShot function to provide a CC-like function known as virtual concurrent copy. Refer to “Performance Considerations” on page 57 for more information on CC and virtual concurrent copy.

Invocation from an Application Program

Usage of the concurrent copy function can also be controlled through the installation options exit, a product-sensitive programming interface intended for customer use. Refer to *z/OS DFSMS Installation Exits* for more information.

Using DFSMSdss as a Backup Utility for CICSVR

CICSVR users can choose DFSMSdss as their backup utility by specifying the CICSVRBACKUP keyword. DFSMSdss notifies the CICSVR server address space every time that a CICSVR backup is made for a VSAM base cluster. CICSVR stores the backup information in its recovery control data set (RCDS). This enables CICSVR to manage backups that are made by DFSMSdss. Through the CICSVR dialog panels, CICSVR (as of Version 3 Release 1) provides complete data set, forward recovery automation by using backups that DFSMSdss makes.

To use the DFSMSdss DUMP command to make CICSVR backups, you must create DFSMSdss DUMP jobs that can be regularly submitted with a production planning system. Specify the CICSVRBACKUP keyword on the logical data set DUMP command. The output data set name must be unique each time the job is run so that multiple backup copies can be maintained.

You can also use the DFSMSdss COPY command to make CICSVR backups.

There are advantages to using the COPY command instead of the DUMP command:

- You can use SnapShot to create the backup instantaneously when the data set resides on a RAMAC Virtual Array (RVA). You can use SnapShot to recover the data set instantaneously to an RVA device.
- You can use data set FlashCopy to create the backup instantaneously when the data set resides on an ESS that supports data set FlashCopy. You can use data set FlashCopy to recover the data set instaneously to a data set FlashCopy-capable ESS.

To use the DFSMSdss COPY function to make CICSVR backups, you must create DFSMSdss COPY jobs that can be regularly submitted with a production planning system. Specify the CICSVRBACKUP and RENAMEUNCONDITIONAL keywords on the data set COPY command. CICSVR provides DFSMSdss with a new name for each VSAM base cluster that is copied when the CICSVRBACKUP keyword is specified. DFSMSdss uses the CICSVR-generated new name instead of the one you specify.

Related reading:

- For additional information about using the CICSVRBACKUP keyword on the DUMP command, see the *z/OS DFSMSdss Storage Administration Reference*.
- For additional information about using different methods to generate a unique output data set name, see the *CICS® VSAM Recovery Implementation Guide*.
- For additional information about the CICSVR generated new name, its naming convention, and required RENAMEU specification, see the *CICS VSAM Recovery Implementation Guide*.

A Backup Scenario

As discussed under “Backup and Recovery” on page 29, you should consider using a combination of incremental and volume backup to fulfill your general availability requirements. Some ways to implement this strategy are:

- Dump a full volume at a given interval—perhaps once a week. Use the RESET keyword to reset the data-set-changed flag. To do full-volume dumps of two volumes at once (in parallel, which is most effective if tapes are on separate channels), code the following:

```
PARALLEL
DUMP INDYNAM(111111) OUTDD(TAPE1) RESET OPTIMIZE(1)
DUMP INDYNAM(222222) OUTDD(TAPE2) RESET OPTIMIZE(2)
```

- Dump only changed data sets at a shorter interval—perhaps daily.

```
DUMP LOGINDY((111111),(222222))  OUTDD(TAPE3) RESET -
    OPTIMIZE(3) DATASET(INCLUDE(**) -
                        BY(DSCHA,EQ,YES))
```

- Use data set naming conventions to set up a dumping scheme that takes account of the relative importance of the data. For example, include CRIT in the

first-level qualifier of all your critical data sets. With this convention in place, you can back up your critical data sets as follows:

```
DUMP LOGINDY((111111),(222222))  OUTDD(TAPE4) RESET -
   OPTIMIZE(4) DATASET(INCLUDE(CRIT*.**) -
                       BY(DSCHA,EQ,YES))
```

Other naming conventions can also be used to identify groups of data sets. For instance, you can use department numbers, charge numbers, user initials, or project codes to identify data sets you want to dump together. For more information on naming conventions, refer to *MVS/ESA SML: Managing Data.*

For data set operations, SYSPRINT contains the names of all the data sets that were dumped for each run. You should keep them for reference if you have to restore a data set and you want it to be at the latest level. This prints a listing of all the data sets that might be on the restore tape, and you can now find the latest dumped version of a particular data set.

Backing Up Data Sets with Special Requirements

Some data sets require special processing when they are backed up. The sections below describe how to back up data sets that have special requirements.

Dumping HFS Data Sets

The following topics present guidelines for backing up an HFS data set with either logical data set dump or physical data set dump.

Logical Dump

Back up mounted HFS data sets with logical data set dump. Logical data set dump provides the quiesce serialization mechanism (BPX1QSE) to ensure data integrity. The quiesce ability allows you to dump an HFS data set while it is in use, as long as you run the dump job on the same system that the HFS data set is currently mounted on.

With DFSMS/MVS® prior to Version 1 Release 5, you were required to specify the SHARE keyword when you wanted to dump a mounted HFS data set. When an HFS data set is mounted, z/OS UNIX System Services (z/OS UNIX) has a shared SYSDSN ENQ. Before DFSMS/MVS Version 1 Release 5, DFSMSdss obtained an exclusive SYSDSN ENQ if you did not specify the SHARE keyword.

Since DFSMS/MVS Version 1 Release 5, DFSMSdss no longer obtains a SYSDSN ENQ, so the SHARE keyword is not required during a logical dump of a mounted HFS data set.

Related Reading: For additional information about the serialization of HFS data sets, see the serialization appendix in *z/OS DFSMSdss Storage Administration Reference.*

Physical Dump

Physical dump does not provide the quiesce serialization mechanism, and it is not recommended for backing up mounted HFS data sets. If you do perform a physical dump of an HFS, do not specify the SHARE keyword. The SHARE keyword applies to the SYSDSN ENQ, and therefore does not provide protection against updates during dump.

Attention: Exercise caution if you use TOL(ENQF) during a physical dump of HFS data sets. Unlike other types of data sets, if an HFS is updated during a physical dump with TOL(ENQF), a subsequent restore will likely result in an unusable data set.

Dumping zFS Data Sets

The following topics present guidelines for backing up an zFS data set with either logical data set dump or physical data set dump.

Logical Dump

Back up mounted zFS data sets with logical data set dump. Logical data set dump provides the quiesce serialization mechanism (BPX1PCT) to ensure data integrity. The quiesce ability allows you to dump a zFS data set while it is in use, as long as you run the dump job on the same system that the zFS data set is currently mounted on.

Physical Dump

Physical dump does not provide the quiesce serialization mechanism, and it is not recommended for backing up mounted zFS data sets.

Attention: Exercise caution if you use TOL(ENQF) during a physical dump of zFS data sets. Unlike other types of data sets, if a zFS data set is updated during a physical dump with TOL(ENQF), a subsequent restore will likely result in an unusable data set.

Dumping Multivolume Data Sets

An important advantage of DFSMSdss as a backup tool is that it can back up multivolume data sets without having to specify any or all of the input volumes. If you do not specify any input volumes (you are using catalog filtering), multivolume data sets will be automatically processed in their entirety. The catalogs are scanned to select an entire data set; that is, the data set is processed in its entirety from all the volumes it resides on. Logical processing consolidates the extents of the data set in one dump data set for you.

If you specify input volumes using the LOGINDDNAME or LOGINDYNAM volume list, a data set is selected based on the following criteria:

- When you either specify SELECTMULTI(ALL) or specify input volumes without specifying the SELECTMULTI keyword, *all* of the volumes that contain a part of a non-VSAM or VSAM cluster must be in the volume list.

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:

  – Specify SPHERE and you must list all parts of the base cluster in the volume list.

  – Do not specify SPHERE and you must list all parts of the base cluster and the associated indexes in the volume list.

- When you specify SELECTMULTI(ANY), *any part* of the non-VSAM data set or VSAM base cluster can be on a volume in the volume list.

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:

  – Specify SPHERE and you must list any part of the base cluster in the volume list.

  – Do not specify SPHERE and you must list any part of the base cluster and the associated alternate indexes in the volume list.

- When you specify SELECTMULTI(FIRST), the volume list must include the volume that contains the *first part* of either the non-VSAM data set or the primary data component of the base cluster for a VSAM sphere.

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:

  – Specify SPHERE and you must list the volume that contains the first extent of the data component for the base cluster in the volume list.
  – Do not specify SPHERE and you must specify the following information in the volume list:
    - The volume that contains the first extent of the data component for the base cluster.
    - The volume that contains the first extent of the data component for the associated alternate indexes.

Guideline: You are not required to specify the SELECTMULTI option when you build a list of volumes using the STORGRP keyword. The volume list contains all of the volumes in a storage group.

The following is an example of the DUMP command with SELECTMULTI specified:

```
DUMP  -
    DATASET(INCLUDE(**))  -
    SELECTMULTI  -
    LOGINDYNAM(338001)  -
    OUTDDNAME(TAPE)  -
    COMPRESS
```

SELECTMULTI works only for logical data set dumps. If you dump a multivolume data set physically, you must ensure that the segments from all the volumes are dumped together. If you dump a multivolume data set physically, it is dumped from all the volumes that are passed. The output dumped data contains a logical file for each selected volume.

A DFSMSdss logical data set dump operation attempts to ensure that all parts of a multivolume non-VSAM data set exist. In cases where a part of the data set is missing, such as an inadvertent scratching of the VTOC entry on a volume, DFSMSdss issues an error message and discontinues processing the data set.

DFSMSdss cannot process the following non-VSAM data sets because they are missing one or more parts:

- Multivolume data sets whose catalog volume order differs from the VTOC volume order
- Single-volume data sets with the same name that are cataloged as one multivolume data set
- Multivolume data sets whose last volume indicator in the VTOC entry is not set

A multivolume data set standard user label is not supported.

Dumping Integrated Catalog Facility User Catalogs

Another important use of DFSMSdss as a backup tool is the backing up of integrated catalog facility user catalogs and their aliases (using logical data set dump). The user catalog name must be fully qualified with the INCLUDE keyword on the DUMP command. The LOCK attribute of an integrated catalog facility user catalog is dumped. The LOCK status is preserved if the catalog does not exist at restore time. Otherwise, the LOCK status of the existing catalog is used.

The following example shows the JCL used to dump an integrated catalog facility user catalog. RACF access to the catalog is not required if you have RACF DASDVOL update access or if the installation authorization exit routine bypasses authorization checking.

```
//STEPT006 EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPE     DD DISP=(NEW,PASS),LABEL=(1,SL)
      VOL=SER=(A00760),DSN=PUBSEXMP.DUMP,
      UNIT=3590,DCB=(BLKSIZE=32760)
//SYSIN    DD *
  DUMP DS(INC(TEST.CAT.PUBSEXMP)) -
            OUTDDNAME (TAPE)
/*
```

Figure 1 shows printed output produced by the dump.

```
PAGE 0001     5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES     1999.211 14:54
   DUMP                          -
      DS(INCL(TEST.CAT.PUBSEXMP)) -
      OUTDDNAME (TAPE)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
ADR109I (R/I)-RI01 (01), 1999.211 14:54:32 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED.
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 1999.211 14:54:32 EXECUTION BEGINS
ADR801I (001)-DTDSC(01), DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA SETS WERE SELECTED: 0 FAILED
                         SERIALIZATION AND 0 FAILED FOR OTHER REASONS.
ADR454I (001)-DTDSC(01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                          CLUSTER NAME   TEST.CAT.PUBSEXMP
                          CATALOG NAME   SYS1.MVSRES.MASTCAT
                          COMPONENT NAME TEST.CAT.PUBSEXMP
                          COMPONENT NAME TEST.CAT.PUBSEXMP.CATINDEX
ADR006I (001)-STEND(02), 1999.211 14:54:32 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 1999.211 14:54:32 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 1999.211 14:54:32 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

Figure 1. Output from a Dump of an Integrated Catalog Facility User Catalog

Related reading

- For additional information about the LOCK attribute, see *z/OS DFSMS Managing Catalogs.*
- For additional information about the installation authorization exit routine, see *z/OS DFSMS Installation Exits.*

Dumping Non-VSAM Data Sets That Have Aliases

DFSMSdss does not support INCLUDE filtering of non-VSAM data sets using an alias. To include a non-VSAM data set that has an alias for dump processing, you must use the data set's real name, as shown in the VTOC. DFSMSdss does not detect or preserve aliases of non-VSAM data sets. You will need to redefine the aliases after the data set is dumped and restored.

Dumping VSAM Spheres

Using the SPHERE keyword, you can dump an entire VSAM sphere (base cluster and all associated alternate index clusters and paths). To dump the base cluster and the other components, all you need to specify is the base cluster name.

An example of the DUMP command with the SPHERE keyword is:

```
DUMP  -
        OUTDDNAME(TAPE)  -
        DATASET(INCLUDE(PARTS.VSAM1))  -
        SPHERE  -
        PSWD(PARTS.VSAM1/MASTUPW1)  -
        COMPRESS
```

Note: You should be aware that you cannot restore a sphere unless it is dumped as a sphere with the SPHERE keyword.

Dumping Indexed VSAM Data Sets

Indexed VSAM data sets (such as key sequenced or variable relative record data sets) can be logically dumped either without regard for the track contents or with validity checking of each track as the tracks are written. If dumped in the latter format, they must be restored on a system that supports the validate function.

The VALIDATE keyword, which is the default, specifies that the index and data track contents are to be validated as the tracks are dumped. Spanned record errors and split errors are detected and reported, but the dump continues. If other errors are detected, a message is issued and the dump stops.

The validate function can be overridden with the NOVALIDATE keyword, which specifies that no validation is done as the tracks are dumped. Some errors may not be detected until the data set is restored.

Note: Extended-format VSAM data sets cannot be dumped with the NOVALIDATE keyword.

Dumping SYS1 System Data Sets

DFSMSdss allows data sets with a high-level qualifier of SYS1 to be dumped, deleted, and uncataloged. You must use the PROCESS(SYS1) keyword with the DUMP command. The SYS1.VVDS and SYS1.VTOCIX data sets are an exception to this processing.

SYS1.VVDS and SYS1.VTOCIX data sets can be physically, but not logically, dumped. Also, the SYS1.VVDS data set cannot be deleted or uncataloged.

Guideline: To limit the use of the PROCESS keyword, it is recommended that the PROCESS keyword be protected by a security program, such as RACF.

Related reading: For additional information about the RACF facility class profile, see the *z/OS Security Server RACF Security Administrator's Guide.*

Dumping Data Sets Containing Records Past the Last-Used-Block Pointer

Some data sets on your system may contain records past the last-used-block pointer in the data set's VTOC entry. This could be a result of a data set not being properly closed or an application that accesses data in such a way as to bypass the updating of this field. In this case, special consideration needs to be given to these data sets as DFSMSdss recognizes this block pointer as the end of the used space in the data set and, therefore, the end of the real data.

Using the ALLDATA or ALLEXCP keyword will result in all the allocated space being dumped for applicable data sets. This includes all the data up to the last block pointer as well as all the data to the end of the allocated space. However, whether or not all the data is restored depends on data set characteristics and device characteristics during the restore. For example, if the data set must be reblocked (either because the target is an unlike device, the REBLOCK keyword is specified, or the data set is marked reblockable) only the used space will actually be restored. This limitation is due to the fact that any residual data (that is in the unused portion of the data set) will likely have different characteristics than the real data (that is in the used portion of the data set). This inconsistency would result in data incompatibilities causing the restore to fail, and thereby inhibiting the ability to restore the real data. Because of this, DFSMSdss will only restore the data in the used portion of the data set when the data characteristics must change.

If you require that all of the unused space is restored, then you should ensure that the data set is restored to a like device type and not reblocked or compressed. (Compress is the default for PDS data sets on restore unless you use the NOPACKING keyword.) In this case, the characteristics of the data do not change, and DFSMSdss will restore all the allocated space.

Backing Up SMS-Managed Data Sets

When backing up data sets in an SMS-managed environment, you need to think about some special conditions in addition to those discussed under "Backing Up Data Sets" on page 33. The following sections discuss how you can back up SMS-managed data sets in an SMS-managed environment.

In most cases, you should let DFSMShsm back up SMS-managed data sets for you. However, if you do not have DFSMShsm or if you prefer not to rely on it for all your backup requirements, you can use DFSMSdss to back up your SMS-managed data sets.

Filter on Class Names

DFSMSdss can select data sets for dump processing based on their storage, management, and data class names. Because management class is the construct that contains a data set's availability attributes, you might want to filter on it when selecting data sets for dump processing.

If you want to back up data sets in a particular management class, you can filter on the management class name. For example, if you want to perform incremental

backup on data sets in management classes MCNAME1 and MCNAME2, specify the DUMP command as follows:

```
DUMP   -
    DATASET(INCLUDE(**)  -
             BY((MGMTCLAS,EQ,(MCNAME1,MCNAME2)) (DSCHA,EQ,YES))) -
    OUTDDNAME(OUTVOL1)
```

Class Names Saved

DFSMSdss saves the class names of the data sets it dumps. These names are then used as input to ACS routines when the data set is restored.

Backing Up Data Sets Being Accessed with Record Level Sharing

During logical data set dump operations of SMS-managed VSAM data sets, DFSMSdss communicates with VSAM RLS to perform quiesce processing of data sets that are being accessed by another job using Record Level Sharing (RLS).

By default, DFSMSdss does not use timeout protection during RLS quiesce processing. You can control whether or not DFSMSdss uses timeout protection during RLS quiesce processing and what the timeout value should be using the DSSTIMEOUT parameter of the IGDSMS*xx* PARMLIB member.

You can also change the timeout value without IPLing the system using the SETSMS DSSTIMEOUT(*nnnnn*) command.

Related reading:

- For additional information about using the IGDSMS*xx* member of PARMLIB to control the RLS timeout value used during DFSMSdss operations, see the *z/OS DFSMSdfp Storage Administration Reference*.
- For additional information about using the SETSMS command, see*z/OS MVS System Commands*.

Backing Up Volumes

With DFSMSdss, you can back up volumes either logically or physically. If the volume is to be restored to an unlike device, you must dump it logically. See Appendix B, “Linux-z/OS DFSMSdss Dump or Restore HOW-TO,” on page 169 to find out how to use z/OS DFSMSdss to back up Linux for OS/390 or Linux for zSeries partitions and volumes.

Logical Volume DUMP

To perform a logical volume dump, you specify DATASET(INCLUDE(**)) with either LOGINDDNAME or LOGINDYNAM. LOGINDDNAME identifies the input volume that contains the data sets to be dumped. LOGINDYNAM specifies that the volumes containing data sets to be dumped be dynamically allocated.

Here is an example of how you specify the DUMP command to perform a logical volume dump:

```
DUMP DATASET(INCLUDE(**)) -
     LOGINDDNAME(DASD1) -
     OUTDDNAME(TAPE)
```

Note: Certain data sets can be restored only to like devices even though they were dumped logically.

Physical Volume Dump

To perform a physical volume dump, specify the DUMP command with INDDNAME or INDYNAM and OUTDDNAME. Because FULL is the default keyword for the DUMP command, you need not specify it. Unallocated tracks are not dumped. The following example shows how you can specify the DUMP command to physically back up a volume:

```
DUMP INDDNAME(DASD1) OUTDDNAME(TAPE)
```

Backing up System Volumes

If you plan to use Stand-Alone restore to restore a volume (with no operating system), the volume must be dumped physically. In addition, when doing a full physical volume dump to back up a system residence volume, you must use JCL to invoke DFSMSdss.

Backing up VM-Format Volumes

You can use DFSMSdss to back up VM-format volumes that are accessible to your MVS system. The volumes must have OS-compatible VTOCs starting on track zero, record five. DFSMSdss can only retrieve device information from the OS-compatible VTOC; it cannot interpret any VM-specific information on the volume.

Use the CPVOLUME keyword and specify the range of tracks to be backed up with the TRACKS keyword. You can use concurrent copy on VM-format volumes by specifying the CONCURRENT keyword. Because DFSMSdss cannot check access authorization for VM data, CPVOLUME is only allowed with the ADMINISTRATOR keyword.

Exercise caution when using DFSMSdss to back up VM-format volumes, because DFSMSdss does not serialize any VM data in any way. Dumps of VM-format volumes cannot be restored with Stand-Alone Restore.

Dumping Data Efficiently

When backing up data, you can specify both the OPTIMIZE and the COMPRESS keywords to improve performance and save dump space. The two keywords can be used together.

A selective data set dump operation saves space, while a full-volume dump operation saves time. The same applies to the COMPRESS keyword. It saves dump space, but involves some processing overhead. In general, if you are dumping to tape, saving space is probably less of a concern than performance. Usually, saving space is important only when it results in using fewer tapes to store the data. Using fewer tapes reduces the number of tape mounts that are necessary to recover the data.

Combining Volume Copy and Volume Dump to Reduce Your Backup Window

You can use physical full volume copy in conjunction with FlashCopy or SnapShot to reduce the amount of time that your data is unavailable when you back it up.

Full volume copy, in conjunction with FlashCopy or SnapShot, can produce a copy of a volume in seconds. Then, DFSMSdss can dump the copy to tape while your applications are accessing the data on the original volume.

Related reading: For additional information about full volume copy, FlashCopy, and SnapShot, see "Moving Volumes" on page 116.

Combining the Functions

To combine the volume copy and volume dump functions to reduce your backup windows, perform the following procedure:

1. Stop application access to the volumes.
2. Copy the volumes by using full volume copy. Do not specify FASTREPLICATION(NONE). The copies complete very quickly if DFSMSdss can use FlashCopy or SnapShot.
3. Enable application access to the volumes.
4. Backup the copies to tape using full volume dump.

Special Considerations

When you combine the volume copy function with the volume dump function, you must give consideration to how you will use the following keywords:

- DUMPCONDITIONING
- FCNOCOPY
- FCWITHDRAW

DUMPCONDITIONING — You can use the DUMPCONDITIONING keyword on the full volume copy command in step 2 to allow the target volume to remain online for dumping.

The target volume of a full volume copy operation with DUMPCONDITIONING specified is referred to as a *conditioned* volume. A full volume dump of a conditioned volume looks as if it was dumped from the original source volume of the copy operation. For example, if you perform a full volume copy with DUMPCONDITIONING of a volume called VOL001 to a volume called VOL002 and then you perform a full volume dump of VOL002, the dump data set looks as if it were created by a full volume dump of VOL001. This assumes that the source volume VOL001 has an indexed VTOC.

If the source volume does not have an indexed VTOC, a full volume dump of the conditioned volume VOL002 does not look as if was dumped from the original source volume VOL001. Rather, it will be an exact image of the conditioned volume. A subsequent full volume restore with the COPYVOLID keyword specified results in the target volume having the same serial number as the conditioned volume.

FCNOCOPY/FCWITHDRAW — You want to use these two keywords in your procedure when using FlashCopy. The FCNOCOPY keyword on the COPY command prevents the ESS subsystem from performing a full physical copy of the volume. Performing the full physical copy uses subsystem resources and can impact the performance of other I/O operations that are issued to the ESS.

Specifying the FCWITHDRAW keyword on the DUMP command causes DFSMSdss to withdraw the FlashCopy relationship after the volume has been successfully dumped. This frees the subsystem resources that are used to maintain the FlashCopy relationship.

However, timing is a major factor in the successful use of these keywords in your procedure. Only a short amount of time must pass between the completion of step 2 (copy function) and the start of step 4 (backup function). When this is the case, you can specify the FCNOCOPY keyword in step 2 and the FCWITHDRAW keyword in step 4.

Do not use the FCNOCOPY and FCWITHDRAW keywords if the backup (step 4) will not be performed within a reasonable amount of time after the copy (step 2). Otherwise, the use of FlashCopy consumes the subsystem resources for an extended amount of time.

Restrictions: Using the FCNOCOPY keyword on the full volume copy and the FCWITHDRAW keyword on the full volume dump leaves the target volume (of the copy) in an indeterminate state. Some of the tracks on the volume may contain data from the source volume. Other tracks may contain residual data that was on the target volume before the copy. This indeterminate state can cause problems when accessing the target volume following the dump, if the VTOC locations of the source volumes and the target volumes are different before the copy. To avoid these problems, do one of the following:

- Ensure that the VTOC locations for the source volumes and the target volumes are the same before you initiate the copy.
- Add an ICKDSF INIT step for the target volume of the COPY in step 2. Add this step after step 4 (Backup the copies to tape using full volume dump). The target volume of the copy initializes and returns to a consistent state.

Related reading

- For additional information about using the DUMPCONDITIONING keyword, see the *z/OS DFSMSdss Storage Administration Reference*.
- For additional information about using the FCNOCOPY and FCWITHDRAW keywords, see the *z/OS DFSMSdss Storage Administration Reference*.
- For additional information about ICKDSF, see the *Device Support Facilities User's Guide and Reference*.

Backing Up and Restoring Volumes with Incremental FlashCopy

Backing Up a Volume and Refreshing the Backup

Incremental FlashCopy operates at the full volume level. You can use Incremental FlashCopy to create an initial point-in-time copy of a source volume and refresh the target volume by copying only the changed data. After the initial full volume copy of the source volume to the target volume, the FlashCopy relationship remains (persists) between the source and target volume pair and the changes on the source and target volumes since the last point-in-time copy are tracked. When you refresh the target volume at a new point-in-time, only the changed tracks are copied. Incremental FlashCopy helps reduce the physical background copy time when only a subset of the data on the source or target has changed.

Restoring a Volume

The direction of the refresh can be reversed when you indicate the original target now becomes the source and the original source becomes the target. Only the changed data since the last point-in-time copy is copied. If no updates were made to the target since the last incremental copy, the reverse of FlashCopy direction can be used to restore the original source back to the previous point-in-time state.

FCINCREMENTAL

You can use the FCINCREMENTAL keyword for a COPY FULL or COPY TRACKS CPVOLUME operation to perform an initial full volume copy if no Incremental FlashCopy relationship exists between the volume pair. If there is an existing Incremental FlashCopy relationship between the volume pair, DFSMSdss will copy the changed tracks in the (new) direction designated by the INDDNAME/INDYNAM and OUTDDNAME/OUTDYNAM keywords. The new direction can be the same or reverse of the original (existing) direction.

FCINCREMENTALLAST

You can use the FCINCREMENTALLAST keyword on the COPY FULL or COPY TRACKS CPVOLUME commands to copy the changed tracks when there is an Incremental FlashCopy relationship between the volume pair. FCINCREMENTALLAST specifies that the new FlashCopy relationship will be non-persistent and Change Recording will be stopped after the final increment has been established. The FlashCopy relationship will end when background copy for the final increment has completed.

FCINCRVERIFY(NOREVERSE | REVERSE)

You can use the FCINCRVERIFY(NOREVERSE | REVERSE) keyword to verify the existing Incremental FlashCopy direction is what you expected. DFSMSdss will fail the copy attempt if the existing direction is not as expected.

FCWAIT

When you reverse the direction of an Incremental FlashCopy, the storage facility requires that the previous background copy has completed. You can specify the FCWAIT keyword with a query interval value in seconds and a number of retries value to direct DFSMSdss to wait for background copy completion before initiating the new Incremental FlashCopy.

Notes:

1. Incremental FlashCopy relationship is limited to one full volume incremental relationship per volume. However, an Incremental FlashCopy relationship can coexist with other non-incremental FlashCopy relationships. Other limits remain, such as each source track can have up to 12 targets.
2. Incremental FlashCopy is only possible with a persistent relationship. With persistent relationships, the relation between the source and target is maintained after the background copy has completed.
3. DFSMSdss allows you to copy full volumes to target devices of greater capacity. However, if the original FlashCopy target volume is bigger than the source volume, you will not be able to reverse the FlashCopy direction.
4. When you specify DUMPCONDITIONING with FCINCREMENTAL, the volume serial number of the target volume does not change, and the target volume remains online after the copy. A subsequent incremental copy can be made without additional procedure.
5. When you specify COPYVOLID with FCINCREMENTAL, the volume serial number of the target volume is changed to match the source's. When the volume serial number on a DASD volume is changed, the operator is notified.

& The operating system then initiates a demount of the volume. Prior to
& performing a subsequent incremental copy using DFSMSdss, the offline
& volume's volume serial number must be changed by using a utility such
& ICKDSF and the volume must be varied online.

& 6. The PURGE keyword may be required on subsequent incremental copies.

Usage Scenario 1: Periodic Dump to Tape

The following example describes how Incremental FlashCopy can be used to create periodic backups to tapes.

& • Step 1 - Copy volume VOL00A->VOL00B by performing initial Incremental
& FlashCopy from volume VOL00A to VOL00B. Background copy task copies all
& the tracks on the source volume to the target volume.

```
& COPY FULL INDYNAM(VOL00A) OUTDYNAM(VOL00B) DUMPCONDITIONING -
&      ADMIN PURGE FCINCREMENTAL
```

& • Step 2 - Backup volume VOL00A by performing full volume dump from volume VOL00B to tape.

```
DUMP FULL INDYNAM(VOL00B) OUTDD(TAPE01)
```

• Step 3 - Allow update to volume VOL00A

& • Step 4 - Refresh volume VOL00B by performing subsequent Incremental
& FlashCopy from volume VOL00A to volume VOL00B. Background copy task
& copies changed tracks from volume VOL00A to VOL00B.

```
& COPY FULL INDYNAM(VOL00A) OUTDYNAM(VOL00B) DUMPCONDITIONING -
&      ADMIN PURGE FCINCREMENTAL
```

& • Step 5 - Repeat steps 2-4.

Usage Scenario 2: Check-point Batch Processing with Incremental FlashCopy

Below is an example of how DFSMSdss Incremental FlashCopy can be used in check-point batch processing. DFSMSdss Incremental FlashCopy function does not inhibit target writes. In order to be able to restore the original source to the state of the previous point-in-time copy, the user must not have written to the target volume after DFSMSdss finished the previous copy operation. To improve performance, the user can also specify the optional ADMINISTRATOR keyword.

& • Step 1 - Backup volume VOL00A->VOL00B by performing initial Incremental
& FlashCopy from volume VOL00A to VOL00B. Background copy task copies all
& the tracks on the source volume to the target volume.

```
& COPY FULL INDYNAM(VOL00A) OUTDYNAM(VOL00B) DUMPCONDITIONING -
&      ADMIN PURGE FCINCREMENTAL
```

& • Step 2 - Start batch application which makes updates to volume VOL00A.

& • Step 3 - Backup volume VOL00A->VOL00B by performing subsequent
& Incremental FlashCopy. Background copy task copies only updated tracks from
& volume VOL00A to volume VOL00B.

```
& COPY FULL INDYNAM(VOL00A) OUTDYNAM(VOL00B) DUMPCONDITIONING -
&      ADMIN PURGE FCINCREMENTAL
```

& • Step 4 - Start batch application which makes updates to volume A.

& • Step 5 - When batch updates are incomplete or the batch job ends abnormally,
& tell DFSMSdss to wait for background copy completion and perform Incremental
& FlashCopy in reversed direction to restore the previous point-in-time copy (made
& in step 3). Background copy task copies only updated tracks from volume
& VOL00B to VOL00A. The user indicates the original source (VOL00A) is now the

& target and the original target (VOL00B) is now the source on the DFSMSdss
& COPY command. FCINCREMENTAL indicates Change Recording and Persistent
& Relationship should continue.

```
& COPY FULL INDYNAM(VOL00B) OUTDYNAM(VOL00A) DUMPCONDITIONING -
&      ADMIN PURGE FCINCREMENTAL
&      FCINCRVFY(REVERSE) FCWAIT(30,10)
```

& Optionally, FCINCRVERIFY(REVERSE) can be specified to verify FlashCopy
& direction:

```
& COPY FULL INDYNAM(VOL00B) OUTDYNAM(VOL00A) DUMPCONDITIONING -
&      ADMIN PURGE FCINCREMENTAL -
&      FCINCRVFY(REVERSE) FCWAIT(30,10)
```

& • Step 6 - Restart batch application which makes updates to volume VOL00A.
& • Step 7 - When batch application errors occur, perform Incremental FlashCopy
& without reversing the direction. There is no need to wait for background copy
& completion.

```
& COPY FULL INDYNAM(VOL00B) OUTDYNAM(VOL00A) DUMPCONDITIONING -
&      ADMIN PURGE FCINCREMENTAL
```

& Restart batch application which updates volume VOL00A.

& If batch application errors occur again, repeat step 7.

& • Step 8 - Upon successful batch application completion, perform Incremental
& FlashCopy in reversed direction and make a new backup copy of VOL00A:

```
& COPY FULL INDYNAM(VOL00A) OUTDYNAM(VOL00B) DUMPCONDITIONING -
&      ADMIN PURGE FCINCREMENTAL FCWAIT(30,10)
```

& • Step 9 - Restart batch application which updates volume VOL00A.

Related reading

& • For additional information about using the FCINCREMENTAL,
& FCINCREMENTALLAST, FCINCRVERIFY(NOREVERSE | REVERSE), and
& FCWAIT(numsecs, numretries) keywords, see*z/OS DFSMSdss Storage*
& *Administration Reference*.
• For additional information about Incremental FlashCopy and Persistent FlashCopy, see the *z/OS DFSMS Advanced Copy Services*.

& Using Encryption to Secure Backups

& When backing up your data, you may secure it by requesting that DFSMSdss
& encrypt the data before writing it to the dump data set on DASD or tape. There are
& three types of symmetric encryption that DFSMSdss uses to secure your data:
& Triple-length Data Encryption Standard (TDES) clear keys, secure TDES keys, and
& 128-bit Advanced Encryption Standard (AES) clear keys. DFSMSdss offers two
& types of key management options. Certain hardware and software requirements
& must be satisfied before DFSMSdss is able to perform encryption of user data.

& DFSMSdss makes use of the IBM Cryptographic Services Facility (ICSF) to manage
& cryptographic keys for encrypted data. ICSF supports the following cryptographic
& standards and architectures:

& • IBM Common Cryptographic Architecture (CCA) that is based on the ANSI Data
& Encryption Standard (DES)
& • Advanced Encryption Standard (AES).

& In the secret key cryptography system based on DES, two parties share secret keys
& that are used to protect data and keys that are exchanged on the network. The

& sharing of secret keys establishes a secure communications channel. The only way
& to protect the security of the data in a shared secret key cryptographic system is to
& protect the secrecy of the secret key.

& ICSF also supports triple DES encryption for data privacy. TDES triple-length keys
& use three, single-length keys to encipher and decipher the data. This results in a
& stronger form of cryptography than that available with single DES encipher.

& With AES, data can be encrypted and decrypted using 128-bit, 192-bit, and 256-bit
& clear keys. CBC and ECB encryption are also supported. For public key
& cryptography, ICSF supports both the Rivest-Shamir-Adelman (RSA) algorithm 1,
& and the NIST Digital Signature Standard (DSS) algorithm. RSA and DSS are the
& most widely used public key encryption algorithms. In this system, each party
& establishes a pair of cryptographic keys, which includes a public key and a private
& key. Both parties publish their public keys in a reliable information source, and
& maintain their private keys in secure storage.

& Cryptographic keys and DFSMSdss

& DFSMSdss makes use of TDES triple-length keys and 128-bit AES keys for data
& encryption. On a system with secure cryptographic hardware, you can use
& DFSMSdss to generate TDES and AES keys and encrypt them for protection
& through RSA public keys. On systems without secure cryptographic hardware, a
& password allows the generation of clear TDES and AES keys. The use of these
& cryptographic keys with DFSMSdss depends on the type of processor and the type
& of cryptographic hardware that you have installed.

& RSA public and private keys for encryption can be stored in the ICSF Public Key
& Data Set (PKDS). These RSA keys are used by DFSMSdss to protect the symmetric
& keys that protect the data. You can use RACF commands to store public/private
& keys.

& Encryption Considerations

& The choice of which type of encryption to use depends on several factors,
& including performance and level of security. A clear TDES key is requested via the
& CLRTDES sub parameter of the ENCRYPT keyword. A secure TDES key is
& requested via the ENCTDES sub parameter of the ENCRYPT keyword. A clear
& 128-bit AES key is requested via the CLRAES128 subparameter of the ENCRYPT
& keyword.

& The decision to use CLRTDES or ENCTDES key values depends on the kind of
& cryptographic hardware you have, the level of security you want, and the level of
& performance you want from DFSMSdss.

& For DFSMSdss, a CLRTDES key is a triple-length TDES key that is generated
& dynamically. Unlike the ENCTDES key value the CLRTDES key value can appear
& in application storage. If DFSMSdss is running on a z890, z990, or System z9 109,
& the data is encrypted using the clear TDES key on the CPACF, and this usually
& results in better performance than if you are using the ENCTDES key value.

& The ENCTDES key is a triple-length TDES key that is generated within the secure
& boundary of the cryptographic hardware (CCF, PCICC, PCIXCC, or CEX2C), and it
& uses the ICSF symmetric master key to encrypt the data. The clear value of an
& ENCTDES key never leaves the boundary of the secure cryptographic hardware.
& Encryption and decryption of data using an ENCTDES key requires secure
& cryptographic hardware to be available.

& Each type of key is equally secure in regards to the data that appears in the output
& data set.

& During DUMP processing, only user data may be encrypted. This means that
& VTOC and VVDS tracks that are processed are not encrypted. The data set names
& and other content from the VTOC will appear unencrypted in the output dump
& data set.

& Key Management Considerations

& The RSA and KEYPASSWORD keywords are used for key management by
& DFSMSdss. The choice of one over the other depends on your environment and
& needs.

& KEYPASSWORD Keyword: Generally, if you are encrypting low volumes of data or
& if you do not have secure cryptographic hardware installed, you can specify the
& KEYPASSWORD keyword. NOTE: Passwords are case sensitive.

& The iteration count (ICOUNT) in Password Based Encryption (PBE) is intended to
& strengthen weak passwords. If the password is robust (that is, 32 random
& characters), the default of 16 provides reasonable security and performance. Most
& PBE schemes assume that weak password are chosen; thus, iteration counts of 1000
& or higher are often normal.

& **Note:**

& You must take care when using the KEYPASSWORD keyword. The same
& password specified on the DUMP task must be specified on the RESTORE
& task. The password is not stored in the dump data set in any form. If the
& password is lost, the encrypted data in the dump data set cannot be
& decrypted.

& The same password with the same iteration count (ICOUNT) generates the
& same data key. This means that if the same password is used for many
& DUMP tasks, all of the data from those DUMP jobs are protected by the
& same key. If the password is compromised, all of the dump data is
& vulnerable.

& RSA Keyword: The RSA keyword makes use of public/private keys for encryption
& and the exchange of digital certificates. You specify the label of the public key that
& is stored in the ICSF PKDS on the RSA keyword when you dump and encrypt the
& data. The corresponding RSA private key must be present at the recovery site
& when you decipher the data. A recipient at another site can only decrypt the data
& through the private key that is specified on the RSA keyword during the RESTORE
& job. If the original RSA key and label exist in the system's ICSF PKDS, then the
& RSA keyword need not be specified. The original RSA label is stored on the dump
& data set for convenience.

& If the same RSA label and key are used during multiple dumps, each dump has its
& data encrypted with a different symmetric key. Thus, if the symmetric key of one
& dump is discovered, the data in the other dumps is still secure.

& Use of Compression With Encryption

& When you plan to archive large amounts of encrypted data, you might consider
& compressing the data (for example, to reduce the number of tape volumes needed).

& Some tape devices make use of their own compression when you store data.
& Encrypted data is not highly compressible, so you might want to compress your
& data before encryption using the HWCOMPRESS keyword.

& During DUMP processing, only user data may be compressed. This means that
& VTOC and VVDS tracks that are processed are not compressed. The data set names
& and other content from the VTOC will appear uncompressed in the output dump
& data set.

& Encryption Examples

& You can use the following examples when using encryption:

& This example shows the SYSIN parameters passed to DFSMSdss to perform a full
& volume dump, and to compress and encrypt the volume data using a clear TDES
& key. The clear TDES key is protected using an RSA private key.

```
& DUMP FULL INDYNAM(VOL001) OUTDD(TAPE1) -
& ENCRYPT(CLRTDES) RSA(SYSTEM.PRIVATE.S01024) -
& HWCOMPRESS OPTIMIZE(4)
```

& This example shows the RESTORE command needed to restore the data on a
& system that has the same RSA private key with the same label.

```
& RESTORE FULL INDD(TAPE1) OUTDYNAM(VOL001)
```

& This example shows the keywords needed to restore the data on a different system
& that has had the original RSA private key loaded into ICSF under a different label.

```
& RESTORE FULL INDD(TAPE1) OUTDYNAM(VOL001) -
& RSA(NEWSYSTEM.PRIVKEY.S01024)
```

& This example shows the backup and recovery of data sets while using 128-bit AES
& encryption to secure the data. Note that a password is used for key management.

```
& DUMP DATASET(INCLUDE(SOURCE.**)) -
& OUTDDNAME(TAPE2) HWCOMPRESS -
& KEYPASSWORD(mySecretPASSWORD) -
& ENCRYPT(CLRAES128)
&
& RESTORE DATASET(INCLUDE(SOURCE.**)) -
& INDDNAME(TAPE2) -
& REPLACE KEYPASSWORD(mySecretPASSWORD)
```

& Hardware Requirements for Encryption

& RSA private tokens: When you use the RSA option with DFSMSdss to encrypt the
& data- encrypting key, you must consider the cryptographic hardware that exists at
& the site that decrypts the data. Not all types of RSA private keys are supported by
& all types of cryptographic hardware. The table below summarizes the RSA private
& tokens and required cryptographic hardware for decryption. For more details, see
& z/OS Cryptographic Services ICSF Application Programmer's Guide.

& New table:

```
& RSA private key token (internal) Required cryptographic hardware
& ---------------------------------------------------------------
& RSA private key token 1024       One of the following:
& Modulus-Exponent Internal form   o Cryptographic Coprocessor
&                                    Feature
&                                  o PCI X Cryptographic Coprocessor
&                                  o Crypto Express2 Coprocessor
& ---------------------------------------------------------------
& RSA private key token 1024       One of the following:
```

Chinese Remainder Theorem Internal form	o PCI Cryptographic Coprocessor o PCI X Cryptographic Coprocessor o Crypto Express2 Coprocessor
RSA private key token 2048 Chinese Remainder Theorem Internal form	One of the following: o PCI Cryptographic Coprocessor with LIC January 2005 or later and z/OS ICSF HCR770B or later o PCI X Cryptographic Coprocessor o Crypto Express2 Coprocessor

Performance and Hardware Types

The performance of encryption may vary depending on the type of zSeries System that the job is being run on and the type of encryption being performed. DFSMSdss can use the Cipher Message with Chaining (KMC) zSeries instruction for some types of encryption if it is running on the appropriate type of hardware. If the KMC instruction is not available or the type of encryption requires it, DFSMSdss must use the appropriate ICSF service to perform the encryption. The table below describes the method of encryption that is used under various encryption types and zSeries hardware.

zSeries System Encryption Type	Method of Encryption
z/800, z/900	
o Clear TDES	ICSF Service
o Clear 128-bit AES	ICSF Service
o Secure TDES	ICSF Service
z/890, z990	
o Clear TDES	KMC instruction
o Clear 128-bit AES	ICSF Service
o Secure TDES	ICSF Service
z8, z9 109	
o Clear TDES	KMC instruction
o Clear 128-bit AES	KMC instruction
o Secure TDES	ICSF Service

Software Requirements for Encryption

DFSMSdss requires the installation of the Encryption Facility DFSMSdss Encryption Feature (HCF773D) before it can perform encryption related tasks.

IBM Cryptographic Services Facility (ICSF) (HCR770B or higher) must be installed and operating for DFSMSdss to perform encryption.

ICSF Callable Services for DFSMSdss

DFSMSdss invokes the following ICSF callable services for the DUMP command. If you are using RACF or similar security product, ensure that the security administrator authorizes DFSMSdss to use the following services and any cryptographic keys that are specified as input. For information about the ICSF callable services, see z/OS Cryptographic Services ICSF Application Programmer's Guide.

- CSFCKM Multiple clear key Import
- CSFENC Encipher
- CSFRNG Generate a random number
- CSFSYE Encipher using clear DES/AES key

& • CSFPKE Public key encrypt
& • CSFSYG Generate and wrap a symmetric key
& • CSFSYX Export a symmetric key
& • CSFOWH One-way hash

& DFSMSdss invokes the following ICSF callable services for the RESTORE
& command.
& • CSFCKM Multiple clear key import
& • CSFDEC Decipher
& • CSFSYD Decipher using clear DES/AES key
& • CSFOWH One-way hash
& • CSFPKD Public key decrypt
& • CSFSYI Import a symmetric key

Space Considerations

Using larger block sizes saves dump space and improves performance by minimizing the number of I/O operations performed during a dump operation.

The default block size for output records that are written to **tape** is 65 520 bytes (65 520 is also a maximum). You can change this default to 32 760 bytes by using the installation options exit routine. Refer to *z/OS DFSMS Installation Exits* for more information on the installation options exit routine.

For output records that are written to DASD, the block size is the track length of the output volume for devices whose track length is less than 32KB. It is one half the track length for devices whose track length is greater than 32KB. You can select a different block size for tape or DASD by coding DCB=BLKSIZE=*block size* in the corresponding data set definition (DD) statement. The minimum block size is 7 892 bytes; the maximum is 32 760 bytes.

Note: To include the block size specification in the tape label, specify the BLKSIZE parameter in the tape DD statement.

You can also use the following options to save dump space:

- Dump only the used space (the default if you do not use keywords ALLDATA or ALLEXCP), instead of all allocated space, in sequential and partitioned data sets or in data sets with a null DSORG field. For VSAM key sequenced data sets, the VALIDATE keyword (the default) dumps only the used data instead of all of the allocated space.
- Use the COMPRESS keyword.

  Notes:

  1. DFSMSdss ignores the COMPRESS keyword if you specify it during a logical data set dump for physical sequential extended-format data sets.
  2. If your tape drive has the improved data recording capability (IDRC) and you want to use hardware data compaction, you do not need to use the COMPRESS keyword with the DUMP command. If you want software compression, specify the COMPRESS keyword, but you do not need to specify DCB=TRTCH=COMP in the JCL. In most cases, hardware data compaction without software data compression gives the best performance. However, you can use software compression and hardware compaction at the same time.

- Perform incremental data set backup instead of volume backup. This reduces the amount of dumped data and decreases processing time.

Performance Considerations

This section provides tips for improving the performance of copy and dump operations.

DUMP

Dump to tape, where the larger block size reduces the number of I/O operations.

- Use OPTIMIZE(2), (3), or (4) to read more than one track per read operation. This results in the reading of two tracks, five tracks, or a full cylinder, respectively. The default, OPTIMIZE(1), reads one track at a time. OPTIMIZE(2), (3), or (4) results in less elapsed time and fewer I/O operations on the DASD device whenever the load on the tape channel is low enough and the tape speed is high enough to keep pace with the data being read from the DASD volume. In order to get the best performance with DFSMSdss and 3592's, you need to specify OPTIMIZE(4).
- Use the PARALLEL feature to simultaneously dump multiple DASD volumes.

  Guideline: Simultaneous dumping occurs only when the output goes to separate output devices. If the OUTDDNAME keyword specifies the same device, DFSMSdss runs the steps serially.

Concurrent Copy

To get an exact copy of your data at a specific time, do not update it during the concurrent copy (CC) initialization.

CC initialization includes the time DFSMSdss spends filtering data sets. Therefore, the more precisely you specify the data sets to be processed, the sooner the initialization is completed and the sooner you can update your data again. Here are some ways to reduce initialization time:

- Keep data to be dumped by one DUMP command cataloged in one catalog, if possible.
- Do not specify the DYNALLOC keyword if you do not need dynamic allocation for your data sets.
- Specify fully or almost fully qualified data set names. This reduces the amount of time that DFSMSdss spends searching the catalogs for data sets to process.
- Specify smaller groups of data sets to process together in a DFSMSdss operation.
- Minimize the use of wildcards with the INCLUDE keyword.
- Minimize the use of sophisticated BY filtering to determine the data to be processed.
- Ensure that DFSMSdss can obtain serialization on all data sets being processed.
- Specify WAIT(0,0) to prevent DFSMSdss from waiting for serialization when it cannot be obtained.
- Do not specify the NOTIFYCONCURRENT keyword if you do not need notification of each data set included in the CC session.
- Do not specify the SPHERE keyword if you are not processing VSAM spheres.
- Use the ADMINISTRATOR keyword or DASDVOL RACF protection (where applicable) to bypass authorization checks for each data set being processed.

- Ensure that the volumes containing the VTOC and catalog entries for the data sets to be processed have caching enabled. Also ensure that the catalogs involved are enabled for the in-storage cache (ISC) or the catalog data space cache (CDSC).
- Ensure that data sets being processed have not been migrated by DFSMShsm.

CC uses storage in the control unit cache and in the processor. Here are some ways to minimize the storage needed:

- Limit the amount of data included in the CC operation.
- Use CC during periods of low update activity (as most backups are currently done today).
- Concentrate the update activity in a subset of the data being processed by CC.

Concurrent Copy Storage Requirements

The concurrent copy (CC) support for the 3990 Model 6 Storage Control uses data spaces to contain track image copies of the data being processed by the DFSMSdss. MVS data spaces are backed by expanded storage and local paging spaces. The amount of expanded storage and local paging space required for CC usage is dependent on a number of variables. Based on simulations and test scenarios, a typical data space size is about 10% of the amount of data being dumped or copied with CC.

If your data space size exceeds this nominal value, you may need to consider the following planning guidelines for determining how much expanded storage or local paging space may be required for the following CC functions:

- Full volume and tracks copy; and full volume, tracks and physical data set dump operations:

  All volumes are processed on a track-by-track basis by DFSMSdss. The data space requirements can vary from 0% for a volume that has no updates during the DFSMSdss operation to 100% if the entire volume is updated before DFSMSdss can process it. For example, a 3390–3 that is 80% full (2671 cylinders) may require up to 2671 cylinders of data space storage if the volume is completely rewritten before DFSMSdss can process it. An example of this situation would be that a volume contains many VSAM data sets and a reorganization is done for all of the VSAM data sets on the volume while the CC job is being run for the volume.
- Logical data set copy and dump processing of non-VSAM data sets and nonindexed VSAM data sets (for example, VSAM ESDS), logical data set copy of indexed VSAM data sets (for example, VSAM KSDS), and logical data set dump of indexed VSAM data sets processed with NOVALIDATE are described as follows:

  These data sets are processed on a track-by-track basis by DFSMSdss. The data space is used to contain updates for tracks that have not yet been processed by DFSMSdss. The data space requirements can vary from 0% for a data set that has no updates during the DFSMSdss operation to 100% if the entire data set is updated before DFSMSdss can process it. For example, a 50–cylinder data set may require up to 50 cylinders of data space storage if the data set is completely rewritten before DFSMSdss can process it.
- Logical data set dump of indexed VSAM data set (for example, VSAM KSDS) processed with VALIDATE is described below:

  These data sets are processed with numerous accesses to sequence set information in the index component and track-by-track accesses to the data component. In all cases, update activity to either the data component or the

index component maintains a copy of the updated track in the data space until the track is either processed by DFSMSdss or the dump operation is ended for all data sets.

Index component tracks that do not contain sequence set information and data component tracks that are beyond the high used relative byte address are included in the CC operation but are never read by DFSMSdss. if those tracks are updated, they will remain in the data space for the duration of the dump operation for all data sets. If the data set has the sequence set information imbedded in the data component (using the IMBED attribute), no additional (nonupdated) tracks are maintained in the data space. If the data set has the sequence set information in the index component, then all index component tracks containing sequence set information will be maintained in the data space (whether they were updated or not) for the duration of the dump processing for the data set. For example, if the index for a VSAM data set is 20 cylinders and the data is 2500 cylinders, plan paging space of 20 cylinders for the index component.

Based on the update activity during the dump operation, plan to use a paging space of between 0 and 2500 cylinders for the data. The most data space is used when doing a complete reorganization while dumping the VSAM data set. This requires 2520 cylinders of space. If only 10% of the data will change during the operation, you will need 20 cylinders for the index and 250 cylinders for the data or 270 cylinders of paging space.

In using CC against aggregate groups, determine the data space storage requirements based on the expected update rate to the data sets during the dump operations. Failure to allocate sufficient local paging space may result in system failures due to insufficient paging storage.

Note: All storage requirements will be in addition to the working set of storage required by *all* other applications active (including all other CC operations) during the execution of the DFSMSdss CC operation.

Virtual Concurrent Copy Working Space

To perform a SnapShot copy, space is required on one or more volumes in the same RAMAC Virtual Array (RVA) subsystem as the source data set. Ensure that space is available before using the concurrent keyword on DFSMSdss commands referring to data sets on an RVA subsystem.

Allocation of some number of data sets to be used as working space by virtual concurrent copy is necessary. The naming convention for these data sets is:

`SYS1.ANTMAIN.S`*`sysname`*`.SNAP`*`nnnn`*

Variable *sysname* is the system identifier and *nnnn* is a four-digit decimal number in the value range 0001–9999. If the system identifier is eight characters, ‘S’ replaces the first character.

A numerically sequential catalog search is done for each data set, starting with SYS1.ANTMAIN.S*sysname*.SNAP0001 until a catalog locate error is encountered, indicating that the data set was not found. The working space data sets must be cataloged. Data sets with this naming convention beyond the data set which was not found will not be used as working space. Data sets must be allocated as physical sequential and must not be extended-format. The data sets may be SMS-managed or non-SMS-managed. If the catalog search for a working space data set indicates that the data set is multivolume, the data set will not be used as a working space data set.

System Data Mover (SDM) does not extend a working space data set. If secondary space allocation is desired, the user must extend the data set by filling it with data prior to starting the DFSMSdss job. SDM holds an enqueue for the data set while the working space is being used during a SnapShot operation; it then releases the enqueue after all usage of the data set is completed. A working space data set may be reallocated or extended when SDM does not have the data set enqueued. On subsequent use of the reallocated or extended working space data set by SDM, the new size of the data set will be used by SDM. Additional working space data sets may be added after ANTMAIN has completed the initialization process.

SDM uses these data sets the first time an out-of-working-space condition is encountered during a SnapShot operation. When this condition occurs, SDM refreshes the list of working space data sets by performing a catalog search that starts with SYS1.ANTMAIN.S*sysname*.SNAP0001.

The LRECL and block size may be any valid combination. The tracks within the data set are used as the target of SnapShot operations, and you should not try to access them using normal data access methods.

Note: It is recommended that the working space data sets be protected by security, such as RACF, to ensure that sensitive data is not made available to unauthorized users.

Data sets must be allocated on some volume in each RVA subsystem which will be used for virtual concurrent copy. If more than one device type is defined on the RVA subsystem, a working data set must be allocated on each device type that contains a data set that will be processed using DFSMSdss SnapShot support.

If each system and each device type for DFSMSdss jobs (with concurrent copy specified) runs simultaneously from more than one system and accesses data on the same RVA subsystem, it requires at least one work data set each. For example, DFSMSdss must allocate three working space data sets to process data on an RVA subsystem from three MVS systems, on devices of each device type containing data processed with concurrent copy.

The total size of all work data sets allocated on each RVA subsystem should be equal to or exceed the largest total amount of data to be processed in a single DFSMSdss COPY or DUMP command on that RVA. If there is insufficient space, the concurrent copy initialization for one or more data sets in the job fails.

Read DASD I/O Pacing

You can tune the performance of a system by pacing the DFSMSdss read DASD I/O operations. Pacing reduces the channel utilization and lets other I/O (for example, from the data base application) be processed in a more timely fashion. The pacing is done by waiting a specified amount of time before issuing each channel program that reads from DASD.

Note: The additional wait time does not apply to error recovery channel programs or concurrent copy I/O. The System Data Mover dynamically controls pacing for concurrent copy I/O.

Invocation from a Customer Program

The value of the READIOPACING parameter can also be controlled through the installation options exit, a product-sensitive programming interface intended for customer use.

Related reading: For additional information about the installation options exit, see *z/OS DFSMS Installation Exits.*

Shared DASD Considerations

Shared DASD presents volume and data set serialization problems not encountered in nonshared DASD environments. Care should be taken when you enlist data set operations if programs operating in another processor might be accessing the data sets at the same time.

A data set can be dumped from one processor while being processed from another. The dumped version may be partially updated on JES2 systems. This same exposure is present on a full dump operation.

Chapter 7. Restoring Data Sets

With the RESTORE command, you can restore data to DASD volumes from DFSMSdss-produced dump volumes, which are identified with the INDDNAME keyword.

The restore function is logical or physical, depending upon the dump volume. If the dump volume was made physically, a physical restore is made. If it was made logically, a logical restore is made. If the data was compressed when it was dumped, it is automatically expanded to its original form during the restore operation.

Using the ALLDATA or ALLEXCP keyword during a dump affects target data set allocation during a restore. Only used space is dumped for both a physical and logical data set dump unless the ALLDATA or ALLEXCP keyword is specified as part of the DUMP command.

When ALLDATA or ALLEXCP is specified, the total allocated space is dumped. During a physical data set restore, the target data set is allocated with the same amount of space as the source data set. During a logical data set restore (without the ALLDATA or ALLEXCP keyword), the target data set is allocated according to the amount of space used in the data set, thereby releasing unused space. If logical data set processing is used and the target data set must preserve the total allocation of the source, the ALLDATA or ALLEXCP keyword should be specified during the dump.

When ALLDATA or ALLEXCP is specified for an extended-format sequential data set, data beyond the last-used-block pointer is not retained. The target data set is allocated with the same amount of space as the source data set during a logical restore or copy operation.

As with a data set DUMP command, you can use filtering to select data sets for restore processing. DFSMSdss reads the entire dump data set once during a restore regardless of how much data is actually being restored. This will result in multiple tape mounts if the dump data set is on multiple tapes. See “Choosing Data Sets for Processing—Filtering” on page 18, for more information on filtering.

Note: Fully qualified names are required to restore the following data sets:

- VVDS
- VTOCIX
- SYS1.STGINDEX
- Integrated catalog facility catalogs
- OS catalog
- VSAM read-only data sets (temporarily exported with the INHIBITSOURCE parameter)

Related reading: For information about the automatic class selection (ACS) routines during DFSMSdss restore operations, See Appendix A, “ACS Routine Information,” on page 165.

Logical Data Set Restore

A logical data set restore is performed if you are restoring from a volume created with a logical dump operation and if you specify the DATASET keyword. For instance, the following RESTORE command generates a logical data set restore if the volume was created with a logical dump operation.

```
RESTORE INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER1.OLDDS))  -
        REPLACE
```

Note: DFSMSdss logical restore processing cannot be used to process partitioned data sets containing location-dependent information that does not reside in note lists or the directory.

Output Volume Selection

In most cases, specifying output volumes is optional for a logical data set RESTORE command. Output volume specification is required only if the data set:

- Exists and is to be restored to a volume that is different from the current location
- Does not exist and is to be restored to a volume that is different from the source volume.

Specify output volumes with the OUTDDNAME or OUTDYNAM keywords. An example of a logical data set RESTORE command with OUTDDNAME is:

```
RESTORE  -
        INDDNAME(TAPE) OUTDDNAME(DASD1)  -
        DATASET(INCLUDE(**))
```

When not specified, the volume on which the source data set currently resides is found from the catalog and is dynamically allocated. To do this, you must include the REPLACE keyword. This is particularly useful on a data set restore operation from a data-set-selection-by-catalog dump because you need not know where the data sets resided at dump time.

You can specify multiple output DASD volumes on a logical data set RESTORE command. This is required when all the data sets to be restored cannot fit on a single volume. An example of a logical data set restore operation with a spill volume specified is:

```
RESTORE  -
        INDDNAME(TAPE) OUTDYNAM((338001),(338002))  -
        DATASET(INCLUDE(PARTS.**))  -
        PCTU(80)
```

Note the use of the PERCENTUTILIZED (PCTU) keyword in the above example. With PERCENTUTILIZED, you can set a limit on the amount of space DFSMSdss can fill on the volume. When this limit is reached, subsequent data sets are allocated to other volumes. In the above example, PERCENTUTILIZED is used to specify that only 80% of the first target volume is to be filled. This leaves 20% free space for the data sets to extend, if necessary.

PERCENTUTILIZED is ignored for SMS-managed volumes.

Note: User data-set labels on DASD volumes are supported during a data set restore operation. However, either the data set on both the source and target volumes must have these labels, or neither must have them.

Restoring to Preallocated Target Data Sets

In some instances, you might want to control the placement of a data set on a volume when you restore it. Some data sets (such as data sets allocated by absolute track) have location-dependent data and must be preallocated. Others (such as catalogs) should be placed for performance reasons. For details on restoring such data sets, see "Restoring Indexed Sequential, Unmovable, Direct, and Absolute Track Data Sets" on page 73 and "Restoring Integrated Catalog Facility Catalogs" on page 72.

In order to use a preallocated data set, the REPLACE or REPLACEUNCONDITIONAL keyword must be specified. If the REPLACE keyword is specified, the preallocated target data set name must be identical to the source data set name. If the REPLACEUNCONDITIONAL keyword is specified and the RENAME or RENAMEUNCONDITIONAL keyword is also specified, the preallocated target data set name must match the new name filter criteria.

If a target data set is preallocated, it is scratched and reallocated if it is not large enough to contain the dumped data set. VSAM preallocated target data sets are also scratched and reallocated when:

- Any of the following source and target data set attributes do not match:
  - CI size
  - Record length
  - IMBED (only KSDS and key range data sets)
  - Key length (only KSDS and key range data sets)
  - REPLICATE (only KSDS and key range data sets)
  - SPANNED
- The preallocated target is multivolume and the space of the target data set on the first volume is not large enough to contain all of the dumped data.
- The data set was not defined as reusable and the high-used relative byte address (RBA) of a target VSAM KSDS is not 0.

You may use data set RESTORE to upgrade your standard format sequential data sets to large format data sets. To do this, simply preallocate a large format data set. When restoring a standard format sequential data set and a preallocated large format data set is found, the preallocated large format data set will be the target of the restore. If the preallocated large format data set is not large enough to hold the data being restored, it will be scratched and reallocated as a large sequential data set. When restoring a large format data set and a preallocated standard format sequential data set is found, the standard format sequential data set will be used and will be upgraded to a large format data set. If the preallocated standard format sequential data set is not large enough to hold the data being restored, it will be scratched and reallocated as a large format data set.

If a user wishes to downgrade a large format data set to a standard format sequential data set, restore the large format data set with no preallocated target, allocate a standard format sequential data set, and use a utility such as IEBCOPY to copy the data from the large format data set to the standard format sequential data set.

During logical restore, a compression is performed when partitioned data sets are restored to both like and unlike devices. If the partitioned data set is being restored to an unlike device, the device-dependent information (such as TTR pointers and note lists) is in a usable form after the restore. DFSMSdss is unable to resolve device-dependent information for all other data set types being restored to unlike devices.

The NOPACKING keyword is effective only for partitioned data sets. If NOPACKING is specified for preallocated partitioned data sets, the preallocated target must reside on the same or a like device. Processing is stopped for the data set if the target resides on an unlike device. The target is not deleted and reallocated.

Cataloging Data Sets During Logical Restore Processing

When you restore a data set, you might need to catalog it in the standard order of search or recatalog it in its original catalog. The CATALOG keyword catalogs the data set in the standard order of search. The RECATALOG(*) keyword catalogs it in the same catalog that points to the source data set.

When a data set is restored as an SMS-managed data set, it is cataloged using the standard order of search. The RECATALOG keyword is ignored.

Examples of the CATALOG and RECATALOG keywords in a logical data set

RESTORE command follow:

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER1.**))  -
        CATALOG
```

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER1.**))  -
        RECATALOG(*)
```

When a VSAM KSDS or key range data set is being restored to an unlike device, the data set must be cataloged in the standard order of search.

Renaming Data Sets during Logical Restore Processing

In addition to cataloging data sets when they are restored, you can rename restored data sets by using the RENAME keyword. For instance, you can code the following to rename a data set you are restoring:

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER2.OLDDS))  -
        RENAME(*.OLDDS,*.NEWDS)
```

Note: The RENAME keyword works only if the data set exists on the output DASD with the old name. If you really want to unconditionally rename a data set, use RENAMEU. Both VSAM and non-VSAM data sets can be renamed. The rules for renaming VSAM clusters are the same as for non-VSAM data sets. You can rename only clusters. DFSMSdss assigns a new name for the components of VSAM clusters. SMS considerations require DFSMSdss to ensure that VSAM component names resolve to the same catalog as the cluster name. DFSMSdss uses the cluster name as a guide to determine the component names. This applies equally to SMS and non-SMS data sets.

Expiration Date Handling during Logical Restore

For preallocated targets, the expiration date of the preallocated target is preserved. For non-preallocated targets, the expiration date is dependent upon whether the data set is VSAM or non-VSAM, whether the source data set is SMS-managed or not SMS-managed, and whether the target data set is SMS-managed or not SMS-managed. SMS also ensures that the expiration date conforms with the target's management class retention period.

Allocating to SMS

VSAM data sets: The expiration date from the source catalog entry is used to set the target's expiration date in both the catalog and the VTOC. For an indexed VSAM data set, the expiration date in the VTOC for the index component will be zero.

Non-VSAM data sets: The expiration date from the source VTOC is used to set the target expiration date in both the catalog and VTOC. If the expiration date violates the target's management class retention period, SMS will modify the date to conform with the management class.

Allocating to non-SMS

VSAM data sets: The expiration date from the source catalog entry is used to set the target's expiration date in the catalog. The target expiration date in the VTOC will be 99365. For an indexed VSAM data set, the expiration date in the VTOC for the index component will also be 99365.

Non-VSAM data sets: The expiration date from the source VTOC is used to set the expiration date in the target VTOC. If the target is cataloged, the expiration date in the catalog is set to the date from the source VTOC if the source data set is SMS-managed. If the target is cataloged, and the source data set is not SMS-managed, the expiration date in the catalog is not set.

Physical Data Set Restore

A physical data set restore is done if you are restoring from a dump volume created by physical dump processing and you specify the DATASET keyword. If the dump volumes resulted from a physical data set dump operation, you must do a physical data set restore or a tracks restore operation. A tracks restore operation can consist of a subset of the dump data.

Notes:

1. When you perform a physical restore of many data sets, there is an initial delay while DFSMSdss allocates the target data sets.
2. To prevent index components from being restored inadvertently, you must specify the fully qualified name of the cluster.

On a physical data set restore operation, data sets from one or more logical volumes can be restored to a single DASD volume. If you want to restore data sets from specific source DASD volumes, use the LOGICALVOLUME keyword to specify the volume serial numbers of the source DASD volumes you want to restore. For example:

```
RESTORE  -
       INDDNAME(TAPE) OUTDDNAME(DASD1)  -
       DATASET(INCLUDE(**)) LOGICALVOLUME(111111)  -
       REPLACE
```

The following data sets cannot be processed by physical data set DUMP or RESTORE operations:

- VSAM data sets not cataloged in an integrated catalog facility catalog.
- Page, swap, and SYS1.STGINDEX data sets.

Note: Data from a specific volume can be restored only to a DASD volume of like device type.

Output Volume Selection

For a physical data set RESTORE command, you must specify an output volume with the OUTDDNAME or OUTDYNAM keyword. A physical data set restore operation restores only to the first volume in a list passed in the OUTDDNAME or OUTDYNAM parameter.

Cataloging Data Sets During Physical Restore Processing

If you specify CATALOG on a physical data set restore operation, DFSMSdss creates catalog entries for single-volume, non-VSAM data sets that were allocated by DFSMSdss. The cataloging is done immediately after successful allocation of a data set. Failure in cataloging does not prevent the data set from being restored. A data set that was allocated and cataloged but encountered errors during the restore operation is neither uncataloged nor scratched by DFSMSdss. You must not specify the RECATALOG keyword for physical restore.

The catalog that DFSMSdss uses to catalog a data set is determined as follows:

- If the first qualifier of the data set name is an alias for a user catalog, the catalog pointed to is used for that data set.
- Otherwise, the master catalog is used.

DFSMSdss does not catalog VSAM data sets during physical restore processing. If the CATALOG keyword is specified, it is ignored when processing VSAM data sets. You should use the IDCAMS DEFINE RECATALOG command to catalog VSAM data sets that were allocated by DFSMSdss (not preallocated). To recatalog and later access the VSAM data set, the volume serial numbers for the target and source volumes must match and the data set must be cataloged in the same catalog from which it was dumped. The volume serial number and the catalog name are printed in message ADR4181 during restore.

Note: To catalog multivolume non-VSAM data sets, use the IDCAMS DEFINE NONVSAM command.

Chapter 8. Restoring Data Sets with Special Requirements

Some data sets have special requirements for being restored. The sections that follow describe some of the special cases you might encounter when you restore data sets.

Restoring Multivolume Data Sets and Restoring Data Sets Using Multiple Target Volumes (Spill Volumes)

Multivolume data sets from a logical data set dump tape can be restored either to a single volume or to multiple volumes. When they are not preallocated and the specified output volumes are different from the input volumes, multivolume data sets are restored to a single volume, space permitting. When multiple target volumes are specified, DFSMSdss selects target volumes as follows:

- If a target volume that has the same volume serial as the source volume is available and has adequate space, it is chosen.
- If a volume of the *same* device type is available, and if it has adequate space, it is selected.
- A volume of a *like* device type is selected if it has adequate space.
- A volume of an *unlike* device type is selected if it has adequate space.

If you are restoring a multivolume data set from a physical dump, be sure the segments from all volumes are restored with successive RESTORE commands. Restoring a portion of a multivolume non-VSAM data set to a preallocated data set is allowed only if the volume sequence numbers of the source and target data sets are the same.

A VSAM data set that has its index component defined on more than one volume (that is, a multivolume KSDS defined with the imbed attribute) should always be processed logically. If it must be processed physically, it should be treated as an absolute track allocation data set, and its extents restored to their original location. This can be accomplished by performing either a full-volume restore, or a tracks restore of the relevant tracks. If this procedure is not done, the index may become unusable.

During logical restores of VSAM data sets whose data and index components are on different source volumes, DFSMSdss preserves the volume spread if enough target volumes of like device types are specified.

Note: DFSMSdss preserves the volume spread by placing the data and index components on separate devices only if all of the following are true:

- The source data and index components reside on separate devices.
- The target data set is preallocated with the data and index components on separate devices.
- DFSMSdss does not need to scratch and reallocate the preallocated target data set.

See “Restoring to Preallocated Targets” on page 74 for details on when DFSMSdss scratches and reallocates target data sets.

Restoring Integrated Catalog Facility Catalogs

Integrated catalog facility user catalogs can be restored only to the same volumes (the same volume serial and the same device type) from which they were dumped. The component names of the source and target user catalog must be the same. In addition, you must specify the fully qualified name to restore a catalog.

Restore from a logical dump is generally the best way to restore a catalog because it restores user catalog aliases if they are present in the logical dump data set. For logical restore operations, user catalog aliases are restored as follows:

- If DFSMSdss allocated the user catalog, aliases are restored if the catalog is successfully restored.
- If the target catalog was preallocated and is not empty, aliases are not restored.
- If the target catalog was preallocated and is empty, aliases are restored.

A physical restore operation does not restore aliases, and physically dumped catalogs cannot be restored if they are open. In addition, if the entries in the catalog during the dump operation do not match the entries during the physical restore operation, some of the data sets may become inaccessible.

An integrated catalog facility user catalog can be restored dynamically. Catalog recovery jobs should be modified to include the IDCAMS ALTER LOCK command to lock the existing catalog before the DFSMSdss restore operation. After the recovery is complete, unlock the catalog using IDCAMS ALTER UNLOCK. The LOCK attribute on the dump tape is used if the catalog does not exist.

The following example shows the JCL used to restore an integrated catalog facility user catalog. If the master catalog is RACF-protected, RACF access to it is required, unless you have DASDVOL update access or the installation authorization exit routine bypasses authorization checking.

```
//STEPT007 EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPE  DD DISP=OLD,LABEL=(1,SL)
      VOL=SER=(A00760),DSN=PUBSEXMP.DUMP,
      UNIT=3590
//SYSIN    DD *
  RESTORE DS(INCL(TEST.CAT.PUBSEXMP)) -
    OUTDYNAM ((D9S060)) -
    REPLACE -
    INDDNAME (TAPE)
/*
```

Figure 2 on page 73 shows the printed output produced by the RESTORE command.

```
PAGE 0001     5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES     1999.211 14:54
   RESTORE                      -
      DS(INCL(TEST.CAT.PUBSEXMP)) -
      OUTDYNAM (D9S060)         -
      REPLACE                   -
      INDDNAME (TAPE)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'RESTORE '
ADR109I (R/I)-RI01 (01), 1999.211 14:54:46 INITIAL SCAN OF USER CONTROL STATEMENTS
                         COMPLETED.
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 1999.211 14:54:46 EXECUTION BEGINS
ADR780I (001)-TDDS (01), THE INPUT DUMP DATA SET BEING PROCESSED IS IN LOGICAL
                         DATA SET FORMAT AND WAS CREATED BY DFSMSDSS VERSION 2
                         RELEASE 10 MODIFICATION LEVEL 0
ADR442I (001)-FRLBO(01), DATA SET TEST.CAT.PUBSEXMP PREALLOCATED, IN CATALOG
                         SYS1.MVSRES.MASTCAT, ON VOLUME(S): D9S060
ADR489I (001)-TDLOG(02), CLUSTER TEST.CAT.PUBSEXMP WAS RESTORED
                          CATALOG        SYS1.MVSRES.MASTCAT
                          COMPONENT      TEST.CAT.PUBSEXMP
                          COMPONENT      TEST.CAT.PUBSEXMP.CATINDEX
ADR454I (001)-TDLOG(01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                          TEST.CAT.PUBSEXMP
ADR006I (001)-STEND(02), 1999.211 14:54:47 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 1999.211 14:54:47 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 1999.211 14:54:47 DFSMSDSS PROCESSING COMPLETE. HIGHEST
                         RETURN CODE IS 0000
```

Figure 2. Output from Restore of Integrated Catalog Facility User Catalog

Related reading

- For additional information on the LOCK attribute and the access authority, see *z/OS DFSMS Managing Catalogs.*
- For additional information about the installation authorization exit routine, see *z/OS DFSMS Installation Exits.*

Restoring Non-VSAM Data Sets That Have Aliases

DFSMSdss does not support INCLUDE filtering of non-VSAM data sets using an alias. To include a non-VSAM data set which has an alias for restore processing, you must use the data set's real name, as shown in the VTOC. DFSMSdss does not detect nor preserve aliases of non-VSAM data sets. You will need to redefine the aliases after the data set is restored.

Restoring Indexed Sequential, Unmovable, Direct, and Absolute Track Data Sets

One important use of DFSMSdss is restoring data sets that contain device-dependent information. In some cases, such data sets can be restored without preallocating the target data sets. In other cases, however, you must preallocate the target in order to restore the data set.

DFSMSdss does not support restoring Indexed Sequential data sets.

Restoring without Preallocated Targets

If an unmovable data set is not preallocated, DFSMSdss tries to allocate the data set on the same 'relative tracks from which it was dumped. If this allocation fails, the unmovable data sets are allocated to any available location if the FORCE keyword is specified.

When you specify the FORCE keyword and some of the data sets have truly location-dependent data, you should specify their names in the EXCLUDE parameter to prevent DFSMSdss from restoring them. Subsequently, you must either restore the data set onto a scratch volume or free up the area of the DASD

where these data sets were located on the source volume and rerun the restore operation. When restoring a direct, undefined data set and the target data set is not preallocated, DFSMSdss allocates the data set. The allocation may result in a new data set with a different configuration from the data set that was dumped (for example, fewer volumes). This situation may cause problems when processing the restored data set.

Restoring to Preallocated Targets

If you are restoring any of these types (unmovable, direct, or absolute track) of data sets by preallocating them, the size and location of the extents on the dump volume and on the restore volume should match. Restoring a data set to a larger preallocated data set can cause problems because of the extraneous data beyond the end of the original dumped data. If the preallocated data set is too small, DFSMSdss deletes it and reallocates a new data set. The allocation may fail, or it may result in a new data set with a different configuration (for example, fewer volumes). The latter situation can cause problems processing the data set.

For unmovable data sets (those allocated as ABSTR, PSU, POU, or DAU), a preallocated data set is restored if the extents match and the REPLACE or REPLACEUNCONDITIONAL keyword is specified. Even if the extents do not match, the data set is restored if you specify both the REPLACE and FORCE keywords or the REPLACEUNCONDITIONAL and FORCE keywords. When an unmovable data set cannot be restored, the extents of the data set on the source volume are listed so that you can take action to restore it.

Note: Indexed sequential data sets cannot be restored to unlike devices.

Restoring Direct Access Data Sets

When DFSMSdss restores direct data sets, several processing options can be used. Direct data sets can be organized by relative block address or by track-track record (TTR).

Relative block addressable direct access data sets can be processed block by block to like and unlike target devices if the block size fits on the target track. When the data sets are processed block by block, DFSMSdss updates the block reference count of dummy records contained in the relative block addressed direct access data sets. To process block by block, the direct access data sets must have neither a variable record format nor a standard user label.

TTR direct access data sets may become unusable if they are processed block by block. TTR and relative block addressable data sets can be processed track by track to like and unlike target devices whose track capacity is equal to or greater than the source. Block by block processing is more efficient because track by track processing to an unlike device of larger track capacity can leave some unused space on each track of the target data set.

Several DFSMSdss keywords implement the BDAM processing options:

AUTORELBLOCKADDRESS If the data set is accessed with Optional Services Code (OPTCD) indicating relative block addressing, it is processed as if it were specified in the RELBLOCKADDRESS subkeyword list, and processing is block by block. If your installation has many relative block address direct access data sets, you can use the DFSMSdss installation

options exit to turn on the AUTORELBLOCKADDRESS function.

RELBLOCKADDRESS — If the data set is specified in the subkeyword list, the data set is processed block by block.

TTRADDRESS — If the data set is specified in the subkeyword list, the data set is processed track by track.

FORCE — If the track capacity of the receiving volume is smaller than the source, FORCE may be required for variable or undefined length TTR-organized direct access data sets. These data sets may be unusable after restore and, if possible, should be restored to a like device. Use RELBLOCKADDRESS to restore relative block address direct access data sets to unlike devices.

Note: If you do not specify a keyword, data is moved to the target on a track by track basis.

Related reading

- For additional information about using BDAM processing options with the DFSMSdss keywords, see the *z/OS DFSMSdss Storage Administration Reference*.
- For additional information about using the installation options exit, see *z/OS DFSMS Installation Exits*.
- For information about using DFSMS macros for non-VSAM data sets, see *z/OS DFSMS Macro Instructions for Data Sets*.

Restoring an Undefined DSORG Data Set

The PROCESS(UNDEFINEDSORG) keyword permits logical data set restore of an undefined DSORG data set to an unlike device of larger track capacity. The restore yields a usable data set; however, some unused space might remain on each track of the target data set. It may not always be possible to restore all undefined DSORG data sets to an unlike device type, even when the unlike device type has a track capacity greater than or equal to the source device. For example, if the source device is a 3380, the output device is a 3390, and the data set's block size is less than 277 bytes, a track on the target cannot contain as much data as a track on the source, and message ADR366W (invalid track format) is issued.

Note: A data set with an undefined DSORG or with a block size of 0 cannot be restored to a device of smaller track capacity than the source.

Restoring a VSAM Sphere

With DFSMSdss, you can restore an entire VSAM sphere (base cluster and all associated alternate index clusters and paths). The SPHERE keyword causes DFSMSdss to restore the entire VSAM sphere. If the dump was also taken with the SPHERE keyword specified, you need to specify the SPHERE keyword and the base cluster name to restore the base cluster and the other components.

When you restore a sphere to a preallocated target, all components (base clusters, alternate indexes, and paths) of the sphere must be preallocated. DFSMSdss does not restore a sphere if only some parts of the sphere are preallocated.

An example of the RESTORE command with the SPHERE keyword is:

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(PARTS.VSAM1))  -
        SPHERE  -
        REPLACE  -
        PSWD(PARTS.VSAM1/MASTUPW1)
```

Restrictions for Restore Processing

- You can restore a sphere only if all parts of the sphere resolve to the same catalog.
- You may not need to rename all the parts in the VSAM sphere as you would with the copy function.
- Multiple path names to an alternate index are not supported. Only the last path name listed in the catalog is preserved.
- When restoring a sphere with one or more alternate indexes missing from the dump tape, DFSMSdss issues a message to indicate that the sphere was incompletely restored.

Restoring a Preallocated VSAM Cluster

Restoring a VSAM cluster that has been preallocated is allowed if the following are the same on the source and the destination volumes:

- The number of components on the volume
- The beginning relative byte address (RBA)
- The component names
- Catalog names

The size of the cluster must be equal to or greater than that on the source volume. (Only the tracks that were dumped are restored.)

You should ensure that the control interval size, allocation unit, and secondary allocation quantity are the same as in the initial definition.

Restoring the VVDS and the VTOCIX

To restore a VVDS or VTOCIX data set, you must specify the fully qualified data set name. The VVDS and the VTOCIX data set cannot be restored with other data sets in the same RESTORE command. VVDS and VTOCIX data sets should not be restored by data set as a normal recovery procedure. The VTOCIX data set is an extension of the VTOC and can be rebuilt using IBM's Device Support Facilities program (ICKDSF).

The VVDS is an extension of the VTOC and of the catalogs for the VSAM data sets on the volume. If it is restored by a data set restore operation, it is possible that some of these data sets can become unusable because of a mismatch between the catalog, the VVDS, and the VTOC. If this occurs, run the diagnose function of access method services to determine the extent of the problem and to take appropriate corrective action.

DFSMSdss/VVDS Manager does not support dumping a multiple-extent VVDS and restoring the VVDS to a nonpreallocated VVDS. DFSMSdss can restore to a nonpreallocated VVDS only when the source VVDS resides on one extent.

The user can consolidate the VVDS extents by doing the following:

- DFSMSdss dump the multiple-extent VVDS
- IDCAMS delete the multiple-extent VVDS
- Preallocate a single-extent VVDS
- DFSMSdss restore the multiple-extent VVDS into the preallocated, single-extent target VVDS.

Restoring a PDSE

DFSMSdss lets you restore a PDSE.

Restoring a Damaged PDS

During a logical restore, a PDS is monitored by DFSMSdss for conditions that are not normal. The following conditions are detected and reported:

- Missing high-key entry in the PDS directory
- Missing directory EOF
- Invalid member start TTR:
  - TTR points before directory EOF
  - TTR points after end of data set
- Missing member EOF. Each member of a partitioned data set is normally terminated by an EOF record.
- Invalid note or note list TTR:
  - Note pointing before the start of member data
  - Note pointing after the member EOF
  - Note pointing past the last valid record on a track
  - Note pointing to record 0 of a track

DFSMSdss notes all of these conditions with a message.

During compression, DFSMSdss repairs all missing high-key directory entries, missing directory EOFs, and missing member EOFs.

Invalid start TTRs prevent DFSMSdss from compressing data for that member. DFSMSdss translates all valid note and note list TTRs during compression.

Use the NOPACKING keyword to restore damaged partitioned data sets to same or like device target volumes. This results in an exact track-for-track image of the source data set. Obviously, no compression is performed in this case. During physical restore operations, DFSMSdss uses only track-level I/O. Therefore, no compression takes place against the PDS.

Restoring Data Sets in an SMS-Managed Environment

Use the RESTORE command to recover data sets in an SMS-managed environment. If the data set was dumped logically, it is recovered logically. If it was dumped physically, it is recovered physically.

As discussed earlier, an SMS-managed environment can contain both SMS-managed and non-SMS-managed data. The following sections discuss how you can use the RESTORE command to recover these data sets.

Programming Interface information

The following sections discuss variables available to automatic class selection (ACS) routines during DFSMSdss processing. This information is provided for guidance purposes only. It is not associated with any interface provided by DFSMSdss. For details on writing ACS routines, refer to *z/OS DFSMSdfp Storage Administration Reference*.

End of Programming Interface information

Converting Non-VSAM Data Sets to Multivolume

Programming Interface information

The number of volumes allocated for certain VSAM and non-VSAM data sets can be changed with VOLCOUNT keyword options. The output data set must be SMS-managed. Single-volume data sets can be converted to multivolume; multivolume data sets can be converted to single-volume; or the number of volumes allocated for multivolume data sets can be changed. Allocation depends on which VOLCOUNT keyword is selected, and on whether output volumes are specified.

Note: TTR-BDAM and unmovable data sets cannot be converted to multivolume with the VOLCOUNT keyword. If an existing multivolume TTR-BDAM or unmovable data set is encountered, a DADSM error occurs. Partitioned data sets (PDS and PDSE) cannot be made multivolume by the VOLCOUNT keyword. If an existing multivolume PDS or PDSE data set is encountered, it is converted to single-volume.

End of Programming Interface information

Restoring SMS-Managed Data Sets

When you use the RESTORE command in an SMS-managed environment, automatic class selection (ACS) routines are invoked. ACS routines are written for each installation by the installation's own storage administrator.

When you use the RESTORE command, you are in the ACS RECOVER environment.

DFSMSdss passes a data set's classes at the time of the dump to ACS as input, and the ACS routines can assign or override these input classes.

VSAM alternate indexes do not have SMS constructs of their own; they use the same constructs as the base cluster. When restoring alternate indexes as independent clusters (because you did not specify the SPHERE keyword on the DUMP and RESTORE commands), DFSMSdss passes null classes to ACS. If you want DFSMSdss to pass the base cluster's classes to ACS, you must invoke sphere processing by specifying the SPHERE keyword on the DUMP and RESTORE commands.

If the source data set is not SMS-managed and has no class names, DFSMSdss passes null classes to ACS. If the source data set is SMS-managed and you do not specify otherwise, DFSMSdss passes ACS the source data set's classes. If you specify what you want passed to ACS with the STORCLAS, MGMTCLAS,

NULLSTORCLAS, or NULLMGMTCLAS keywords, DFSMSdss passes ACS what you specify. In all cases, the ACS routines ultimately decide the classes assigned to the data set.

You can, however, force the storage class and management class you specify to be assigned to a data set by using the BYPASSACS keyword with the RESTORE command.

The following RESTORE command results in ACS routines determining the target classes, using the source classes as input:

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER12.**))
```

If you preallocate a data set and specify the REPLACE or REPLACEUNCONDITIONAL keyword, the preallocated data set's classes are used.

Related reading:

- For additional information about the variables available to ACS routines during restore processing, see "ACS Variables Available during RESTORE and CONVERTV Processing" on page 166.
- For additional information about using the RESTORE command to convert data to and from SMS management, see Chapter 11, "Converting Data to and from SMS Management," on page 127.
- For additional information about the ACS routines, see Appendix A, "ACS Routine Information," on page 165.

Changing Storage Class with the RESTORE Command

In some cases, you might want to pass ACS a storage class that is different from that of the source data set. You can specify the RESTORE command with the STORCLAS keyword to pass ACS a storage class name as follows:

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER12.**))  -
        STORCLAS(SCNAME1)
```

However, using STORCLAS does not guarantee that the data set is assigned the storage class you specify. It means only that the storage class you specified is passed to the ACS routines. Depending on how your installation's ACS routines are written, the storage class you specify can be ignored, assigned to the data set, or used in combination with other input variables to determine a new storage class for the data set.

RACF checks if the RESOWNER field of a given data set is authorized to define the data set with the given STORCLAS. Ensure that the RESOWNER field of the data set has the proper authority to use the indicated storage class.

To make certain that the storage class you specify is assigned to the data set, you can use the BYPASSACS keyword as follows:

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER12.**))  -
        STORCLAS(SCNAME1)  -
        BYPASSACS(**)
```

In this case, ACS is not invoked, and therefore the data set is assigned whichever storage class that you have specified with STORCLAS. If you do not use STORCLAS, the data set is assigned the storage class of the source data set.

To limit the use of BYPASSACS, an installation can set up a RACF class profile.

You can use the NULLSTORCLAS keyword in conjunction with the BYPASSACS keyword to make a data set non-SMS-managed. For example, the following specification of the RESTORE command causes the specified data sets not to be SMS-managed:

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER12.**))  -
        NULLSTORCLAS  -
        BYPASSACS(**)
```

Changing Management Class with Restore Processing

In addition to influencing a data set's storage class when you restore it, you can also give ACS input for assigning or overriding the data set's management class. By specifying MGMTCLAS, you can pass a management class name to ACS and, as with STORCLAS, ACS can ignore it, assign it to the data set, or use it in combination with other input variables to determine the data set's management class. By specifying NULLMGMTCLAS, you can pass a null management class to ACS, which may or may not assign a management class.

An example of the RESTORE command with the MGMTCLAS keyword is:

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER12.**))  -
        MGMTCLAS(MCNAME1)
```

As with STORCLAS, RACF checks if the RESOWNER field of a given data set is authorized to define the data set with the given MGMTCLAS. Ensure that the RESOWNER field of the data set has the correct authority to use the indicated management class.

Just as you can with STORCLAS, you can use MGMTCLAS with BYPASSACS to ensure that the data set is assigned the management class you specify. For instance:

```
RESTORE  -
        INDDNAME(TAPE)  -
        DATASET(INCLUDE(USER12.**))  -
        MGMTCLAS(MCNAME1)  -
        BYPASSACS(**)
```

You should ensure that the management class you specify with MGMTCLAS is valid, or you will get an error. Remember that BYPASSACS skips both the STORCLAS and MGMTCLAS ACS routines.

To limit the use of BYPASSACS, an installation can set up a RACF class profile.

When you influence or assign the management class of a data set, you also need to be careful that the data set resides in a storage group capable of providing for the management class attributes associated with the management class you specify. For instance, if a data set has a management class that makes it eligible for migration, it needs to reside in a storage group on which DFSMShsm does migration. Otherwise, the data set will never migrate. For this reason, you might have to change the storage class along with the management class to ensure that the data set resides on volumes that can accommodate its management class.

However, if you are having to continually override your installation's ACS routines, you should see your storage administrator about changes to the ACS routines that would make it possible to let SMS do its job.

Restoring SMS-Managed Data Sets Physically

In general, it is recommended that you use logical data set restore processing in an SMS-managed environment. If you use physical data set restore processing, you should be aware of the special rules for volume and SMS construct selection.

When restoring a non-SMS-managed user catalog on an SMS-managed volume or an SMS-managed user catalog on a non-SMS-managed volume, physical restore does ***not*** convert the catalog. Instead, DFSMSdss physical restore ensures that the user catalog looks exactly like the source catalog (SMS or non-SMS-managed) and then places the output volume in INITIAL status.

DFSMSdss physical data set restore processing is sensitive to the number of logical volumes in a dump data set. A DFSMSdss physical dump tape can contain multiple logical volumes. Because a physical dump operates at the track-image level, every volume from which data was dumped is on the tape in the form of a logical volume.

The following example shows how a dump tape can contain more than one logical volume:

```
DUMP  -
     DATASET(INCLUDE(**))  -
     INDYNAM((338001),(338002))  -
     OUTDD(TAPE)  -
     COMPRESS
```

If data sets are dumped from both volumes, two logical volumes are on the dump tape.

During physical data set restore processing, the SMS class selection is similar to logical data set restore processing. The source data set's SMS classes (if any) are used as input to the ACS routines. You can influence the classes selected for the target data set by using the STORCLAS, MGMTCLAS, NULLSTORCLAS, NULLMGMTCLAS, and BYPASSACS keywords.

The major difference in physical data set restore (as opposed to logical data set restore processing) is that all the data will be restored to the first volume you specify in the OUTDDNAME or OUTDYNAM keyword.

Note: If the specified target volume is SMS-managed, no non-SMS-managed data sets are restored; conversely, if the specified target volume is not SMS-managed, no SMS-managed data sets are restored.

Restoring GDG Data Sets

For generation data group (GDG) data sets, filtering on generations is supported. Generation names in relative generation number, dsn(n), can be specified in the INCLUDE and EXCLUDE parameters. The GDG base must be defined (cataloged) before restoring GDG data sets. Otherwise, messages indicating that catalog errors have occurred may be issued during the restore.

Restoring SMS-Managed GDG Data Sets

SMS-managed GDG data sets can be in any one of the following states:

- ACTIVE
- DEFERRED
- ROLLED-OFF

When restoring a GDG data set to SMS-managed storage, DFSMSdss does one of the following:

- Preallocated restore retains the status of the preallocated generation data set (GDS).
- Restore function places the GDS in DEFERRED status, if the TGTGDS keyword is not specified. DFSMSdss leaves the GDS in DEFERRED status to enable you to (1) roll it back as an ACTIVE generation or (2) leave it as DEFERRED.
- If the TGTGDS keyword is specified, the appropriate status is assigned to the data set as long as the requested target status does not violate rules of the generation data group. The default status of logical and physical data set restore operation is DEFERRED.

Restoring Non-SMS-Managed Data Sets

To restore a data set to a non-SMS-managed target volume, you can use the NULLSTORCLAS and BYPASSACS keywords on the RESTORE command. If you use these keywords, the data set is placed on a non-SMS-managed volume, regardless of whether or not the source data set was SMS-managed.

Extended-format data sets cannot be restored to non-SMS-managed target volumes during a physical or logical data set restore.

Data sets with DFM attributes (created by DFM/MVS) can be restored to non-SMS-managed target volumes but the DFM attributes will be lost and a warning message will be issued.

Related reading: For information about DFM, see the *z/OS DFSMS DFM Guide and Reference*.

Logical Restore of Data Sets with Phantom Catalog Entries

During a disaster recovery, the logical restore of data sets may be unsuccessful because of *phantom catalog entries*; that is, the target data set names are cataloged but the target data sets do not exist. This condition can occur if you have:

- Scratched the target volumes and have not deleted the catalog entries for the corresponding data sets
- Restored the catalogs before restoring the data sets
- Restored the target data sets to different volumes from the offline source volumes, and the target data sets have not been renamed

DFSMSdss provides two parameters to support disaster recovery operations: DELETECATALOGENTRY and IMPORT.

DELETECATALOGENTRY Keyword

DELETECATALOGENTRY tells DFSMSdss to do a DELETE NOSCRATCH operation for any phantom catalog entry for a target data set being restored.

Attention: DELETECATALOGENTRY should be used with extreme care. **Do not** use it if:

- Any volumes on the restoring system are varied offline. If you do, DFSMSdss does a DELETE NOSCRATCH for any data set being restored that exists on the varied offline volume. Then, when the volume is varied online, you will have two data sets: a cataloged, restored data set and an uncataloged original data set on the volume that was varied offline.

  Also note that if the volumes are varied offline, catalog messages will be issued for each cataloged data set informing you that the volume is offline and requesting that you reply with 'CANCEL' or a device name.
- The restoring system is sharing catalogs with another system but not sharing the data set volumes. If you do, DFSMSdss does a DELETE NOSCRATCH for any data set that is cataloged in the shared catalog on the other system but that is on a volume not available to the restoring system. After the restore, you may have two data sets: a cataloged, restored data set and an uncataloged original data set on a volume in the other system.

IMPORT Keyword

IMPORT specifies that you are restoring data sets that were dumped from a system other than the one into which the restore is being done. Because the data sets to be restored are new to the system, the usual source data set authorization checks are not done. If you are authorized to read the input dump data set containing the data sets being restored, you have the authority to read any data set being restored. DFSMSdss continues to ensure that you are authorized to create a new target data set or replace an existing one.

Logical Restore of Preformatted Empty VSAM Data Sets

During a Logical Restore of preformatted empty VSAM data sets, DFSMSdss opens the target data set in order to preformat it. Open processing requires the data set to be cataloged in the standard catalog search order. So, in order to restore a preformatted empty VSAM data set, the target data set must be cataloged in the standard catalog search order.

Chapter 9. Restoring Volumes

You can recover a volume or ranges of tracks from a full-volume dump operation. If the dump volumes resulted from a full dump operation, you can do a full or a tracks restore (that is, ranges of tracks) or a data set restore operation. If the dump volumes resulted from a tracks dump operation (that is, ranges of tracks), you must do a tracks RESTORE command, which can consist of a subset of the dump data. See Appendix B, "Linux-z/OS DFSMSdss Dump or Restore HOW-TO," on page 169 to learn how to use z/OS DFSMSdss to restore Linux for OS/390 or Linux for zSeries partitions and volumes.

An example of a full-volume restore operation is:

```
RESTORE  -
        INDDNAME(TAPE)  -
        OUTDDNAME(DASD1)  -
        PURGE
```

With the restore operation, you can copy the volume serial number to the output DASD with the COPYVOLID keyword. For example:

```
RESTORE  -
        INDDNAME(TAPE)  -
        OUTDDNAME(DASD1)  -
        COPYVOLID  -
        PURGE
```

Notes:

1. COPYVOLID is required if you are restoring an SMS-managed volume, unless the source and target volume serial numbers match.
2. Data set restore of VSAM extended-addressable data sets from a physical volume dump is not supported.

You must consider several factors when restoring volumes in an SMS environment. Before you start to restore a full volume, you must ensure that the status of the target volume is synchronized with its environment. For example, if the target volume is a non-SMS-managed volume, the volume must not be defined in a storage group. Conversely, if the target volume is an SMS-managed volume, the volume must be defined in a storage group. Finally, if the target volume is SMS-managed, then SMS must be active for the full-volume restore operation.

If you are using Record Level Sharing (RLS), be careful when restoring volumes with the FULL or TRACKS keywords. If the target volume has data sets on it that have retained locks or data in the coupling facility associated with them, a full-volume or tracks restore can result in data integrity problems.

Specifying Output Volumes

For a full or tracks restore, you must specify an output volume by using the OUTDDNAME or OUTDYNAM keywords.

The device type of the source volume used in the dump operation and the device type of the target volume used in the restore operation must be the same. However, the following exceptions are possible:

- Data from a smaller-capacity IBM 3380 model can be restored to a larger-capacity IBM 3380 model.
- Data from a smaller-capacity IBM 3390 model can be restored to a larger-capacity IBM 3390 model.
- Data from a minivolume or a virtual volume can be restored to a real volume of like device type, and vice versa, device capacity permitting.
- Data can be restored from a from a larger-capacity IBM 3380, 3390, or 9345 model to a smaller-capacity IBM 3380, 3390, or 9345 model, if you are restoring specific track ranges using the TRACKS keyword and if the range of data to be processed falls within the capacity of the output device.
- Data from a smaller-capacity IBM 9345 can be restored to a larger-capacity IBM 9345.

Note: When you perform a full-volume restore to a DASD that is shared between multiple systems, the DASD should be offline to all systems except the one performing the restore.

When doing a full-volume restore operation to DASD, DFSMSdss automatically corrects the free-space information on the volume and rebuilds, as necessary, the VTOC index. DFSMSdss does this whenever it copies data to a larger-capacity DASD from a tape dumped from a smaller-capacity DASD or whenever both of the volumes, including volumes of equal capacity, contain a VTOC index. DFSMSdss allocates a large (more than 65 535 tracks) dummy data set to recalculate the free-space information; ignore any IEC614I messages that DFSMSdss generates as part of this procedure.

During a full-volume restore operation, other jobs may be enqueued on the output volume. If so, DFSMSdss cannot enqueue on the output volume to perform the full-volume restore operation. To determine if the output volume is allocated before performing the full-volume restore operation, issue the following operator command:

```
D U,DASD,ALLOC,cuu,1
```

This command displays the specified volume and the names of the jobs enqueued on it.

If, for example, the catalog has enqueued on the output volume, you can take the following steps:

1. Use the following catalog modify command to display a list of all open catalogs:

```
MODIFY CATALOG,LIST
```

2. Use the following catalog modify command to make CAS unallocate the catalog:

```
F CATALOG,UNALLOCATE(catname)
```

   If no other allocated catalogs are on the volume and the volume is not allocated by any other users, you can proceed with your full-volume restore.

For more information on CAS allocation, refer to *z/OS DFSMS Managing Catalogs*.

Processing RACF-Protected Data Sets

On a physical restore operation, DFSMSdss does not delete the profiles of RACF-protected data sets on the volume before a full restore operation. After a full restore, RACF profiles are not built for RACF-indicated data sets on the restored volume. If RACF data set profiles do not exist for these data sets, these data sets are inaccessible until RACF profiles are built for them.

If you use the COPYVOLID keyword to change the volume serial number or if the volume serial for the dump volume and the restored volume are different, DFSMSdss does not build profiles for the RACF-protected data sets on the restored volume or for the RACF DASDVOL for the RACF-protected DASD volume.

The protection status of data sets that are restored through a full restore is unpredictable if:

- RACF profiles (generic or discrete) of the data sets were changed between dump and restore functions.
- The dump was produced on a system (that supports RACF generic profiles) other than the one used for the restore.

Recovering System Volumes

You can use the Stand-Alone restore program of DFSMSdss to perform either a full or a tracks restore from the first data set of DFSMSdss-produced dump tapes, without the use of a host system environment. Using Stand-Alone restore you can recover system volumes in order to bring up the host environment.

You can also use Stand-Alone restore in a VM environment. Stand-Alone restore operates in ESA/370 mode, System/390 mode, System/370 XA mode, or System/370 mode.

Related reading: For additional information about how to perform the restore using the Stand-Alone Services, see the *z/OS DFSMSdss Storage Administration Reference*.

Recovering VM-Format Volumes

You can use DFSMSdss to recover VM-format volumes that are accessible to your MVS system. The volumes must have OS-compatible VTOCs starting on track zero, record five. DFSMSdss can only retrieve device information from the OS-compatible VTOC, and cannot interpret any VM-specific information on the volume.

Use the CPVOLUME keyword and specify the range of tracks to be restored with the TRACKS keyword. Because DFSMSdss cannot check access authorization for VM data, CPVOLUME is only allowed with the ADMINISTRATOR keyword.

Exercise caution when using DFSMSdss to recover VM-format volumes, because DFSMSdss does not serialize any VM data in any way. If you restore OS-format volumes to VM-format volumes or VM-format volumes to OS-format volumes, you must restore all of the volume's tracks. Failure to restore all of the tracks may render the volume unusable.

Chapter 10. Managing Data Movement with DFSMSdss

Data movement is necessary when you are doing the following tasks:

Replacing devices	When you remove devices to be replaced with other ones, you must move the data off the devices you are removing.
Adding devices	If you add new devices at your site, you must move data onto them to take advantage of the added capacity.
Maintaining devices	When you are servicing a volume, you might need to move data off the volume so users can continue to access the data.
Tuning performance	If a volume is performing poorly, it might be because data sets on the volume are being frequently accessed and causing an I/O bottleneck. In this case, you might move the data sets to another volume that is better able to handle it (either because it is less full or because it is cached).

You can use the DFSMSdss COPY command to move data between volumes.

Preparing for Data Movement

Before moving your data, determine the amount of space the data requires. You can determine this by building a data set or volume list with ISMF. A data set list indicates how much space is allocated for each data set and how much space it actually uses. A volume list indicates how much free space is on each volume in the list. You can use this information to calculate how much space the data to be moved requires and to ensure that enough free space exists on the target volumes. This calculation is especially important when combining multiple devices onto one larger-capacity device.

Note: In an SMS-managed environment, this calculation is unnecessary if enough DASD space is provided because the system finds the necessary free space and places the data for you.

Ensure that enough free space exists to contain the data, and back up the data before moving it to guard against its loss during the movement. You can use the DFSMSdss DUMP command to back up volumes or data sets.

Related reading: For additional information about using the DUMP command to back up data, see the following:

- Chapter 6, “Managing Availability with DFSMSdss.”
- *z/OS DFSMSdss Storage Administration Reference.*

Evaluating the Use of Logical and Physical Copy

As previously stated, you can use the COPY command to perform the actual data movement. However, you must determine whether to use logical or physical copy

function to move the data. The physical copy function gives you better performance, but the logical copy function allows you to move data to unlike devices.

Initiate a copy operation, logical or physical, at a time of low activity. Logical processing involves copying data sets, while physical processing involves volumes and tracks. Logical processing generally takes more time than physical processing.

Note: It is best not to specify the TOLERATE(ENQFAILURE) option when you move data with the COPY command. If you move data while updating it, you may lose the updates. Also, the TOLERATE(ENQFAILURE) option is not honored for a source HFS data set or a source zFS data set.

After DFSMSdss has finished processing, you can verify that the data has moved by looking at the ISMF data set or volume list.

Controlling What DFSMSdss Copies

DFSMSdss copies only used space for sequential or partitioned data sets and data sets with null DSORG fields (X'0000'), unless overridden by ALLDATA or ALLEXCP. Use the ALLDATA(*) and ALLEXCP keywords to process allocated space when the following conditions exist:

- You are not sure of the data set organization (DSORG) of a data set on a volume when doing a full-volume copy operation.
- There are sequential, partitioned, or individual data sets with a null DSORG field (X'0000') that is not accessed using SAM or PAM.

Note: The COPY command requires temporary work space. Ensure that public or storage volumes are available. Some temporary data sets are allocated to nonspecific devices by referring to SYSDA or SYSALLDA generic groups. If DFSMSdss is to function, these allocations must be allowed by the installation. Allocation validation exits must not restrict DFSMSdss allocations.

For temporary data sets allocated to nonspecific devices, DFSMSdss provides no unit type. SYSDA, SYSALLDA, or whatever is specified in the default allocation table is used. In an SMS-managed environment the default unit specified in the SMS base configuration table is taken even for non-SMS-managed temporary data sets.

See Appendix A, “ACS Routine Information,” on page 165 for information on automatic class selection (ACS) routines during DFSMSdss copy operations.

Moving Data Sets

Using the COPY command with the DATASET keyword, you can copy one or more data sets from one DASD volume to another of like or unlike device types. If you specify the DELETE keyword with the COPY command, the data set on the source volume is deleted after it has been successfully copied to the target volume. In this way, you can perform a data set move.

Note: Concurrent copy operation fails and a message is issued if the DELETE keyword is specified.

Specifying Input Volumes

The COPY DATASET command does not require that you specify input volumes. If you do not specify input volumes, data sets are selected from all the data sets cataloged in the standard search order.

When you specify input volumes using the LOGINDDNAME or LOGINDYNAM volume list, a data set is selected based on the following criteria:

- When you either specify SELECTMULTI(ALL) or specify input volumes without specifying the SELECTMULTI keyword, *all* of the volumes that contain a part of a non-VSAM or VSAM cluster must be in the volume list.

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:

  – Specify SPHERE and you must list all parts of the base cluster in the volume list.

  – Do not specify SPHERE and you must list all parts of the base cluster and the associated alternate indexes in the volume list.

- When you specify SELECTMULTI(ANY), *any part* of the non-VSAM data set or VSAM base cluster can be on a volume in the volume list.

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:

  – Specify SPHERE and you must list any part of the base cluster in the volume list.

  – Do not specify SPHERE and you must list any part of the base cluster and the associated alternate indexes in the volume list.

- When you specify SELECTMULTI(FIRST), the volume list must include the volume that contains the *first part* of either the non-VSAM data set or the primary data component of the base cluster for a VSAM sphere.

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:

  – Specify SPHERE and you must list the volume that contains the first extent of the data component for the base cluster in the volume list.

  – Do not specify SPHERE and you must specify the following information in the volume list:

    - The volume that contains the first extent of the data component for the base cluster.

    - The volume that contains the first extent of the data component for the associated alternate indexes.

Note: DFSMSdss, when processing input volumes, filters first based on the VTOC, and second, based on catalog filters, if specified.

If a data set is found on more than one specified input volume and the volume sequence numbers match, DFSMSdss cannot determine which data set is to be selected for processing. You do not need to specify a SELECTMULTI option when you build a list of input volumes with STORGRP. The volume list will contain all of the volumes in a storage group.

Selecting Output Volumes

Specifying output volumes is required for the COPY DATASET command in a non-SMS-managed environment. (For a discussion of SMS considerations for moving data, see “Moving SMS-Managed Data Sets” on page 110.)

You can specify multiple target volumes with the OUTDDNAME or OUTDYNAM keywords. This allows you to specify *spill* volumes. These spill volumes are used if the data sets you are moving require more space than is available on your first choice of volume.

If the output volume has unexpired data sets, you can either not process the data sets or write over them.

DFSMSdss now distinguishes between non-SMS and SMS volumes specified in the OUTDDNAME or OUTDYNAM keywords. For non-SMS allocations, only the volumes that are non-SMS are considered for allocation. Similarly, only SMS volumes are considered for SMS allocations.

The distinction between SMS and non-SMS is also used when determining the volume count for a multivolume allocation. Where volume count is determined from the number of specified volumes, only those volumes eligible for the type of allocation (SMS volume for SMS allocation or non-SMS volume for non-SMS allocation), processing proceeds with a null volume list.

There are several reasons for distinguishing between SMS and non-SMS volumes:

- Non-SMS volumes cannot be used for SMS allocations
- Specifying non-SMS volumes interferes with SMS guaranteed-space allocation
- Reducing volume count problems
- Improving the ability of DFSMSdss to process both non-SMS and SMS allocations in a single operation

Renaming Data Sets

You can rename data sets with the RENAMEUNCONDITIONAL (RENAMEU) keyword for the COPY command. For VSAM data sets, you can only rename clusters. DFSMSdss derives the new names for the components of VSAM clusters as follows:

- If the data set is a linear data set and the new cluster name matches the following convention:

  ```
  HLQ1.DSNDBC.HLQ3.HLQ4.%nnnn.%nnn
  ```

  and the old component matches the following convention:

  ```
  HLQ1.DSNDBD.HLQ3.HLQ4.%nnnn.%nnn
  ```

  where % is any single letter, nnnn is a 4-digit number, and nnn is a 3-digit number, then DFSMSdss generates the target component as follows:

  – If a qualifier from the source cluster name is identical to the corresponding qualifier of the source component name, the corresponding qualifier from the target cluster name is used in the target component name. Otherwise, DFSMSdss uses the qualifier from the source component name in the target component name.

  – DFSMSdss sets the sixth qualifier of the new component name to AD for a data component, or to AI for an index component whenever any of the following are true:

    - The new target component name exceeds 44 characters.
    - The new cluster name and new component name are identical.
    - The old component name and new component name are identical.

  - If the standard order of search directs the new target component name to a different catalog with the new cluster name, the following occurs:
    - DFSMSdss regenerates the target component name using the first five qualifiers of the new cluster name.
    - DFSMSdss appends a sixth qualifier of either AD for the data component or AI for the index component.
- If the following conditions are true:
  - The data set is a linear data set
  - DFSMSdss was invoked using the application interface
  - The UIM set the EI22DB2 bit ON
  - The new cluster matches the following convention:

    HLQ1.DSNDBC.HLQ3.HLQ4.%nnnn.%nnn

    where % is any single character, nnnn is a 4-digit number, and nnn is a 3-digit number.

  then DFSMSdss generates the target component name by using all of the qualifiers of the new cluster name as the corresponding qualifiers of the target component name, with the exception of the second qualifier. The second qualifier of the target component will be "DSNDBD."
- If the old component's name is equal to the old cluster name (plus any suffix), then the new component name will equal the new cluster name, plus the same suffix of the old component.

Example: When RENAMEU(NEW) is specified, the following configuration occurs:

	Cluster Name	Data Component Name	Index Component Name
Old	IBM.DFSMS.DSS	IBM.DFSMS.DSS.DAT1	IBM.DFSMS.DSS.INDX1
New	NEW.DFSMS.DSS	NEW.DFSMS.DSS.DAT1	NEW.DFSMS.DSS.INDX1

- If the old *and* new cluster names have "cluster" as their last qualifier, and the old component names match the cluster name up to the last qualifier, then the new component names will adhere to the old component naming convention.

Example: When RENAMEU(SYS2) is specified, the following configuration occurs:

	Cluster Name	Data Component Name
Old	SYS1.IODF00.CLUSTER	SYS1.IODF00
New	SYS2.IODF00.CLUSTER	SYS2.IODF00

- If the last qualifier of the new cluster name is "cluster," and the old component names do not match the cluster name up to the last qualifier, then new component names will be generated using the new cluster name and replacing the last "cluster" qualifier with "data" or "index."

Example: When RENAMEU(SYS2) is specified, the following configuration occurs:

	Cluster Name	Data Component Name
Old	SYS1.IODF00.CLUSTER	SYS1.DFSMS
New	SYS2.IODF00.CLUSTER	SYS2.IODF00.DATA

- If the new cluster name is less than or equal to 42 characters and the last qualifier is not “cluster”, DFSMSdss creates component names by adding a single character to the new cluster name: “D” for the data component, “I” for the index component.

Example: When `RENAMEU(SYS2)` is specified, the following configuration occurs:

	Cluster Name	Data Component Name
Old	`SYS1.IODF00.DATASET`	`SYS1.DFSMS`
New	`SYS2.IODF00.DATASET`	`SYS2.IODF00.DATASET.D`

- If the new cluster name is more than 42 characters and the last qualifier is not “cluster,” DFSMSdss derives the component names by doing the following:
  - Using up to the first four qualifiers of the new cluster name
  - Appending eight-character qualifiers, generated by using the time clock and system date, until the component names are five qualifiers

  Using up to the first four qualifiers of the new cluster name ensures that the component names will orient to the same catalog as the cluster.

Expiration Date Handling

When you copy a data set, the expiration date of the target data set is dependent upon whether:

- The data set is VSAM or non-VSAM.
- The source data set is SMS or non-SMS-managed.
- The target data set is SMS or non-SMS-managed.
- The source data set is cataloged or not cataloged.
- The SMS target’s expiration date matches the target’s management class.

SMS to SMS

The catalog expiration date and the expiration date in the Volume Table of Contents (VTOC) will have the same value as that of the source data set. For an indexed VSAM data set, the expiration date in the VTOC for the index component will be zero. If the expiration date is different than the target’s management class, SMS will modify the expiration date to match the target’s management class.

SMS to Non-SMS

The expiration date handling is dependent upon whether the data set is VSAM or non-VSAM:

- VSAM data set: The catalog expiration date will be the same as that of the source data set. In the VTOC, the expiration date is set to 99365. For an indexed VSAM data set, the expiration date in the VTOC for the index component will also be 99365.
- Non-VSAM data set: The catalog expiration date and the expiration date in the VTOC will have the same value as that of the source data set.

Non-SMS to SMS

The expiration date handling is dependent upon whether the data set is VSAM or non-VSAM:

- VSAM data set: The catalog expiration date and the expiration date in the VTOC will have the same value as the catalog expiration date of the source data set. For an indexed VSAM data set, the expiration date in the VTOC for the index

component will be zero. If the expiration date violates the target's management class, SMS will change the date to conform with the management class.

- Non-VSAM data set: If there is a catalog expiration date for the source data set, then the catalog expiration date is used for both the VTOC and the catalog expiration date of the target data set. If the source data set does not have a catalog expiration date or is uncataloged, then the VTOC expiration date for the source data set is used for both the catalog and the VTOC of the target data set. If the expiration date violates the target's management class, SMS will modify the date to conform with the management class.

Non-SMS to Non-SMS

The expiration date handling is dependent upon whether the data set is VSAM or non-VSAM and whether the source data set is cataloged or not cataloged:

- VSAM data set: The catalog expiration date is the same as that of the source data set. In the VTOC, the expiration date is set to 99365. For an indexed VSAM data set, the expiration date in the VTOC for the index component will also be 99365.
- Non-VSAM data set: The catalog expiration date of the source data set is used for the catalog expiration date of the target data set. The expiration date in the VTOC of the source data set is used for the VTOC expiration date of the target data set.

Defining RACF Profiles

For information on defining RACF profiles, refer to *z/OS DFSMSdss Storage Administration Reference*.

Moving Data Sets with Utilities

In some cases, DFSMSdss invokes a utility to move a data set. Table 7 on page 96 shows when DFSMSdss invokes a utility for a data set copy operation.

When you move a data set and a utility is used, the data set must be cataloged in the standard search order.

DFSMSdss cannot use fast replication methods to move data if a utility must be used. When FASTREPLICATION(REQUIRED) is specified in such a case, DFSMSdss will not use traditional I/O movement methods and therefore will not call the utility.

When DFSMSdss invokes IEBCOPY to copy a LOADMOD, message IEC507D is issued requesting operator authorization to overwrite an unexpired area when the source data set has an incorrect RLD count and an unexpired date.

When DFSMSdss invokes IEHMOVE to copy data sets, IEHMOVE has DD statement requirements that DFSMSdss cannot always satisfy. To avoid potential abnormal ends, do one or both of the following:

- Specify the source and target volumes as PRIVATE.
- Ensure that the source and target volumes are not in the list of default volumes for dynamic allocation.

When DFSMSdss invokes IDCAMS to copy a KSDS, the data set is automatically reorganized to optimize it for VSAM processing. A large KSDS may require extensive reorganization that could result in greater processing time for the copy operation. If IDCAMS was selected because multiple output volumes were specified, performance may be improved by specifying a single output volume for

the data set.

Table 7. Data Mover Selection Matrix for Data Set Copy

Data Set Type	Like Devices	Unlike Devices
Sequential	DFSMSdss	DFSMSdss
Partitioned (not PDSE)	DFSMSdss (1, 2)	DFSMSdss (1, 2)
Partitioned (not PDSE) load modules	DFSMSdss (3)	IEBCOPY
Partitioned data set extended (PDSE)	DFSMSdss (4)	DFSMSdss (4)
Direct nonrelative block address mode	DFSMSdss	DFSMSdss (5)
Direct relative block address mode (6)	DFSMSdss	DFSMSdss
ESDS	DFSMSdss (7)	DFSMSdss (7, 8)
RRDS	DFSMSdss (7)	IDCAMS (REPRO)
LDS	DFSMSdss (7)	IDCAMS (REPRO)
KSDS or VRRDS	DFSMSdss (9)	IDCAMS (REPRO)
Key range data set	DFSMSdss (10)	IDCAMS (REPRO)
Extended-format VSAM	DFSMSdss (7)	IDCAMS (REPRO)
Integrated catalog facility user catalogs	IDCAMS (EXPORT/IMPORT)	IDCAMS (EXPORT/IMPORT)
Undefined DSORG	DFSMSdss	DFSMSdss

Notes:

1. All partitioned data sets that are not load modules are compressed during a copy to a like or unlike device.
2. DFSMSdss calls the IGWFAMS utility when you are converting a PDS to a PDSE.
3. If copying partitioned load modules with REBLOCK, DFSMSdss calls IEBCOPY to copy the data set to a like device.
4. DFSMSdss calls the IGWFAMS utility when you are converting a PDSE to a PDS. DFSMSdss also calls IGWFAMS when all of the following conditions are met:
   - Fast replication methods cannot be used and FASTREPLICATION(REQUIRED) is not specified.
   - Concurrent copy cannot be used.
5. The source data set is not copied if the target data set is preallocated or if the target device has a smaller track capacity than the source.
6. Specify the DFSMSdss RELBLOCKADDRESS parameter.
7. DFSMSdss calls IDCAMS if the target CISIZE, CASIZE, physical record size, or physical block size of the target is different from that of the source.
8. DFSMSdss calls IDCAMS if the calculated number of blocks per control area is different from the calculated number of usable blocks per control area.
9. DFSMSdss calls IDCAMS if any of the following is true:
   - The CISIZE, CASIZE, physical record size, physical block size, imbed, or span attributes of the target are different from that of the source.
   - The target data set is SMS and has an imbedded index or has key ranges, and the target volume count is greater than one. Refer to the VOLCOUNT keyword in *z/OS DFSMSdss Storage Administration Reference* to determine the volume count.
   - The target data set is non-SMS, the source component or components span multiple volumes, and there is not enough space on one target volume to contain the entire data set.
10. DFSMSdss calls IDCAMS if the source and target CASIZE, physical record size, or physical block size are different; if the components span multiple volumes; for a KSDS with IMBED and either the source HURBA=HARBA or it has extended indexes.

Moving Data Sets with Concurrent Copy

Programming Interface information

The DFSMSdss concurrent copy (CC) function lets you move data but minimizes the time that the data is unavailable. The user determines an appropriate time to move the data (for example, when the data is in a known state and update activity is stopped). DFSMSdss is invoked directly or via the DFSMSdss application program interface (API) to do a CC of the data. After initialization is complete, DFSMSdss releases any serialization it held on the data sets and prints a message both to SYSPRINT and the console that the CC operation is logically complete. If DFSMSdss was invoked via the API, DFSMSdss informs the caller through a new UIM exit option, Eioption 24 (see the *z/OS DFSMSdss Storage Administration Reference* for details). The application can resume normal operation at this time.

End of Programming Interface information

If for any reason data cannot be processed with CC (for example, the hardware being used does not support CC), DFSMSdss uses methods of data movement and does not release the serialization until the copy is completed.

If the source device supports SnapShot, but does not support CC, DFSMSdss uses the SnapShot function to provide a CC-like function known as virtual CC.

Notes:

1. If DFSMSdss invokes a utility such as IDCAMS REPRO or IEBCOPY for a data set copy operation, CC is not done for those data sets.
2. The CONCURRENT keyword applies to all the data being dumped or copied by the function under which it is specified. It cannot be applied to a subset of the data being processed.
3. To improve data integrity, do not update the data during a CC initialization.
4. If a CC operation fails after signaling that the CC initialization is complete (and update activity on the data has resumed), it is not possible to recover the data at the point-in-time at which the CC operation has started. This is because the data may have been updated while the copy operation was progressing.
5. The CONCURRENT keyword cannot be used with the DELETE, UNCATALOG, or FASTREPLICATION(REQUIRED) keywords.
6. VM minivolumes are supported if you are using RVA devices to the extent that they are supported by IBM Extended Facilities Product (IXFP) device reporting.

Related reading: For additional information about virtual concurrent copy, see "Using Concurrent Copy" on page 5.

Moving Data Sets with FlashCopy

DFSMSdss can use FlashCopy to quickly move data from a source location to a target location when the following requirements exist:

- The source and target device types must be the same.
- The source devices and the target devices must be in the same ESS.
- The ESS must support data set FlashCopy (FlashCopy Version 2).
- The FASTREPLICATION(NONE) keyword is not specified.

- The data must not need manipulation. The following types of processing require data manipulation:
  - **Reblocking** — Reblocking occurs when you specify the REBLOCK keyword or when the VTOC indicates that the data set can be reblocked.
  - **PDS compression** — DFSMSdss compresses a PDS data set during copy processing, by default. You can specify the NOPACKING keyword to prevent DFSMSdss from compressing the PDS, thereby allowing the use of FlashCopy.
  - **Changing stripe counts** — The source stripe count must be the same as the target stripe count for a striped extended format data set.
  - **An individual stripe extending to more than one volume** — A single-striped sequential-extended format data set cannot use FlashCopy if either the source data set or the target data set is multivolume.
  - **PDS or PDSE conversion** — Conversion occurs when you specify the CONVERT keyword with these data sets.
  - **Block-by-block processing of direct access data sets** — Block-by-block processing occurs when you specify the REBLOCKADDRESS or the AUTORELOCKADDRESS keyword.
  - **Utilities** — FlashCopy cannot be used if your data must be moved with the use of a utility.

DFSMSdss attempts to allocate the target data set on the same device type in the same ESS if the source data is in an ESS. This increases the probability that FlashCopy can be used to copy the data. However, FlashCopy cannot be used if the source data set is multivolume and is not contained entirely in one ESS subsystem. The reason that FlashCopy cannot be used is because all source and target devices must be in one ESS subsystem in order to establish a FlashCopy relationship. These data sets will not be processed with FlashCopy and will be allocated to whatever volumes are available, irrespective of their FlashCopy capability.

Designating FlashCopy Usage

The FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword tells DFSMSdss how you want FlashCopy to be used. The default is FASTREPLICATION(PREFERRED).

FASTREPLICATION(REQUIRED) specifies that DFSMSdss must use FlashCopy to move data. If FlashCopy cannot be used, DFSMSdss issues error message ADR938E which indicates the processing of the current data set or that the entire COPY task failed. If the processing of the current data set failed, DFSMSdss does not try any other methods of data movement for the current data set and attempts to use FlashCopy for the subsequent data sets. If the entire copy task failed, DFSMSdss terminates the copy operation.

Restriction: You cannot use the FASTREPLICATION(REQUIRED) and CONCURRENT keywords together.

FASTREPLICATION(PREFERRED) specifies that you want DFSMSdss to use FlashCopy before any other method to move data (even when you specify the CONCURRENT keyword). If FlashCopy cannot be used and you have specified the CONCURRENT keyword, DFSMSdss attempts to use concurrent copy. If you have not specified the CONCURRENT keyword or if concurrent copy has failed, DFSMSdss uses traditional data movement methods to copy the data.

FASTREPLICATION(NONE) specifies that DFSMSdss not attempt to use FlashCopy to copy data.

Related reading: For additional information about the FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword, see the *z/OS DFSMSdss Storage Administration Reference.*

Determining Why FlashCopy Cannot be Used

There may be times whey you expect DFSMSdss to use FlashCopy to move the data but FlashCopy was not used. As far as you can tell, your data sets meet the criteria for FlashCopy use. Use the DEBUG(FRMSG (MINIMAL | SUMMARIZED | DETAILED)) keyword to help you resolve this situation. Include this keyword to indicate the applicable fast replication message level (MIN, SUM, or DTL) in your COPY command. The message level controls the type and amount of information that DFSMSdss provides.

DEBUG(FRMSG(MIN | SUM | DTL)) directs DFSMSdss to issue an informational message that indicates why FlashCopy was not used. When you specify FASTREPLICATION(REQUIRED), the informational message is issued in addition to the ADR938E message whether you have specified the DEBUG(FRMSG(MIN | SUM | DTL)) keyword or not.

Related reading: For additional information about the DEBUG(FRMSG(MIN|SUM|DTL)) keyword, see the *z/OS DFSMSdss Storage Administration Reference.*

Freeing Subsystem Resources

& Performing a physical copy of the data uses subsystem resources and can impact
& the performance of other I/O operations that are issued to the ESS. Using the
& FCNOCOPY keyword on a DFSMSdss COPY command prevents the ESS
& subsystem from performing a physical copy of the data. However, when you
& designate the FCNOCOPY keyword, you must either withdraw the FlashCopy
& relationship when you no longer need the copy or convert the existing FlashCopy
& relationship from FCNOCOPY to COPY mode.

& Withdrawing the FlashCopy relationship frees the subsystem resources that are
& used to maintain the FlashCopy relationship. You can withdraw the FlashCopy
& relationship by doing one of the following:

& • Perform a logical data set dump of the target data sets (of the data set copy) and
& specify the FCWITHDRAW keyword on the DUMP command.

& • Issue the TSO FCWITHDR command.

& When an existing FlashCopy relationship is converted from no-background copy
& (FCNOCOPY) to background copy mode, the relationship ends (unless the
& relationship is persistent) when the background copy has completed. When the
& relationship ends, it frees the subsystem resources that are used to maintain the
& FlashCopy relationship. You can change the existing FlashCopy copy mode by
& performing a logical data set copy specifying the FCNOCOPYTOCOPY keyword
& along with the source data sets for which you want background copy to be started.
& The FCNOCOPYTOCOPY function will initiate background copy of any NOCOPY
& FlashCopy relationships in which the specified source data sets are participating.

& In general, if you want a temporary copy of the data, specify FCNOCOPY, and
& then withdraw the FlashCopy relationship when you no longer need the copy. If
& you want a permanent copy, but want to delay background copy until a
& convenient time, specify FCNOCOPY to get a point-in-time copy and then perform

& FCNOCOPYTOCOPY later to start background copy. If you want a permanent
& copy and do not want to delay background copy, do not specify FCNOCOPY.
& Allow the ESS subsystem to perform the physical copy and release the subsystem
& resources that are used to maintain the FlashCopy relationship.

& **Note:** A Persistent FlashCopy relationship does not end when physical background
& copy has completed. The relationship can be removed by performing a
& Withdraw FlashCopy operation (e.g., TSO FCWITHDR command). An
& Incremental FlashCopy relationship is an example of a Persistent FlashCopy
& relationship supported by DFSMSdss. If you want to establish a Persistent
& FlashCopy relationship independent of Incremental FlashCopy, you can use
& the ESS Copy Services Web User Interface.

Related reading:

& • For additional information about using the FCNOCOPY, FCNOCOPYTOCOPY
& and FCWITHDRAW keywords, see the *z/OS DFSMSdss Storage Administration*
& *Reference*.
- For additional information about using the TSO FCWITHDR command, see the *z/OS DFSMS Advanced Copy Services*.
- For additional information about Persistent FlashCopy, see the IBM TotalStorage ESS User's Guide

Moving Data Sets with SnapShot

When the source and target devices are in the same RAMAC Virtual Array (RVA) and the data does not need to be manipulated (such as, reblocked, track packed to unlike), DFSMSdss may be able to use SnapShot to quickly move the data from the source location to the target location. SnapShot is much faster than traditional methods, especially when large amounts of data are moved.

To use SnapShot, the following requirements must be met:

- The source and target device types must be the same.
- The source and target devices must be in the same RAMAC Virtual Array (RVA).
- The FASTREPLICATION(NONE) keyword must not be specified.
- There must *not* be any required data manipulation. The following types of processing require data manipulation:
  - **Reblocking** — Reblocking occurs when the REBLOCK keyword is specified or when the VTOC indicates that the data set is capable of being reblocked.
  - **PDS compression** — DFSMSdss compresses a PDS data set during copy, by default. You can specify the NOPACKING keyword to prevent DFSMSdss from compressing the PDS, thereby allowing the use of SnapShot.
  - **Changing stripe counts** — The source stripe count must be the same as the target stripe count for a striped sequential-extended format data set.
  - **An individual stripe extending to more than one volume** — A single-striped extended format data set cannot use SnapShot if either the source data set or the target data set is multivolume.
  - **PDS or PDSE conversion** — Conversion occurs when you specify the CONVERT keyword with these data sets.
  - **Block-by-block processing of direct access data sets** — Block-by-block processing occurs when you specify the REBLOCKADDRESS OR the AUTORELOCKADDRESS keyword.
  - **Utilities** — SnapShot cannot be used if your data must be moved with a utility.

If the source data is in an RVA, DFSMSdss attempts to allocate the target data set on the same device type in the same RVA, thus increasing the probability that SnapShot can copy the data. If the source data set is multivolume and not contained entirely in one partition of one RVA subsystem, it is not possible to allocate the target so that SnapShot can be used. These data sets are allocated to whatever volumes are available, irrespective of their SnapShot capability.

Designating SnapShot Usage

The FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword tells DFSMSdss how you want fast replication such as SnapShot to be used. The default is FASTREPLICATION(PREFERRED).

FASTREPLICATION(REQUIRED) specifies that DFSMSdss must use fast replication such as SnapShot to move data. If SnapShot cannot be used, DFSMSdss issues error message ADR938E which indicates that the processing of the current data set or the entire COPY task failed. If the processing of the current data set failed, DFSMSdss does not try any other methods of data movement for the current data set. However, DFSMSdss attempts to use fast replication such as SnapShot for the subsequent data sets. If the entire copy task failed, DFSMSdss terminates the copy operation.

Restriction: You cannot use the FASTREPLICATION(REQUIRED) and CONCURRENT keywords together.

FASTREPLICATION(PREFERRED) specifies that DFSMSdss attempt to use SnapShot before any other method to move data (even when you specify the CONCURRENT keyword). If SnapShot cannot be used and you have specified the CONCURRENT keyword, DFSMSdss attempts to use virtual concurrent copy. If you do not specify the CONCURRENT keyword or if virtual concurrent copy fails, DFSMSdss uses traditional data movement methods to copy the data.

FASTREPLICATION(NONE) specifies that you do not want DFSMSdss to use SnapShot to copy data. Instead, DFSMSdss attempts to use virtual concurrent copy if the CONCURRENT keyword is specified. If virtual concurrent copy cannot be used, DFSMSdss uses traditional data movement methods to move the data.

Related reading For additional information about the FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

Determining Why SnapShot Cannot be Used

There may be times when you expect DFSMSdss to use SnapShot to move the data but SnapShot was not used. As far as you can tell, your data sets meet all the criteria for SnapShot use. Use the DEBUG(FRMSG (MINIMAL | SUMMARIZED | DETAILED)) keyword to help you resolve this situation. Include this keyword to indicate the applicable fast replication message level (MIN, SUM, or DTL) in your COPY command. The message level controls the type and amount of information that DFSMSdss provides.

DEBUG(FRMSG(MIN | SUM | DTL)) directs DFSMSdss to issue an informational message that indicates why SnapShot was not used. When you specify FASTREPLICATION(REQUIRED), the informational message is issued in addition to the ADR938E message whether you have specified the DEBUG(FRMSG(MIN | SUM | DTL)) keyword or not.

Related reading: For additional information about the DEBUG(FRMSG(MIN|SUM|DTL)) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

Moving Data Sets with Special Requirements

Some data sets require special treatment when they are moved. The following sections discuss some considerations for moving these special data sets.

Moving Undefined DSORG and Empty non-VSAM Data Sets

To copy a data set with an undefined DSORG, ensure that the following conditions are met:

- The PROCESS(UNDEFINEDSORG) keyword is specified.
- The selected target volume is either of the same device type as the source volume, or a device type with equal or greater track capacity.

To copy an empty non-VSAM data set, ensure that the following conditions are met:

- An EOF record exists in the first track of the source data set.
- If the target data set is to be SMS-managed, the selected target SMS volume must either be of the same device type as the source data set, or a device type with equal or greater track capacity.

Note: It may not be possible to move all undefined DSORG data sets to an unlike device type, even when the unlike device type has a track capacity greater than or equal to the source device. For example, if the source device is a 3380, the output device is a 3390, and the data set's block size is less than 277 bytes, a track on the target cannot contain as much data as a track on the source, and message ADR366W (invalid track format) is issued.

Moving System Data Sets

Some system data sets do not require movement, either because they are allocated during system generation or because they are built at IPL time. Other system data sets, however, can be moved by DFSMSdss for various reasons.

Unless excluded, system data sets are copied. However, they generally remain open while the system is running and cannot be scratched or uncataloged because the DELETE and UNCATALOG options apply only to data sets not in use.

Frequently, system data sets are prefixed with a high-level qualifier of SYS1. The PROCESS(SYS1) keyword can be used for a data set copy operation of a SYS1 data set to move it to a preallocated target or to copy it with the DELETE option. PROCESS(SYS1) does not apply to VTOCIX or VVDS.

To limit the use of the PROCESS keyword, you need to set up a RACF facility class profile. Refer to *z/OS Security Server RACF Security Administrator's Guide* for more information on RACF facility class profiles.

Note: The PROCESS(SYS1) option does not lift the restrictions on the processing of volume VVDSs or VTOC indexes.

When the PROCESS(SYS1) keyword is not specified, you cannot move system data sets the way you normally move data sets with DFSMSdss. In order for DFSMSdss to move system data sets, you must do one of the following:

- Dump the data sets, and then restore them to a different volume.
- Copy the data sets to a different volume and then catalog them in a different catalog.

Note: When a data set copy operation is used to copy the following data sets, space is defined for the target data set but no data is copied:

- Model DSCBs
- Page and swap data sets
- SYS1.STGINDEX.

Moving Catalogs

When you copy an integrated catalog facility user catalog, the DELETE keyword must be specified, but an input volume and the RENAMEU keyword must *not* be specified. You must specify the fully qualified name of the user catalog in the INCLUDE parameter. In any processor in the complex, there should be no other jobs executing that access the user catalog being moved; otherwise, the copy operation might fail or the copied catalog might contain errors.

You need RACF access if the catalog is RACF-protected.

User catalog aliases are automatically redefined after the copy. The LOCK attribute of an integrated catalog facility user catalog is preserved during the copy operation. Refer to *z/OS DFSMS Managing Catalogs* for a description of the LOCK attribute and the correct access authority.

Note: DFSMSdss cannot be used to move an active VSAM master catalog, integrated catalog facility tape volume catalogs (VOLCATALOG), the VVDS, or the VTOCIX.

Moving Non-VSAM Data Sets That Have Aliases

DFSMSdss does not support INCLUDE filtering of non-VSAM data sets using an alias. To include a non-VSAM data set which has an alias for copy processing, you must use the data set's real name, as shown in the VTOC. In most cases DFSMSdss does not detect or preserve aliases of non-VSAM data sets. However, during logical data set copy with the DELETE keyword specified and the RENAMEU keyword not specified, if the data set is SMS-managed and remains SMS-managed during the copy, any aliases associated with the data set are preserved. In all other cases, you must redefine the aliases after the data set is moved.

Moving Multivolume Data Sets

If you are specifying input volumes with the LOGINDDNAME or LOGINDYNAM keywords and you are moving multivolume data sets, use the SELECTMULTI keyword on the COPY command. SELECTMULTI allows you to move multivolume data sets in their entirety, even if you do not specify all the volumes on which the data set resides.

When you specify input volumes using the LOGINDDNAME or LOGINDYNAM volume list, a data set is selected based on the following criteria:

- When you either specify SELECTMULTI(ALL) or specify input volumes without specifying the SELECTMULTI keyword, *all* of the volumes that contain a part of a non-VSAM or VSAM cluster must be in the volume list.

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:

  - Specify SPHERE and you must list all parts of the base cluster in the volume list.
  - Do not specify SPHERE and you must list all parts of the base cluster and the associated alternate indexes in the volume list.
- When you specify SELECTMULTI(ANY), *any part* of the non-VSAM data set or VSAM base cluster can be on a volume in the volume list.

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:
  - Specify SPHERE and you must list any part of the base cluster in the volume list.
  - Do not specify SPHERE and you must list any part of the base cluster and the associated alternate indexes in the volume list.
- When you specify SELECTMULTI(FIRST), the volume list must include the volume that contains the *first part* of either the non-VSAM data set or the primary data component of the base cluster for a VSAM sphere.

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:
  - Specify SPHERE and you must list the volume that contains the first extent of the data component for the base cluster in the volume list.
  - Do not specify SPHERE and you must specify the following information in the volume list:
    - The volume that contains the first extent of the data component for the base cluster.
    - The volume that contains the first extent of the data component for the associated alternate indexes.

If a data set is found on more than one specified input volume and the volume sequence numbers match, DFSMSdss cannot determine which data set to select for processing.

You do not need to specify a SELECTMULTI option when you build a list of input volumes with STORGRP. The volume list contains all of the volumes in a storage group.

A multivolume data set can be copied to a single volume or to multiple volumes. For a multivolume data set with a standard user label, only the standard user label on the first volume is copied to the target volumes.

If you do not specify any input volumes, you can move multivolume data sets without any special keywords.

A DFSMSdss logical data set copy operation attempts to ensure that all parts of a multivolume non-VSAM data set exist. In cases where a part of the data set is missing, such as an inadvertent scratching of the VTOC entry on a volume, DFSMSdss issues an error message and discontinues processing the data set.

DFSMSdss cannot process the following non-VSAM data sets because they are missing one or more parts:

- Multivolume data sets whose catalog volume order differs from the VTOC volume order
- Single-volume data sets with the same name that are cataloged as one multivolume data set

- Multivolume data sets whose last volume indicator in the VTOC is not set

Note: When you are copying or restoring multivolume data sets, be aware of the following considerations:

- DFSMSdss does not preserve candidate volumes. However, for SMS-managed data sets, if you copy and do not specify any output volumes, DFSMSdss preserves the source volume count. If you copy and do specify the output volumes, DFSMSdss sets the volume count to the number of output volumes specified.
- DFSMSdss does not ensure that the copied or restored data set is on the same number of volumes as the original data set, nor does DFSMSdss ensure that the copied or restored data set extents are the same as the original data set. Instead, DFSMSdss tries to allocate the new data set on as few volumes as possible. This may result in the copied or restored data set becoming a single-volume data set.
- In addition, DFSMSdss tries to allocate each volume so that all data is contained in a single primary allocation of contiguous space with few, if any, of the secondary allocations being used.

Converting VSAM and Non-VSAM Data Sets to Multivolume

The number of volumes allocated for certain VSAM and non-VSAM data sets can be changed with VOLCOUNT keyword options. The output data set must be SMS-managed. Single-volume data sets can be converted to multivolume, multivolume data sets can be converted to single-volume, or the number of volumes allocated for multivolume data sets can be changed. Allocation depends on which VOLCOUNT keyword is selected, and on whether output volumes are specified.

Note: TTR-BDAM and unmovable data sets cannot be converted to multivolume with the VOLCOUNT keyword. If an existing multivolume TTR-BDAM or unmovable data set is encountered, a DADSM error occurs. Partitioned data sets (PDS and PDSE) cannot be made multivolume with the VOLCOUNT keyword. If DFSMSdss encounters an existing multivolume PDS or PDSE data set, it converts the data set to single-volume.

Moving VSAM Data Sets

When you move a VSAM data set and the REPLACE or REPLACEUNCONDITIONAL keywords are not specified, you must specify DELETE, RENAMEU, or RECATALOG (to a catalog different from the source catalog). If the REPLACE or REPLACEUNCONDITIONAL keyword is specified and a preallocated target is not found, DELETE, RENAMEU, or RECATALOG must be specified for the data set to be processed.

For VSAM data sets cataloged in an integrated catalog facility catalog that will be copied using the IDCAMS utility, a preallocated target data set will be renamed using a DFSMSdss-generated temporary name. This allows dynamic allocation and IDCAMS REPRO to work, because both are currently undirected in catalog usage.

VSAM data sets cataloged in an integrated catalog facility catalog with alternate index and path associations do not use a preallocated target if the DELETE keyword is specified. No search is made for existing data sets in this case. An integrated catalog facility alternate index cannot use a preallocated target. No search is made for existing data sets when copying an alternate index.

For VSAM components that are larger than one cylinder, DFSMSdss will recognize only an integral number of cylinders of free space on a target volume. Also, the required space for a VSAM data set must be contiguous.

You can move the base cluster, all associated alternate index clusters, and paths by using the SPHERE keyword with the COPY command.

Restrictions for the COPY command

The following information covers restrictions when using the COPY command:

- DFSMSdss must be able to invoke IDCAMS to copy an extended-format VSAM data set.
- When performing a logical copy operation of an extended-format data set, the target data set allocation must be consistent with the source data set allocation as follows:
  - If the source is extended-format VSAM, then the target must be extended-format VSAM.
  - If the source is extended-addressable VSAM, then the target must be extended-addressable VSAM.
  - If the source is a compressed-format VSAM KSDS, then the target must be a compressed-format VSAM KSDS.
  - If the source is an alternate index for an extended-format KSDS, then the target must be an alternate index for an extended-format KSDS.
  - If the source is an alternate index for a compressed-format KSDS, then the target must be an alternate index for a compressed-format KSDS.
  - The target control interval size must be equal to the source.
- You can copy a sphere only if all the parts of the sphere resolve to the same catalog.
- Multiple path names to an alternate index are not supported. Only the last path name listed in the catalog is preserved.
- To copy a sphere logically without the DELETE or RECAT keywords, you must rename every data set in the sphere. This includes all paths, all alternate indexes, and the base cluster. If the target sphere is to be SMS-managed, the data sets must be renamed even if the RECATALOG keyword is specified because the RECATALOG keyword is ignored for SMS-managed data sets.

If you do not use the SPHERE keyword and the base cluster has associated alternate index clusters, only the base cluster is moved as follows:

- If you specify DELETE, only the base cluster is moved, but the alternate index cluster continues to be related to the base cluster.
- If you do not specify DELETE, a second copy of the base cluster is created, and the alternate index cluster continues to be related to the original base cluster.

To move an alternate index cluster, specify DELETE on the COPY command. Only the alternate index cluster is moved, and it continues to relate to its base cluster. An alternate index cannot be moved by itself outside the environment of the base cluster. If the base cluster is not SMS-managed, the alternate index cannot be moved to an SMS-managed volume. If the base cluster is SMS-managed, the alternate index cannot be moved to a volume residing in another storage group.

For an empty VSAM data set (zero data relative block address or zero record count), the data set is defined on the target volume but is not copied. Message ADR474W is issued for the data set.

Note: DFSMSdss does not preserve candidate volumes during copy processing.

Moving a PDSE

The COPY command can be used to move a PDSE. The CONVERT keyword, along with the PDSE and PDS subkeywords, can be used with the COPY command to convert a PDS to a PDSE and vice versa.

Moving a Damaged PDS

DFSMSdss monitors PDSs during compression for conditions that are not normal. The following conditions are detected and reported:

- Missing high key entry in the PDS directory
- Missing directory EOF
- Invalid member start TTR
  - TTR points before directory EOF
  - TTR points after end of data set
- Missing member EOF (each member of a partitioned data set is normally ended by an EOF record)
- Invalid note or note list TTR
  - Note pointing before the start of member data
  - Note pointing after the member EOF
  - Note pointing past the last valid record on a track
  - Note pointing to record 0 of a track

DFSMSdss notes all these conditions with a message.

During compression, DFSMSdss repairs:

- Missing high key directory entry
- Missing directory EOF
- Missing member EOFs

Invalid start TTRs prevent DFSMSdss from compressing data for that member. DFSMSdss translates all valid note and note list TTRs during compression.

You can move damaged partitioned data sets to same or like device target volumes by using the NOPACKING keyword. This results in an exact track-for-track image of the source data set. Obviously, no compression is performed in this case.

Moving Unmovable Data Sets

When copying unmovable data sets to like devices, DFSMSdss places them at the same track locations on the target volume under the following conditions:

- The target volume has an indexed VTOC.
- The space where the unmovable data would be placed is available.

If any of these conditions do not exist, you must specify the FORCE keyword to move the data set. FORCE enables DFSMSdss to treat the unmovable data set as movable and to move it to an unlike device. Because DFSMSdss places the data set in any available location when FORCE is specified, use FORCE with caution.

If some data sets have CCHHR (cylinder, cylinder, head, head, record) location-dependent data and you are using FORCE, exclude these data sets with the EXCLUDE keyword to prevent DFSMSdss from moving location-dependent data sets.

Another way to position data sets in a specific location on a volume is to allocate all space on the target volume except where you plan to place the unmovable data sets. Then move the unmovable data sets with FORCE and afterwards scratch the dummy space allocation.

Moving Data Sets to Unlike Devices

DFSMSdss sets the secondary space to zero when processing data sets defined with the contiguous space attribute and zero secondary allocation. This action, which prevents DFSMSdss from creating an unusable data set, may result in ABEND D37-04 due to underallocation of the data set. Should this occur, the user must preallocate the target with adequate space to allow successful copy processing.

Moving Indexed Sequential Data Sets

DFSMSdss does not support the copy of Indexed Sequential data sets.

Moving Direct Access Data Sets

When DFSMSdss restores direct data sets, several processing options can be used. Direct data sets can be organized by relative block address or by track-track record (TTR).

Relative block addressable direct access data sets can be processed block by block to like and unlike target devices if the block size fits on the target track. When the data sets are processed block by block, DFSMSdss updates the block reference count of dummy records contained in the relative block addressed direct access data sets. To process block by block, the direct access data sets must have neither a variable record format nor a standard user label.

TTR direct access data sets may become unusable if they are processed block by block. TTR and relative block addressable data sets can be processed track by track to like and unlike target devices whose track capacity is equal to or greater than the source. Block by block processing is more efficient because track by track processing to an unlike device of larger track capacity can leave some unused space on each track of the target data set.

Several DFSMSdss keywords implement the processing options (refer to *z/OS DFSMSdss Storage Administration Reference* for details on their use):

AUTORELBLOCKADDRESS
If the data set is accessed with OPTCD indicating relative block addressing, it is processed as if it were specified in the RELBLOCKADDRESS subkeyword list, and processing is block by block. For additional information, refer to *z/OS DFSMS Macro Instructions for Data Sets* for macro instructions on non-VSAM data sets. If your installation has many relative block address direct access data sets, you may wish to consider the DFSMSdss installation options exit to turn on AUTORELBLOCKADDRESS (refer to *z/OS DFSMSdss Storage Administration Reference*).

RELBLOCKADDRESS
If the data set is specified in the subkeyword list, the data set is processed block by block.

TTRADDRESS
If the data set is specified in the subkeyword list, the data set is processed track by track.

FORCE If the track capacity of the receiving volume is smaller than the source, FORCE may be required for variable or undefined length TTR-organized direct access data sets. These data sets may be unusable after restore and, if possible, should be restored to a like device. Use RELBLOCKADDRESS to restore relative block address direct access data sets to unlike devices.

Note: If you do not specify a keyword, data is moved to the target track by track.

Moving GDG Data Sets

For generation data group (GDG) data sets, filtering on generations is supported. You can specify generation names in relative generation number, dsn(n), with the INCLUDE and EXCLUDE keywords. During a copy operation, if you catalog the GDGs in a different catalog or you rename them, you must predefine the target GDG base name because the source GDG base name is unusable.

Moving Generation Data Sets to SMS-Managed Volumes

An SMS-managed generation data set (GDS) can be in one of three states:

- ACTIVE
- DEFERRED
- ROLLED-OFF

When copying a GDS to an SMS-managed volume and the data set is not preallocated, DFSMSdss allocates the target GDS as follows:

- If DELETE is specified and RENAMEU is not specified, the target GDS is allocated with the same state as the source GDS.
- If the TGTGDS keyword is specified, the appropriate status is assigned to the data set. The requested target status must not violate rules of the generation data group.
- When the source is an SMS-managed GDS and the target has the same name (that is, DELETE without RENAME), the target status is the same as the source status.
- When the source is a non-SMS-managed GDS and the target has the same name (that is, DELETE without RENAMEU), the default target status is ACTIVE when the source is cataloged. When the source is not cataloged, the default target status is DEFERRED.
- In all other cases, the default target status is DEFERRED.
- You can use the TGTGDS keyword to alter the target status except when the source is an SMS-managed GDS and the target has the same name.

Table 8 on page 110 describes the default situation for DFSMSdss to allocate the SMS-managed GDG data set (MOVE refers to COPY command with the DELETE keyword specified):

Table 8. Default Situation for DFSMSdss to Allocate the SMS-Managed GDG Data Set

Target Environment	Source Environment	Source Status	DFSMSdss Function	TGTGDS Default
SMS	Non-SMS	Cataloged	COPY	DEFERRED
			MOVE	ACTIVE
		Not Cataloged	COPY	DEFERRED
			MOVE	DEFERRED
	SMS	ACTIVE	COPY	DEFERRED
			MOVE	ACTIVE
		DEFERRED	COPY	DEFERRED
			MOVE	DEFERRED
		ROLLED-OFF	COPY	DEFERRED
			MOVE	ROLLED-OFF

If the data set is preallocated, the state of the target GDS is not altered.

Moving Generation Data Sets to Non-SMS-Managed Volumes

A non-SMS-managed generation data set (GDS) can be in one of two states:

- Cataloged
- Not cataloged

When you copy a GDS to a non-SMS-managed volume, the state of the GDS is determined only by the CATALOG or RECATALOG keywords.

Moving SMS-Managed Data Sets

Programming Interface information

As with the RESTORE command, COPY invokes the Automatic Class Selection (ACS) routines, which in turn assign or override a data set's classes.

When you use the COPY command, you are in the ACS ALLOC environment. The storage class ACS routine is executed first. If the storage class assigned is not null, the management class ACS routine and then the storage group ACS routine are executed. (See "ACS Variables Available during Copy Function" on page 165 for a list of variables available to ACS routines during copy processing.)

If you do not specify otherwise, DFSMSdss passes the source data set's class names as input to ACS. If you want to specify storage and management class names to be passed to ACS, you can use the STORCLAS and MGMTCLAS keywords. You can use the NULLSTORCLAS and NULLMGMTCLAS keywords to pass null storage and management classes to the ACS routines.

VSAM alternate indexes do not have SMS constructs of their own; they use the same constructs as the base cluster. When copying or moving alternate indexes as independent clusters (because you did not specify the SPHERE keyword on the COPY command), DFSMSdss passes null classes to ACS. If you want DFSMSdss to pass the base cluster's classes to ACS, you must invoke sphere processing by specifying the SPHERE keyword on the COPY command.

If you do not want a data set to be SMS-managed, specify the BYPASSACS and NULLSTORCLAS keywords.

All of these keywords work the same for the COPY command as they do for the RESTORE command (see “Changing Storage Class with the RESTORE Command” on page 79 and “Changing Management Class with Restore Processing” on page 80).

End of Programming Interface information

Selecting Target Volumes

Programming Interface information

In an SMS-managed environment, you generally allow the system to place data sets for you. If for some reason you want to control the placement of the data sets (for example, because of performance problems or because you want to put data sets on some new, empty volumes you have just added to a storage group), you must take special steps.

If you use OUTDDNAME or OUTDYNAM to specify a volume list, the volume serial numbers are passed as input to the ACS routines. Depending on how your ACS routines are written, this input might or might not be used in determining where to place the data set.

One way to guarantee that data sets go to particular volumes is to write your storage group ACS routine such that data sets are moved to the volumes you select.

Alternatively, if a data set’s storage class has the guaranteed-space attribute, the data set is placed on the user-specified volumes if the volumes reside in the same storage group and ACS selects that storage group for the data set. By using BYPASSACS and STORCLAS keywords, you can ensure that the storage group selected contains the volumes you specify with OUTDDNAME or OUTDYNAM. However, for this procedure to work, your storage group ACS routine must use storage class to determine the storage group for a data set. This allows you to determine which storage class to specify with the STORCLAS keyword to ensure that the storage group containing the volumes specified with OUTDDNAME or OUTDYNAM is selected.

End of Programming Interface information

Changing Storage Class with Copy

Programming Interface information

You can use the STORCLAS keyword to specify a storage class name for DFSMSdss to pass to ACS. You can specify the NULLSTORCLAS keyword if you want DFSMSdss to pass a null storage class to ACS.

Note: RACF checks if the RESOWNER of a given data set is authorized to define the data set with the specified STORCLAS. Ensure that the RESOWNER of the data set has the correct authority to use the indicated storage class.

Using STORCLAS does not guarantee that the data set is assigned the storage class you specify. To ensure that the storage class you specify is assigned to the data set,

you must specify BYPASSACS. In this case, using BYPASSACS causes the storage class and management class ACS routines to be bypassed, so the data set is assigned whatever you have specified with STORCLAS or, if you do not use STORCLAS, whatever the source data set's storage class is. Ensure that the storage class you specify with STORCLAS is valid, or you will get an error.

You can also use STORCLAS and BYPASSACS to move data sets into a newly defined storage class. For example, suppose you want to combine all your storage classes except two into one new, large storage class. You can code the following:

```
COPY  -
    DATASET(INCLUDE(**)  -
            BY(STORCLAS,NE,(SCNAME1,SCNAME2)))  -
    STORCLAS(SCNAME3)  -
    BYPASSACS(**)  -
    DELETE
```

If you specify NULLSTORCLAS and BYPASSACS together, the target data set becomes non-SMS-managed.

End of Programming Interface information

Changing Management Class with Copy

Programming Interface information

In addition to influencing a data set's storage class with the copy command, you can also give ACS input for assigning or overriding the data set's management class. By specifying MGMTCLAS, you can pass a management class name to ACS and, as with STORCLAS, ACS ignores it, assigns it to the data set, or uses it in combination with other things to determine the data set's management class. By specifying NULLMGMTCLAS, you can pass null management class to ACS, which might or might not assign a management class to the data set.

Note: RACF checks if the RESOWNER of a given data set is authorized to define the data set with the specified MGMTCLAS. Ensure that the RESOWNER of the data set has the correct authority to use the indicated management class.

Also, just as with STORCLAS, you can use MGMTCLAS with BYPASSACS to ensure that the data set is assigned the management class you specify. Ensure that the management class you specify with MGMTCLAS is valid, or you will get an error. You must be authorized to use BYPASSACS and the management class you specify with MGMTCLAS.

End of Programming Interface information

Moving Non-SMS-Managed Data Sets

If the data set you are moving is to be non-SMS-managed, use the NULLSTORCLAS and BYPASSACS keywords on the COPY command. By using these keywords, you can make an SMS-managed data set non-SMS-managed. Using NULLSTORCLAS and BYPASSACS also prevents a non-SMS-managed data set from becoming SMS-managed.

Moving to Preallocated Data Sets

In some cases, you might want to copy data sets to preallocated targets. However, integrated catalog facility catalogs, and system data sets that are named SYS1.* cannot be copied to preallocated data sets unless the PROCESS(SYS1) keyword is specified.

Rules for Moving to Preallocated Target Data Sets

To use a preallocated data set, you must specify the REPLACE or REPLACEUNCONDITIONAL keyword. If the REPLACE keyword is specified, the preallocated data set name must be identical to the source data set name. If the RENAMEUNCONDITIONAL(newname) and REPLACEUNCONDITIONAL keywords are specified, the preallocated data set name must match the new name filter criteria. You cannot, however, copy a data set to a preallocated target data set with the same name within an SMS environment because SMS does not support duplicate data set names.

The rules for moving VSAM and non-VSAM data sets to preallocated data sets follow.

VSAM Preallocation: An existing data set qualifies as a preallocated target for a data set copy operation if the cluster name matches and the complete cluster is available on target volumes.

The preallocated data set is usable if all of the following conditions that apply to the data set being processed are met:

- The user is authorized to update the target data set.
- The cluster types match.
- The number of components match.
- The key length and offset match.
- The KEYRANGES match.
- None of the components are multivolume.
- Sufficient space is available for each component.
- Key sequential data sets (KSDS) are reusable or empty.
- Key range data sets are empty.
- The data set is cataloged in the standard order of search, if required for the copy operation.
- The data set has no alternate indexes or paths defined over it (except for a single path defined directly over the base cluster).

If a target data set is preallocated, it is scratched and reallocated when it is being renamed and:

- Any of the following source and target data set attributes do not match:
  - CI size
  - Record length
  - IMBED (only KSDS and key range data sets)
  - Key length (only KSDS and key range data sets)
  - REPLICATE (only KSDS and key range data sets)
  - SPANNED
- The data set was not defined as reusable and the high-used relative byte address (RBA) of a target VSAM KSDS is not 0.
- The target data set is not large enough to contain the source data set.

Non-VSAM Preallocation: An existing data set qualifies as a preallocated target for a data set copy operation if the data set names match, the complete data set is available on target volumes, and:

- For single-volume target qualification, the data set organization is partitioned *or* the data set's volume sequence number in the VTOC is 1 and the last volume flag is on.
- For multivolume target or single-volume target with the last volume flag off, the data set is cataloged in the standard order of search. All volume serial numbers returned by a locate operation on the data set are in the output volume list. (Candidate volumes are acceptable.)

Note: If a target data set is preallocated, but is not large enough to contain the source data set, it will be scratched and reallocated if it is being renamed.

You may use data set COPY to upgrade your standard format sequential data sets to large format data sets. When copying a data set and a usable preallocated target is found, it will be used as the target of the copy operation. When copying a standard format sequential data set and a preallocated large format data set is found, it will be used. If the preallocated large format data set does not have enough space for the source data, it will be scratched and reallocated as a large format data set. When copying a large format data set and a standard format sequential data set is found, it will be used and upgraded to a large format data set. If the preallocated standard format sequential data set does not have a large enough allocation to hold the source data, it will be scratched and reallocated as a large format data set.

If a user wishes to downgrade a large format data set to a standard format sequential data set, allocate a standard format sequential data set and use a utility such as IEBCOPY to copy the data from the large format data set to the standard format sequential data set.

The preallocated data set is usable if all of the following conditions that apply to the data set being processed are met:

- The user is authorized to update the target data set.
- The DSORG matches.
- For direct access data sets, the target does not exist if the copy operation is done using the IEHMOVE utility. If the RELBLOCKADDRESS keyword is specified for the data set, preallocated targets are allowed.
- For unmovable data sets, extents match exactly when you copy to a like device without specifying the FORCE keyword.
- For movable data sets or unmovable data sets with the FORCE keyword, the amount of allocated space in the target data set is greater than or equal to the amount of allocated space in the source data set.
- For partitioned data sets, the target directory can contain all source members and aliases.
- For preallocated standard user label data sets, the target has more than one extent when the source data set has more than one extent.

If a VSAM or non-VSAM preallocated data set is determined to be unusable, message ADR439E is issued, and the copy operation is stopped only for that data set. No attempt is made to clear or alter the target data set if:

- The source data set is empty.
- The DSORG is not supported.
- The target is preallocated but not empty.

Message ADR363E is issued to inform the user.

Specifying Multiple Target Volumes

When multiple target volumes and the REPLACE or REPLACEUNCONDITIONAL keyword are specified, more than one existing data set may qualify as a preallocated target. The first existing data set that qualifies as a preallocated target when you use the OUTDDNAME/OUTDYNAM list order is used as the target data set. For non-VSAM data sets that require catalog verification, the catalog standard order of search determines the data set used as the preallocated target.

The device-selection criteria used for the data set copy operation (same, like, then unlike device preference) is not observed if a preallocated data set target is used.

How Keywords Work with Preallocated Targets

When you use preallocated data sets with the COPY command, some keywords have a different effect and others have no effect at all.

ALLEXCP and ALLDATA: If ALLEXCP or ALLDATA is specified and the target is a like device, the data in the source data set is moved to the target. When ALLDATA or ALLEXCP is specified for an extended-format sequential data set, data beyond the last-used-block pointer is not retained.

CATALOG and RECATALOG: Data set copy operation cannot change the catalog or the catalog status (cataloged or uncataloged) of the preallocated target data set. As a result, the CATALOG and RECATALOG keywords have no effect on preallocated target data sets. (Similarly, passwords and expiration dates of preallocated data sets cannot be changed.)

NOPACKING: The NOPACKING keyword is effective only for partitioned data sets. If NOPACKING is specified for preallocated partitioned data sets, the preallocated target must reside on the same or a like device. Processing is stopped for the data set if the target resides on an unlike device. The target is not deleted and reallocated.

PERCENTUTILIZED: The PERCENTUTILIZED keyword has no effect when the target data set is preallocated.

PROCESS(SYS1): Data set copy operation permits moving SYS1 data sets to a preallocated target.

REBLOCK: If a data set qualifies for reblocking when REBLOCK is specified (sequential and partitioned only) and a preallocated target is used, the target block size is overwritten with one of the following values:

- The source data set block size
- A DFSMSdss-selected block size
- A user-selected block size passed by the installation reblock exit
- A system-determined block size

The block size used is determined by the installation reblock exit return code and the reblockable indicator for the data set VTOC entry.

If REBLOCK is not specified, the target BLKSIZE of a non-VSAM data set is overwritten with the source BLKSIZE.

If a partitioned data set is specified with both NOPACKING and REBLOCK keywords, the data set is not reblocked.

RENAMEUNCONDITIONAL: RENAMEUNCONDITIONAL has no effect on preallocated target data sets unless you have specified REPLACEUNCONDITIONAL.

Moving Data Sets Being Accessed with Record Level Sharing

During logical data set copy operations of SMS-managed VSAM data sets, DFSMSdss communicates with VSAM RLS to perform quiesce processing of data sets that are being accessed by another job using Record Level Sharing (RLS).

By default, DFSMSdss does not use timeout protection during RLS quiesce processing. You can control whether or not DFSMSdss uses timeout protection during RLS quiesce processing and what the timeout value should be using the DSSTIMEOUT parameter of the IGDSMS*xx* PARMLIB member.

You can also change the timeout value without IPLing the system using the SETSMS DSSTIMEOUT(*nnnnn*) command.

Related reading:

- For additional information about using the IGDSMS*xx* member of PARMLIB to control the RLS timeout value used during DFSMSdss operations, see the *z/OS DFSMSdfp Storage Administration Reference*.
- For additional information about using the SETSMS command, see*z/OS MVS System Commands*.

Moving Preformatted Empty VSAM Data Sets

When moving a preformatted empty VSAM data set, DFSMSdss opens the target data set in order to preformat it. Open processing requires the data set to be cataloged in the standard catalog search order. So, in order to copy a preformatted empty VSAM data set, the target data set must be cataloged in the standard catalog search order.

Moving Volumes

You can move volumes logically or physically with DFSMSdss.

As with moving data sets, if the output volume has unexpired data sets, you can stop the copy operation or write over the unexpired data sets.

Logical Volume Copy Operation

To move a volume logically, use the DATASET keyword, specify input volumes with LOGINDDNAME, LOGINDYNAM, INDDNAME, INDYNAM, or STORGRP, and use INCLUDE(**). This method of moving volumes allows you to move data between unlike devices.

Some data sets require special processing when you move them (see “Moving Data Sets with Special Requirements” on page 102). For example:

- Unmovable data sets
- Multivolume data sets
- Integrated catalog facility catalogs
- Data sets beginning with SYS1
- Data sets used by device-dependent application programs

If you use the COPY DATASET command to move a volume and the volume contains such data sets, you must move them in the correct sequence to achieve the expected results.

You may want to process unmovable data sets first, so you can place them at the same track location on the target device. Move user catalogs only when acquiesced. In addition, do not move catalogs together with the data sets cataloged in them.

See Appendix A, “ACS Routine Information,” on page 165 for information on automatic class selection (ACS) routines during DFSMSdss copy operations.

Note: Some data sets are not eligible for movement by DFSMSdss (for example, VSAM data sets not cataloged in integrated catalog facility catalogs). Others might require special parameters (for example, unmovable data sets).

Physical Volume Copy Operation

If you do not specify DATASET or TRACKS on the COPY command, the COPY command defaults to FULL and moves the volume physically. You must also specify INDDNAME or INDYNAM to indicate the source volume and OUTDDNAME or OUTDYNAM to indicate the target volume. Full-volume copy can move data only between like devices of equal or greater capacity (for example, from a double capacity 3380 model to a double or triple capacity 3380 model).

With full-volume copy, you can physically move volumes only between like devices. However, you can move data:

- From a smaller-capacity IBM 3380 to a larger-capacity IBM 3380
- From a smaller-capacity IBM 3390 to a larger-capacity IBM 3390
- From a smaller-capacity IBM 9345 to a larger-capacity IBM 9345
- From a minivolume or virtual volume to a real volume of like device type, and vice versa, device capacity permitting

With tracks copy, you can move data:

- From a larger-capacity IBM 3380 to a smaller-capacity IBM 3380, if the range of data to be processed falls within the capacity of the output device
- From a larger-capacity IBM 3390 to a smaller-capacity IBM 3390, if the range of data to be processed falls within the capacity of the output device
- From a larger-capacity IBM 9345 to a smaller-capacity IBM 9345, if the range of data to be processed falls within the capacity of the output device

When you perform a full-volume copy operation to a DASD that is shared between multiple systems, the DASD should be offline to all systems except the one performing the copy.

When you use the physical volume COPY command, you can specify the COPYVOLID keyword. If you specify the COPYVOLID keyword, the volume serial number of the source volume is copied to the target volume. This ensures that RACF profiles and catalog entries for the data sets on the volume have the correct volume serial number.

Note: Changing the volume serial number of a volume causes the operating system to demount the target volume at the end of the copy operation. To use the target volume, you must demount the source volume and mount the target volume.

If you are using record level sharing (RLS), be careful when copying volumes with the FULL or TRACKS keywords. If the target volume has data sets on it that have retained locks or data in the coupling facility associated with them, a full-volume or tracks copy can result in data integrity problems.

Related reading: For information about automatic class selection (ACS) routines during DFSMSdss copy operations, see Appendix A, “ACS Routine Information,” on page 165.

Moving Volumes with FlashCopy

FlashCopy is much faster than traditional data movement methods, especially when large amounts of data are moved. DFSMSdss can use FlashCopy during a full volume copy if the following requirements are met:

- The source devices and the target devices both support compatible levels of FlashCopy.
- The volumes must be in the same logical subsystem (LSS) of an ESS if the ESS supports only FlashCopy Version 1.
- The volumes must be in the same ESS.
- The FASTREPLICATION(NONE) keyword must not be specified.

For the best performance during full volume copy operations, specify the following keywords:

- ADMINISTRATOR
- ALLDATA(*)
- ALLEXCP
- PURGE

The performance improvement that is provided by these keywords is most significant when DFSMSdss uses FlashCopy or SnapShot to perform the copy.

Related reading: For additional information about how to use the ADMINISTRATOR, ALLDATA(*), ALLEXCP, and PURGE keywords, see the *z/OS DFSMSdss Storage Administration Reference*.

Designating FlashCopy Usage

The FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword tells DFSMSdss how you want FlashCopy to be used. The default is FASTREPLICATION(PREFERRED).

FASTREPLICATION(REQUIRED) specifies that DFSMSdss must use fast replication such as FlashCopy to move data. If FlashCopy cannot be used, DFSMSdss issues error message ADR938E and the copy operation fails. DFSMSdss does not try any other methods of data movement.

Restriction: You cannot use the FASTREPLICATION(REQUIRED) and CONCURRENT keywords together.

FASTREPLICATION(PREFERRED) specifies that DFSMSdss attempt to use FlashCopy before any other method to move data (even when you specify the CONCURRENT keyword). If FlashCopy cannot be used and you have specified the CONCURRENT keyword, DFSMSdss attempts to use concurrent copy. If you have not specified the CONCURRENT keyword or if concurrent copy has failed, DFSMSdss uses traditional data movement methods to copy the data.

FASTREPLICATION(NONE) specifies that DFSMSdss not attempt to use FlashCopy to copy data.

Related reading: For additional information about the FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

Determining Why FlashCopy Cannot be Used

There might be times whey you expect DFSMSdss to use FlashCopy to move the data but FlashCopy was not used. As far as you can tell, your volumes meet all the criteria for FlashCopy use. Use the DEBUG(FRMSG(MINIMAL | SUMMARIZED | DETAILED)) keyword to help you resolve this situation. Include this keyword to indicate the applicable fast replication message level (MIN, SUM, or DTL) in your COPY command. The message level controls the type and amount of information DFSMSdss provides.

DEBUG(FRMSG(MIN | SUM | DTL)) directs DFSMSdss to issue an informational message that indicates why FlashCopy was not used. When you specify FASTREPLICATION(REQUIRED), the informational message is issued in addition to the ADR938E message whether you have specified the DEBUG(FRMSG(MIN | SUM | DTL)) keyword or not.

Related reading: For additional information about the DEBUG(FRMSG(MIN | SUM | DTL)) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

Freeing Subsystem Resources

& Performing a physical copy of the data uses subsystem resources and can impact
& the performance of other I/O operations that are issued to the ESS. Using the
& FCNOCOPY keyword on a DFSMSdss copy command prevents the ESS subsystem
& from performing a physical copy of the data. However, when you designate the

& FCNOCOPY keyword, you must either withdraw the FlashCopy relationship when
& you no longer need the copy or convert the existing FlashCopy relationship from
& FCNOCOPY to COPY mode.

& Withdrawing the FlashCopy relationship frees the subsystem resources that are
& used to maintain the FlashCopy relationship. You can withdraw the FlashCopy
& relationship by doing one of the following actions:

& • Perform a full volume dump of the target volume and specify the
& FCWITHDRAW keyword on the DUMP command.

& • Issue the TSO FCWITHDR command.

& When an existing FlashCopy relationship is converted from no-background copy
& (FCNOCOPY) to background copy mode, the relationship ends (unless the
& relationship is persistent) when the background copy has completed. When the
& relationship ends, it frees the subsystem resources that are used to maintain the
& FlashCopy relationship. You can change the existing FlashCopy copy mode by
& performing a physical full volume or tracks copy specifying the
& FCNOCOPYTOCOPY keyword along with the source volume or extents for which
& you want background copy to be started. The FCNOCOPYTOCOPY function will
& initiate background copy of any NOCOPY FlashCopy relationships in which the
& specified source volume or extents are participating.

& In general, if you want a temporary copy of the data, specify FCNOCOPY and
& then withdraw the FlashCopy relationship when you no longer need the copy. If
& you want a permanent copy, but want to delay background copy until a
& convenient time, specify FCNOCOPY to get a point-in-time copy and then perform
& FCNOCOPYTOCOPY later to start background copy. If you want a permanent
& copy and do not want to delay background copy, do not specify FCNOCOPY.
& Allow the ESS subsystem to perform the physical copy and release the subsystem
& resources that are used to maintain the FlashCopy relationship.

Note: A Persistent FlashCopy relationship does not end when physical background copy has completed. The relationship can be removed by performing a Withdraw FlashCopy operation (e.g., TSO FCWITHDR command). An Incremental FlashCopy relationship is an example of a Persistent FlashCopy relationship supported by DFSMSdss. If you want to establish a Persistent FlashCopy relationship independent of Incremental FlashCopy, you can use the ESS Copy Services Web User Interface.

Related reading:

& • For additional information about using the FCNOCOPY, FCNOCOPYTOCOPY
& and FCWITHDRAW keywords, see the *z/OS DFSMSdss Storage Administration*
& *Reference*.

• For additional information about using the TSO FCWITHDR command, see the *z/OS DFSMS Advanced Copy Services*.

• For additional information about Persistent FlashCopy, see the IBM TotalStorage ESS User's Guide.

& Backing Up Volumes with FlashCopy Consistency Group

& **Creating Consistent Copies with FlashCopy Consistency Group:** You can use the
& FlashCopy Consistency Group function to minimize application impact when
& making consistent copies of data spanning multiple volumes. The procedure
& consists of freezing the source volume during each volume copy operation, and
& thawing all the frozen volumes using the CGCREATE command after a FlashCopy

& Consistency Group has been formed. During the time period between the first and
& the last volumes are frozen, no dependent write updates will occur which allows a
& consistent copy of logically related data that spans multiple volumes.

& **Freezing the Source Volumes in Copy Operations**: You can use the
& FCCGFREEZE keyword on the COPY FULL or COPY TRACKS CPVOLUME
& command to specify that the FlashCopy source volume is to be part of a
& FlashCopy Consistency Group. Subsequent I/O activity to the source volumes will
& be held (frozen) as each volume is copied. A frozen volume remains in long busy
& state until the ″Consistency Group Created″ (thaw) command is processed on the
& logical subsystem (LSS) where the volume resides or when the FlashCopy
& Consistency Group timer expires.

& **Thawing the Frozen Volumes in CGCREATE Operation**: When all volume copy
& operations have completed, you can use the DFSMSdss CGCREATE command to
& allow I/O activity to resume on the frozen volumes (thaw the volumes) residing in
& the logical subsystems. The required ACCESSVOL keyword specifies one or more
& volumes residing in the LSS to which the ″thaw″ command will be directed. Only
& one volume needs to be specified for each LSS containing frozen volumes in the
& FlashCopy Consistency Group.

& **Verifying the Consistency Group**: You can use the FCCGVERIFY keyword on the
& CGCREATE command to validate the state of the FlashCopy Consistency Group
& before thawing all the volumes. This will help you determine if the copies of the
& group of volumes are consistent. An error message is issued if the frozen state
& cannot be verified. Regardless of the verification result, DFSMSdss will proceed to
& thaw all the volumes in the designated logical subsystems.

& For the verification volume, IBM recommends that you select the first source
& volume that was copied with FCCGFREEZE in the group. When the logical
& subsystems have different Consistency Group timer values, select the volume
& residing in the LSS with the smallest Consistency Group timer value.

& **Example** In the following example, volume SRC101 and SRC102 reside on LSS 01.
& Volume SRC203 resides on LSS 02. LSS01 and LSS02 can be in the same or different
& storage control units. SRC101 is selected as the verification volume.

& • The first COPY command -- by default, in SERIAL mode -- will copy the
& verification volume, SRC101.

& • When the first COPY command completes, the PARALLEL command instructs
& DFSMSdss to switch to parallel mode. DFSMSdss will execute all subsequent
& commands in parallel until it reaches the SERIAL command which tells
& DFSMSdss to wait for all previous commands to finish before proceeding.

& • The user instructs DFSMSdss to verify the state of the FlashCopy Consistency
& Group using the specified verification volume during the ″thaw″ operation. The
& CGCREATE command should always be issued to thaw the volumes whether
& the copy commands completed successfully or not. In other words, the control
& statements do not need to check condition code prior to the CGCREATE
& command.

```
& //SYSIN
&    COPY FULL INDYNAM(SRC101) OUTDYNAM(TGT101) ADMIN DUMPCOND FCFREEZE
&    PARALLEL
&    COPY FULL INDYNAM(SRC102) OUTDYNAM(TGT102) ADMIN DUMPCOND FCFREEZE
&    COPY FULL INDYNAM(SRC203) OUTDYNAM(TGT203) ADMIN DUMPCOND FCFREEZE
&    SERIAL
&    CGCREATE FCCGVFY(SRC101) ACCVOL(SRC101,SRC203)
& /*
```

& **Notes:**

& 1. The freeze and thaw operations require the specified devices support the & FlashCopy Consistency Group function.
& 2. There is one Consistency Group timer per logical subsystem (LSS) for & FlashCopy.
& 3. The Consistency Group timer has a default of 120 seconds. The timer value can & be set via the ESS Web User Interface.
& 4. The CGCREATE operation is processed at LSS level. It thaws all the volumes & currently in ″frozen for consistency grouping″ state in the LSS that received the & command. When a ″thaw″ command is received by an LSS that does not have & any frozen volumes in a FlashCopy Consistency Group, the command is & accepted, but no actual processing takes place.
& 5. Multiple FlashCopy Consistency Groups with volumes in the same LSS must & not be formed at the same time.
& 6. When a FlashCopy Consistency Group timer expires before the ″thaw″ & command is received on the LSS, I/O activity will be allowed to resume on all & currently frozen volumes in the LSS. As a result, the copies of the volumes are & likely inconsistent.
& 7. When FCCGFREEZE is specified, if the FlashCopy pair failed to be established, & DFSMSdss will withdraw all FlashCopy relations previously established with & FCNOCOPY FCCGFREEZE option by the same DFSMSdss invocation. & DFSMSdss will also stop processing the rest of the COPY FCCGFREEZE & commands issued by the same invocation (e.g., in the same job step).

& **Related reading:**

& • For additional information about using the FCCGFREEZE keyword on the COPY & command, and the ACCESSVOL, FCCGVERIFY keywords on the CGCREATE & command, see the *z/OS DFSMSdss Storage Administration Reference*.
& • For additional information about FlashCopy Consistency Group, see the *z/OS* & *DFSMS Advanced Copy Services* and the IBM TotalStorage ESS User's Guide.

Moving Volumes with SnapShot

DFSMSdss can use SnapShot during a physical full volume copy operation when the source and the target devices are in the same RAMAC Virtual Array (RVA). SnapShot is much faster than traditional methods of data movement, especially when you are moving large amounts of data.

For the best performance during full volume copy operations, specify the following keywords:

- ADMINISTRATOR
- ALLDATA(*)
- ALLEXCP
- PURGE

The performance improvement that is provided by these keywords is most significant when DFSMSdss uses FlashCopy or SnapShot to perform the copy.

Related reading For additional information about how to use the ADMINISTRATOR, ALLDATA(*), ALLEXCP, and PURGE keywords, see the *z/OS DFSMSdss Storage Administration Reference*.

Designating SnapShot Usage

The FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword tells DFSMSdss how you want SnapShot to be used. The default is FASTREPLICATION(PREFERRED).

FASTREPLICATION(REQUIRED) specifies that DFSMSdss use fast replication such as SnapShot to move data. If SnapShot cannot be used, DFSMSdss issues error message ADR938E and the copy operation fails. DFSMSdss does not try any other methods of data movement.

Restriction: You cannot use the FASTREPLICATION(REQUIRED) and CONCURRENT keywords together.

FASTREPLICATION(PREFERRED) specifies that DFSMSdss attempt to use SnapShot before any other method to move data (even when you specify the CONCURRENT keyword). If SnapShot cannot be used and you have specified the CONCURRENT keyword, DFSMSdss attempts to use virtual concurrent copy. If you have not specified the CONCURRENT keyword or if virtual concurrent copy has failed, DFSMSdss uses traditional data movement methods to copy the data.

FASTREPLICATION(NONE) specifies that DFSMSdss not attempt to use SnapShot to copy data. Instead, DFSMSdss attempts to use virtual concurrent copy if the CONCURRENT keyword is specified. If virtual concurrent copy cannot be used, DFSMSdss uses traditional data movement methods to move the volume.

Related reading: For additional information about the FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

Determining Why SnapShot Cannot be Used

There may be times when you expect DFSMSdss to use native SnapShot to move the data but SnapShot was not used. As far as you can tell, your volume meets all the criteria for SnapShot use. Use the DEBUG(FRMSG(MINIMAL | SUMMARIZED | DETAILED)) keyword to help you resolve this situation. Include this keyword to indicate the applicable fast replication message level (MIN, SUM, or DTL) in your COPY command. The message level controls the type and amount of information that DFSMSdss provides.

DEBUG(FRMSG(MIN | SUM | DTL)) directs DFSMSdss to issue an informational message that indicates why SnapShot was not used. When FASTREPLICATION(REQUIRED) is specified, the informational message is issued in addition to the ADR938E message whether you have specified the DEBUG(FRMSG(MIN | SUM | DTL)) keyword or not.

Related reading: For additional information about using the DEBUG(FRMSG(MIN | SUM | DTL)) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

VTOC Considerations

When moving volumes, ensure that the VTOC on the target device is large enough to hold entries for all the data sets to be placed on the target device. The following two sections describe how the size of the target VTOC is affected by DFSMSdss processing.

You can also use the EXTVTOC or NEWVTOC functions of ICKDSF to extend or reallocate the VTOC on a volume if it is not large enough.

Note: When doing a full-volume copy operation to DASD, DFSMSdss automatically corrects the free-space information on the volume and rebuilds, as necessary, the VTOC index. DFSMSdss does this whenever the copy is to a larger-capacity DASD from a smaller-capacity DASD or whenever both of the volumes, including volumes of equal capacity, contain a VTOC index. DFSMSdss allocates a large (more than 65 535 tracks) dummy data set in order to cause the free-space information to be recalculated; ignore any IEC614I messages that DFSMSdss generates as part of this procedure.

Moving Volumes to Like Devices of Equal Capacity

When the source and target devices are of equal capacity, you can use logical or physical copy processing. If you use logical processing, the source VTOC is not copied to the target device. In this case, use ICKDSF to initialize the target device with an appropriately sized VTOC, then perform the logical data set copy operation.

If you use physical processing, the source VTOC is copied to the target device and DFSMSdss automatically rebuilds the VTOC index (if present on the source and target devices). If you determine that the source VTOC is not large enough for the target device, use ICKDSF to initialize the target device with an appropriately sized VTOC, then use logical data set copy to move the volume.

Moving Volumes to Like Devices of Greater Capacity

When the target device is of greater capacity than the source (for example, if you are moving from a 3390 Model 2 to a 3390 Model 3), you can use logical or physical copy processing. If you use logical processing, the source VTOC is not copied to the target device. In this case, use ICKDSF to initialize the target device with an appropriately sized VTOC, then perform the logical data set copy operation.

If you use physical processing, the source VTOC is copied to the target device if the target VTOC is within the range of the source device (for example, if you are copying a 3390 Model 2 to a 3390 Model 3 and the VTOC on the 3390 Model 3 starts at or before cylinder 2226). In this case, DFSMSdss automatically rebuilds the free-space information in the target VTOC or the indexed VTOC (if present) on the target device to account for the larger size. If you determine that the source VTOC is not large enough, do *one* of the following two things:

- Use ICKDSF to initialize the target device with an appropriately sized VTOC, then use logical data set copy to move the volume.
- Use ICKDSF to initialize the target device with an appropriately sized VTOC that is outside the range of the source device (for example, if you are copying a 3390 Model 2 to a 3390 Model 3, put the VTOC on the 3390 Model 3 at or after cylinder 2227), and then use full volume copy to move the volume. In this case, the size and location of the target VTOC are preserved and DFSMSdss automatically rebuilds the free-space information in the target VTOC or the indexed VTOC (if present).

Moving Volumes to Unlike Devices

When moving data between unlike devices, you must use logical processing. If you specify DATASET with the COPY command, DFSMSdss does a logical copy operation. To copy all the data on a volume logically, you also need to specify

input volumes with LOGINDDNAME, LOGINDYNAM, INDDNAME, or INDYNAM. LOGINDDNAME or LOGINDYNAM is required if you specify SELECTMULTI.

Moving VM-Format Volumes

You can use DFSMSdss to move VM-format volumes that are accessible to your MVS system. The volumes must have OS-compatible VTOCs starting on track zero, record five. DFSMSdss can only retrieve device information from the OS-compatible VTOC, and cannot interpret any VM-specific information on the volume.

Use the CPVOLUME keyword and specify the range of tracks to be copied with the TRACKS keyword. You can use concurrent copy to move the volume by specifying the CONCURRENT keyword. Because DFSMSdss cannot check access authorization for VM data, CPVOLUME is only allowed with the ADMINISTRATOR keyword.

Exercise caution when using DFSMSdss to copy VM-format volumes because DFSMSdss does not serialize any VM data in any way. You cannot copy VM-format volumes to OS-format volumes, nor can you copy OS-format volumes to OS-format volumes.

Chapter 11. Converting Data to and from SMS Management

DFSMSdss is the primary tool for converting data to and from SMS management. Conversion can be done with or without data movement.

This chapter is organized as follows:

- **"Evaluating Conversion to SMS Management"** discusses the advantages and disadvantages of the two types of conversion.
- **"Conversion by Data Movement" on page 128** describes how to use the COPY and DUMP/RESTORE commands to convert data sets *to* SMS management.
- **"Conversion without Data Movement" on page 129** describes how to use the CONVERTV command to convert volumes *to* SMS management.
- **"Special Data Set Requirements for Conversion to SMS" on page 132** describes some of the data sets that have special requirements for conversion *to* SMS management.
- **"Converting from SMS Management without Data Movement" on page 134** describes how to use the CONVERTV command to convert volumes *from* SMS management.
- **"Special Data Set Requirements for Conversion from SMS" on page 134** describes some of the data sets that have special requirements for conversion *from* SMS management.

Evaluating Conversion to SMS Management

When you convert data to SMS management, the first thing to consider is whether to convert data sets with or without data movement. If you have SMS-managed volumes with sufficient free space, you can convert data sets by simply moving them from non-SMS-managed volumes to SMS-managed volumes. The same is also true if you are converting data from SMS-management. Converting data sets to SMS management by data movement is often preferable because it allows the system to place the data sets for you. This ensures that the data sets are placed on volumes in storage groups that can meet the availability and performance requirements of the data set.

If, however, you do not have sufficient free space on your SMS-managed volumes to convert by data movement, you might have to convert data sets without data movement. The drawback to this method of conversion is that it does not allow the system to place data sets for you. You must ensure that the storage group in which you place the volume can meet the availability and performance requirements of the data sets.

Regardless of how you convert to SMS management, you must determine the eligibility for conversion of your data sets and volumes prior to conversion.

Data Sets Ineligible for Conversion to SMS

The following data sets cannot be converted to SMS management:

- Absolute track allocation data sets
- Direct with OPTCD=A
- GDS with candidate volumes
- Indexed sequential data sets
- Model DSCBs

- SYS1 storage index data sets (SYS1.STGINDEX)
- Indirectly cataloged data sets
- Uncataloged, multivolume data sets
- VSAM data sets not cataloged in an integrated catalog facility catalog
- VVDS/VTOCIX
- Unmovable data sets

Notes:

1. Using the CONVERTV command with the SMS and TEST keywords identifies ineligible data sets without actually converting any data.
2. VVDS/VTOCIX data sets can be SMS-managed, but DFSMSdss cannot be used to convert them, except when using the CONVERTV command to convert the volume that they are on.

Data Sets Ineligible for Conversion from SMS

The following data sets cannot be converted from SMS management:

- Extended-format sequential data sets
- Extended-format VSAM data sets
- Indirectly cataloged data sets
- VSAM data sets that have record level sharing (RLS) information associated with them
- VSAM base cluster or alternate index with a component with greater than 255 extents.

Notes:

1. Using the CONVERTV command with the NONSMS and TEST keywords identifies ineligible data sets without actually converting them.
2. VVDS/VTOCIX data sets can be non-SMS-managed, but DFSMSdss cannot be used to convert them, except when using the CONVERTV command to convert the volume that they are on.

Volumes Eligible for Conversion to SMS

A volume is eligible for conversion if it:

- Is a DASD volume
- Is permanently mounted and accessible online
- Has an indexed VTOC
- Is defined in an SMS storage group in an active configuration

Conversion by Data Movement

By using the logical data set COPY or DUMP/RESTORE command, you can move data sets between non-SMS-managed and SMS-managed volumes. When moving data sets to SMS-managed volumes, COPY and RESTORE commands invoke ACS to assign classes to the data sets. This type of conversion to SMS allows the data sets to be placed on the most appropriate SMS-managed volume.

Converting to SMS Management by Data Movement

When moving data sets to SMS-managed volumes, you can use the COPY or RESTORE command. You can specify storage and management class names with the STORCLAS and MGMTCLAS keywords. You can also specify output volumes with OUTDDNAME and OUTDYNAM. DFSMSdss passes the class names and volume serial numbers to ACS, which might use them in determining the classes and placement of the data set.

This method of converting data sets to SMS management is similar to moving data sets in an SMS-managed environment as described in “Moving SMS-Managed Data Sets” on page 110.

If you use the COPY or RESTORE command on a data set that is ineligible for SMS management and if a non-SMS-managed volume has been specified in the output volume list, DFSMSdss puts it on a non-SMS-managed volume. However, if you specify STORCLAS and BYPASSACS with the COPY or RESTORE command for a data set that is ineligible for SMS management, the copy or restore operation fails.

For data sets cataloged outside the standard order of search, use the INCAT keyword on the COPY or DUMP command to identify what catalog to search. Use the SELECTMULTI keyword on the COPY or DUMP command to convert multivolume data sets. This allows you to specify only the volume with the primary component on the LOGINDD or LOGINDY parameter. You can use the SPHERE keyword on the COPY DUMP/RESTORE command to convert entire VSAM spheres (if you use SPHERE on the RESTORE command, you must specify it on the corresponding dump as well).

Conversion from SMS Management by Data Movement

To take a data set out of SMS management with the COPY or DUMP/RESTORE command, you should specify the BYPASSACS and NULLSTORCLAS keywords. This forces DFSMSdss to make the data set non-SMS-managed.

Conversion without Data Movement

Conversion without data movement is divided into two phases: conversion of data sets and conversion of volumes. Convert data sets and the volumes they reside on without moving data by using the DFSMSdss CONVERTV command. You should set up RACF facility class authorization to limit the people who can use the CONVERTV command. When you use the CONVERTV command to perform conversion, it attempts to convert all the data sets on the volume. After all the data sets are processed, the volume is placed in one of the following three states:

- CONVERTED—the volume and its data sets are converted to SMS management. A volume can be placed in this state with the CONVERTV command and the SMS keyword.
- INITIAL—new allocations cannot be made to the volume and, although users can access their data sets, the data sets cannot be extended to other volumes. A volume may be placed in this state because you have used the CONVERTV command with the PREPARE keyword to reduce activity to the volume prior to conversion. A volume may also be placed in this state if you are attempting to convert it but it contains data sets that are not eligible for conversion.
- NONSMS—the volume and its data sets were taken out of the CONVERTED or the INITIAL state and are non-SMS-managed. A volume can be placed in this state with the CONVERTV command and the NONSMS keyword.

Simulating Conversion

Before you convert a volume to SMS management, you should simulate the conversion to ensure that all the data sets on the volume are eligible for conversion to SMS. In addition, simulating conversion shows you the classes ACS would assign to the data sets eligible for conversion.

You can simulate conversion by using the CONVERTV command with the SMS and TEST keywords. If the volume is ineligible for conversion, the data sets on the volume are still examined to determine their eligibility for conversion (provided the volume is permanently mounted and online).

When you use CONVERTV SMS TEST, you are in the ACS CONVERT environment. Only the storage class and management class ACS routines are executed. (See “ACS Variables Available during RESTORE and CONVERTV Processing” on page 166 for a list of variables available to ACS routines during CONVERTV processing.)

Simulated conversion creates a report that identifies data sets ineligible for conversion. For a sample of this report, see “SMS Report” on page 132. Note that this report indicates the management class and storage class that would be assigned to each data set. Careful analysis of this report allows you to determine if your ACS routines will assign appropriate classes to the data sets before doing the actual conversion.

Move data sets unsupported by SMS off the volume prior to actual conversion. Other data sets (for example, uncataloged data sets) can be made eligible for conversion by taking some action (for example, using the CATALOG keyword to catalog uncataloged data sets).

If you have ineligible data sets on a volume and you run the CONVERTV function with SMS, DFSMSdss still converts the eligible data sets on the volume. It then puts the volume in the INITIAL state. You must then take action to make the ineligible data sets eligible for conversion or move them off the volume. Once all the ineligible data sets are dealt with, you can run CONVERTV processing again to complete the conversion.

Preparing a Volume for Conversion

Before you convert a volume to SMS management, you should reduce the amount of activity to the volume being converted. The CONVERTV command with the SMS keyword automatically places the volume in a state of reduced activity before doing the actual conversion. You might, however, want to reduce activity without doing the actual conversion (for example, if you want to simulate conversion). Do this by specifying the PREPARE keyword on the CONVERTV command.

Specifying PREPARE prevents data sets from extending and new allocations from being made on the volume. However, users can still access the data on the volume from either the SMS system or a system sharing the volume.

When you use PREPARE, a report is generated that tells you the volumes that have been placed in the INITIAL state. If any of the volumes are ineligible to be placed in the INITIAL state, the report also lists them and the reason they were ineligible (for example, they did not have an indexed VTOC or were offline).

If you use the TEST keyword with PREPARE, you still get the report indicating which volumes would and would not be placed in the INITIAL state, but the PREPARE is not actually performed. You can then take some action to make those volumes eligible or simply not run PREPARE against those volumes.

The CONVERTV command with the NONSMS keyword reverses the effect of PREPARE and takes a volume out of the INITIAL state.

Converting to SMS Management without Data Movement

To convert data to SMS management, use the CONVERTV command with the SMS keyword. (Because SMS is the default for the CONVERTV command, you can simply specify CONVERTV.) Of course, the volume and all its data sets must be eligible for conversion to successfully run CONVERTV with SMS.

If the volume is eligible for conversion, the INITIAL indicator on the volume is set. This means the volume is in the same state as when you specify the CONVERTV command with the PREPARE keyword. When a volume has its INITIAL indicator set on, DFSMSdss begins processing the data sets on the volume.

If a data set is eligible for conversion, ACS is called to assign SMS classes to the data set. When you use the CONVERTV command with SMS, you are in the ACS CONVERT environment. The storage class ACS routine is executed first. If the storage class assigned is not null, the management class ACS routine is executed. (See "ACS Variables Available during RESTORE and CONVERTV Processing" on page 166 for a list of variables available to ACS routines during CONVERTV processing.)

RACF checks if the RESOWNER of a given data set is authorized to define the data set with the given STORCLAS, MGMTCLAS, or both. Ensure that the RESOWNER has the correct authority.

If no errors occur, the catalog entry for the data set is updated to include the classes. For VSAM data sets, the catalog entry is updated to indicate that it is SMS-managed. For non-VSAM data sets, a catalog entry is added that indicates the data set is SMS-managed. After the catalog updates and additions are successfully made, the data set's VTOC entry is updated to indicate it is SMS-managed.

If a VSAM data set has the guaranteed-space attribute, a check is done to verify the eligibility of its candidate volumes. If this check fails, the data set is not converted to SMS management. Non-VSAM data sets have candidate volumes in their catalog entries made nonspecific.

When DFSMSdss encounters a data set that is not eligible for conversion, it does not process the data set, but it continues to process other data sets on the volume. The only time conversion of data sets stops is when an error prevents ACS from returning class information for any data set.

DFSMSdss does not mark a volume as SMS-managed until all the data sets on the volume are SMS-managed. If a volume contains data sets that are ineligible for conversion, you must take some action to make them eligible or move them off the volume. You can then resubmit the CONVERTV command to convert any data sets not already converted and mark the volume as an SMS-managed volume.

On subsequent invocations of CONVERTV processing, DFSMSdss processes only those data sets not yet converted unless you specify the REDETERMINE keyword. If REDETERMINE is specified, DFSMSdss processes data sets already converted if their SMS management class or SMS storage class do not match those returned by the current ACS routines and data sets not yet converted. You may want to do this if your ACS routines changed since the last time you ran the CONVERTV operation on the volume.

SMS Report

Figure 3 shows a sample report generated by DFSMSdss during CONVERTV SMS processing.

```
PAGE 0001     5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES     1999.211 14:55
  CONVERTV         -
     SMS           -
     DYNAM(D9S060)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'CONVERTV'
ADR109I (R/I)-RI01 (01), 1999.211 14:55:22 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED.
ADR016I (001)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (001)-STEND(01), 1999.211 14:55:22 EXECUTION BEGINS
ADR860I (001)-KVSMS(01), PROCESSING BEGINS ON VOLUME D9S060
ADR873I (001)-KVSMS(01), VOLUME D9S060 IN STORAGE GROUP XFMT9SSG IS ELIGIBLE FOR CONVERSION TO SMS MANAGEMENT
ADR877I (001)-KVSMS(01), THE FOLLOWING DATA SETS ON VOLUME D9S060 WERE SUCCESSFULLY PROCESSED
                          PUBSEXMP.ESDS.S01                          CATALOG: TEST.CAT.PUBSEXMP
                              STORCLAS: XFMT9SSC                     MGMTCLAS: NONE
                          PUBSEXMP.KSDS.S01                          CATALOG: TEST.CAT.PUBSEXMP
                              STORCLAS: XFMT9SSC                     MGMTCLAS: NONE
                          TEST.CAT.PUBSEXMP                          CATALOG: TEST.CAT.PUBSEXMP
                              STORCLAS: XFMT9SSC                     MGMTCLAS: NONE
                          PUBSEXMP.SAM.S01                           CATALOG: TEST.CAT.PUBSEXMP
                              STORCLAS: XFMT9SSC                     MGMTCLAS: NONE
                          PUBSEXMP.PDS.S01                           CATALOG: TEST.CAT.PUBSEXMP
                              STORCLAS: XFMT9SSC                     MGMTCLAS: NONE
ADR885I (001)-KVSMS(01), VOLUME D9S060 HAS BEEN SUCCESSFULLY CONVERTED TO SMS MANAGEMENT

PAGE 0002     5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES     1999.211 14:55
ADR892I (001)-KVRPT(01), THE STATUS OF EACH VOLUME IS AS FOLLOWS
                                VOLUME            FINAL STATUS         REASON FOR FAILURE
                          ------------------      ------------       ----------------------
                          D9S060 - CONVERTED      SMS
ADR006I (001)-STEND(02), 1999.211 14:55:23 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 1999.211 14:55:23 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 1999.211 14:55:23 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS 0000
```

Figure 3. SMS Report

Special Data Set Requirements for Conversion to SMS

Some data sets have special requirements for conversion to SMS management. The sections below describe the special considerations for converting these data sets to SMS management.

VSAM Sphere Eligibility

A VSAM sphere is considered to be a single data set by the CONVERTV command. As a result, either all the data sets of the sphere are converted or none of them are.

If any of the following parts are ineligible for conversion, then all the clusters that compose the sphere are ineligible for conversion:

- Components of a base cluster
- Alternate indexes related to the base cluster
- Alternate index components
- Paths relating alternate indexes to the base cluster

You must direct all parts of a VSAM sphere (the base cluster, base cluster components, alternate indexes, alternate index components, and paths) to the same catalog by using an alias. If they are not directed to the same catalog, the sphere cannot be converted to SMS management. To correct this problem you can either rename the data sets in the sphere, or add or delete catalog aliases, and rerun the CONVERTV command.

Multivolume Data Sets

If you do not specify SELECTMULTI, all volumes must be included in DDNAME or DYNAM volume lists.

If you specify input volumes (with either the DDNAME or DYNAM volume list), a data set is selected based on the following criteria:

- When you either specify SELECTMULTI(ALL) or specify input volumes without specifying the SELECTMULTI keyword, **all** of the volumes that contain a part of the non-VSAM data set or VSAM base cluster must be in the volume list.
- When you specify SELECTMULTI(ANY), **any part** of the non-VSAM data set or VSAM base cluster can be on a volume in the volume list.
- When you specify SELECTMULTI(FIRST), the volume list must include the volume that contains either the **first part** of the non-VSAM data set or the primary data component of the base cluster for a VSAM sphere.

Multivolume data sets are not eligible for conversion if any part of the data set resides on volumes that:

- Do not have indexed VTOCs
- Are not defined in an SMS storage group
- Are defined to a different storage group
- Are not permanently mounted and online

If the previous requirements are satisfied, DFSMSdss verifies that all the volumes on which the data set resides:

- Are permanently mounted and online
- Have indexed VTOCs
- Are defined to the same storage group

If all these criteria are met, the data set is converted to SMS management.

Notes:

1. If SELECTMULTI(FIRST) or SELECTMULTI(ANY) is specified, volumes not specified in the DDNAME or DYNAM volume lists are put in the INITIAL state following a successful conversion of the data set to SMS (unless the volume is already in the INITIAL or SMS state.)
2. If SELECTMULTI is not specified or if SELECTMULTI(ALL) is specified, volumes not specified in the DDNAME or DYNAM volume lists are *not* put in the INITIAL state.

DFSMSdss cannot determine whether or not a volume being converted is a candidate volume for one or more data sets in the system. If such a volume is converted, DFSMSdss cannot ensure consistent conversion for all of the volumes of the data set (or sets) for which the volume is a candidate. This can result in a data set having both SMS-managed and non-SMS-managed volumes in its volume list, which can cause the data set to become unusable.

To avoid this situation when performing CONVERTV operations, if you specify any volume of a multivolume data set in the list of volumes to be converted, ensure that you also include at least one of the primary volumes of the data set. This allows DFSMSdss to ensure that all of the volumes of the data set are converted consistently.

GDG Data Sets

Generation data groups (GDGs) require special consideration while being cataloged or uncataloged during SMS conversion. Uncataloged GDGs are converted to SMS

management, but are left uncataloged. Messages ADR877I and ADR879I indicate NOT CATALOGED for the catalog name in the data set name lists for SMS processing.

Temporary Data Sets

Data set VTOC entries of temporary data sets are updated to indicate uncataloged SMS status.

VTOC and VVDS

Data set VTOC entries for the VTOC, VTOC index, and VVDS are updated to SMS management.

Converting from SMS Management without Data Movement

If you want to take volumes out of SMS management, you can use the CONVERTV command with the NONSMS keyword. All volumes and most data sets (see note 2 below) are eligible for NONSMS processing. After you execute this command, the volume indicators that designate the volume as an SMS-managed volume are turned off. The active SMS configuration should be updated to remove the volume from its storage group, otherwise data set allocations to the volume will fail. Thereafter, only non-SMS-managed data sets can be allocated to the volume.

As with the SMS keyword, you can specify the TEST keyword with NONSMS. No conversion is actually done, but a report is generated that identifies the data sets that are and are not eligible for conversion from SMS management. The report also indicates whether the volume as a whole is eligible for conversion from SMS management.

To convert a data set from SMS management, the data set's classes are deleted from its catalog entry. Nonspecific volumes also are deleted from the catalog entry. For a VSAM data set, the SMS-related items are deleted from the catalog entry. For a non-VSAM data set, the catalog entries are updated to remove the SMS information. After the catalog and VVDS updates and deletions are made, the VTOC entry is updated to be non-SMS-managed.

Note: You cannot specify the CATALOG and REDETERMINE keywords with NONSMS.

Special Data Set Requirements for Conversion from SMS

When being converted from SMS management, some data sets require special consideration. The following sections discuss some of the special requirements for converting data sets from SMS management.

Multivolume Data Sets

All pieces of a multivolume data set must be converted from SMS management at the same time. You can do this by using the SELECTMULTI keyword.

If you do not specify SELECTMULTI, then you must specify all the volumes in the DDNAME or DYNAM volume list on which the data set resides.

If you specify input volumes (with either the DDNAME or DYNAM volume list) for

NONSMS processing, a data set is selected based on the following criteria:

- When you either specify SELECTMULTI(ALL) or specify input volumes without specifying the SELECTMULTI keyword, **all** of the volumes that contain a part of the non-VSAM data set or VSAM base cluster must be in the volume list.
- When you specify SELECTMULTI(ANY), **any part** of the non-VSAM data set or VSAM base cluster can be on a volume in the volume list.
- When you specify SELECTMULTI(FIRST), the volume list must include the volume that contains the **first part** of either the non-VSAM data set or the primary data component of the base cluster for a VSAM sphere.

Those volumes not included in the volume list will be placed in the INITIAL state. Being in the INITIAL state locks all allocations to the volume until all data sets residing on it are converted.

DFSMSdss cannot determine whether or not a volume being converted is a candidate volume for one or more data sets in the system. If such a volume is converted, DFSMSdss cannot ensure that all of the volumes of the data set (or sets) for which the volume is a candidate, are converted consistently. This can result in a data set having both SMS-managed and non-SMS-managed volumes in its volume list, which can cause the data set to become unusable.

To avoid this situation when performing CONVERTV operations, if you specify any volume of a multivolume data set in the list of volumes to be converted, ensure that you also include at least one of the primary volumes of the data set. This allows DFSMSdss to ensure that all of the volumes of the data set are converted consistently.

GDG Data Sets

When you convert from SMS management, generation data group (GDG) data sets require special consideration with regard to cataloging. Data sets marked as “deferred roll in and rolled out” are uncataloged.

Temporary Data Sets

Data set VTOC entries for temporary data sets are updated to non-SMS status.

VTOC and VVDS

Data set VTOC entries for the VTOC, VTOC index, and VVDS are updated to non-SMS status.

Special Considerations for Using Non-SMS-Managed Targets

When moving to non-SMS-managed targets, there are some special considerations for certain data sets:

- Extended-format data sets cannot be moved to a non-SMS-managed target.
- COPY with DELETE and without RENAMEU is not supported for data sets with DFM attributes. DFM attributes are not maintained for non-SMS data sets.

Chapter 12. Managing Space with DFSMSdss

You can use DFSMSdss to help manage your DASD space. This chapter is organized as follows:

- **"Reclaiming DASD Space"** discusses how to use DFSMSdss to reclaim DASD space.
- **"Consolidating Free Space on Volumes" on page 141** discusses how to use the DEFRAG command to reduce fragmentation on volumes.

Reclaiming DASD Space

You can reclaim DASD space with DFSMSdss in the following ways:

- Releasing unused space in data sets
- Compressing partitioned data sets to consolidate unused space at the end of the data sets and then releasing the unused space
- Deleting unwanted data sets
- Combining data set extents

Releasing Unused Space in Data Sets

The RELEASE command releases allocated but unused space from all sequential, partitioned, and extended-format data sets that you select with INCLUDE, EXCLUDE, or BY criteria. For an explanation of these criteria, see "Choosing Data Sets for Processing—Filtering" on page 18. DFSMSdss selects only data sets that have space that can be released. You can also use ISMF to build a list of data sets based on the amount of unused space and to invoke DFSMSdss to release the unused space in them.

Exclude data sets whose last block pointer in the data set VTOC entry is not maintained in the VTOC by using the EXCLUDE keyword. This can occur if you use an access method other than BSAM, QSAM, BPAM, or VSAM. DFSMSdss does not release space for data sets whose last block pointer in the data set entry is 0.

The following options can help you use the release function more effectively:

MINSECQTY(n)
Allows you to specify that space not be released unless the user's secondary allocation is greater than or equal to *n*. In this way, you ensure that the user can still add to the data set after the release. The default value for *n* is 1.

MINTRACKSUNUSED(n)
Allows you to specify that space not be released unless the number of unused tracks is greater than or equal to *n*. Without MINTRACKSUNUSED, space is released if the data set has one or more unused tracks.

Note: When space in a data set is released, all unused space is released, not just the amount beyond the minimum unused (as specified by MINTRACKSUNUSED).

To protect the user, DFSMSdss does not release any space in a data set if:

- The data set has the maximum number of used extents. A data set with the maximum number of allocated extents but fewer than the maximum number of used extents will have the unused space released.
- The cylinder-allocated data set has unused tracks but not an entire unused cylinder.
- The data set's name begins with SYS1, unless the PROCESS(SYS1) keyword is specified. To limit the use of PROCESS, you need to set up a RACF facility class profile.

Compressing a PDS

The COMPRESS command compresses a PDS on a specified volume. Compression removes unused space between members in a partitioned data set. This recovered space is then available for reuse at the end of the data set. Depending on the filtering criteria you specify, you can compress all the partitioned data sets or only some of the data sets. This command is useful for compressing system partitioned data sets before applying maintenance (thus avoiding certain space-related abends). You must not compress the data sets that contain DFSMSdss or IEBCOPY executable code.

The actual PDS compression is done in place. To prevent loss of data if the system or the compression operation abnormally ends during processing, back up your volume or data sets that meet the filtering criteria before using this command.

COMPRESS does not support processing partitioned data sets that:

- Are unmovable
- Have no directory

Deleting Unwanted Data Sets

You can use the DELETE and PURGE keywords and data set filtering with a *physical* data set dump to delete unwanted data sets from DASD.

Note: This does not apply to VSAM data sets, multivolume non-VSAM data sets, or migrated data sets.

On a logical data set dump when using the DELETE keyword, VSAM, non-VSAM, and multivolume data sets are deleted. DFSMSdss cannot be used to delete migrated data sets.

The following steps show how to delete (scratch and uncatalog) all data sets that have expired and all data sets that have not been referred to in the last year. The data sets are not actually moved to a dump volume.

1. JCL requirement:

```
//NOTAPE DD DUMMY
```

   The above JCL prevents moving any data sets.

2. Issue the following control statements to delete (scratch and uncatalog) all data sets not referred to in the last year:

```
DUMP INDD(VOL111) OUTDD(NOTAPE) -
   DATASET(BY(REFDT,LE,*,-366)) -
   DELETE PURGE
```

3. Issue the following control statements to delete all expired data sets:

```
DUMP INDD(VOL111) OUTDD(NOTAPE) -
   DATASET(INCLUDE(**) -
   BY(EXPDT,LT,*)) -
   DELETE
```

   Note: You can modify the above example to apply to VSAM and multivolume data sets by omitting the INDD statement or by specifying LOGINDD. This JCL results in a logical data set dump operation.

4. Issue the following control statements to delete uncataloged non-VSAM data sets:

   For a physical data set dump:

```
DUMP DATASET(INCLUDE(**) -
   BY((DSORG NE VSAM) -
   (CATLG EQ NO))) -
   INDDNAME(DASD1,DASD2) -
   OUTDDNAME(TAPE) -
   DELETE PURGE
```

   Note: The DD named TAPE can be a DD dummy if a dump of the uncataloged data sets is not wanted. DASD1 and DASD2 identify the input volumes. Because a physical data set dump processes each volume in order one at a time, it can handle multiple, uncataloged, single-volume data sets with the same name when multiple input volumes are specified. It cannot handle a multivolume data set even if all the volumes on which it resides are specified as input volumes.

   For a logical data set dump:

```
DUMP DATASET(INCLUDE(**) -
   BY((DSORG NE VSAM) -
   (CATLG EQ NO))) -
   LOGINDDNAME(DASD1,DASD2) -
   OUTDDNAME(TAPE) -
   DELETE PURGE
```

   Note: The DD named TAPE can be a DD dummy if a dump of the uncataloged data sets is not wanted. DASD1 and DASD2 identify the input volumes. A logical data set dump cannot handle multiple, uncataloged data sets with the same name in the same job even when all the volumes on which they reside are specified as input volumes.

   A logical dump can handle a legitimate multivolume uncataloged data set if all the volumes on which it resides are specified as input volumes and if there is no cataloged data set by the same name on the system.

Combining Data Set Extents

The DUMP command used with the DELETE and PURGE keywords scratches and uncatalogs the data sets from DASD after they are dumped. If you restore those data sets to the same DASD, allocation attempts to get the space for the entire data set. If the DASD volume has sufficient contiguous unused space, the allocated space will most likely be in one contiguous extent. Because unmovable data sets are not deleted, however, the volume might be fragmented, preventing a complete restore for all data sets.

If you copy those data sets to the same DASD, allocation attempts to get the space for the entire data set. If the DASD volume has sufficient contiguous unused space, the allocated space will most likely be in one contiguous extent. If you do not specify ALLDATA and ALLEXCP for sequential and partitioned data sets, only used spaces are allocated.

Note: Do not use this technique for unmovable data sets such as ABSTR allocated or indexed sequential data sets.

The following steps show how to dump and delete (scratch and uncatalog) all movable non-VSAM data sets, defragment volumes, and restore all movable non-VSAM data sets.

1. Issue the following control statements to dump and delete all movable, single-volume, non-VSAM data sets:

```
DUMP INDD(DASD1) OUTDD(TAPE1) OPTIMIZE(3) -
  DATASET(BY((DSORG,NE,VSAM),(ALLOC,EQ,MOV),(MULTI,EQ,NO))) -
  DELETE PURGE
```

2. Issue the following control statement to defragment the volume:

```
DEFRAG DDN(DASD1)
```

3. Issue the following control statements to restore and catalog all dumped data sets:

```
RESTORE INDD(TAPE1)  OUTDD(DASD1) -
   DATASET(INCLUDE(**)) -
   CATALOG
```

You can specify the CONSOLIDATE keyword when you defragment the volume as an alternative to the above three-step approach (dump and delete, defragment, and restore).

Issue the following control statement to defragment the volume, and perform extent reduction if possible:

```
DEFRAG DDN(DASD1) CONSOLIDATE
```

Consolidating Free Space on Volumes

Because of the nature of allocation algorithms and the frequent creation, extension, and deletion of data sets, free space on DASD volumes becomes fragmented. This results in:

- Inefficient use of DASD storage space
- An increase in space-related abends (abnormal endings)
- Performance degradation caused by excessive DASD arm movement
- An increase in the time required for functions that are related to direct access device space management (DADSM)

By using the DEFRAG command, you can consolidate the free space on volumes and avoid this problem. The DEFRAG command relocates data set extents on a DASD volume to reduce or eliminate freespace fragmentation, and prints a report about free space and other volume statistics. Also, you can specify which data sets, if any, are to be excluded from data-set-extent relocation. Data set extents are not combined as a result of DEFRAG processing unless you also specify the optional CONSOLIDATE keyword.

When you specify the CONSOLIDATE keyword, the DEFRAG command attempts to consolidate data set extents and perform extent reduction for data sets that occupy multiple extents. When you process a volume with the CONSOLIDATE keyword, DFSMSdss searches each moveable data set. A data set that has multiple extents and is not excluded from data movement is eligible for extent consolidation and extent reduction. For eligible data sets that consist of contiguous extents that are in sequential order, DFSMSdss consolidates without extent relocation. Otherwise, eligible data sets are relocated if enough contiguous free space exists on the volume to hold the resulting data set. When DFSMSdss has completed consolidation for all eligible data sets, DEFRAG processing consolidates the remaining free space extents using existing DEFRAG algorithms.

Notes:

1. The process of combining data set extents can cause the freespace to be more fragmented than it was before the operation began.
2. Despite the fact that DFSMSdss performs freespace defragmentation following the consolidation of data set extents, there is a possibility that the fragmentation index may be higher following a defrag operation with CONSOLIDATE specified than before the operation began.

When to Run the DEFRAG Function

You can run the DEFRAG function on a volume at any time. However, running DEFRAG processing locks the VTOC (through the RESERVE macro) and the VVDS, if it exists on the volume. The DEFRAG function also serializes on data sets through ENQ or dynamic allocation. These activities might cause excessive wait time for other jobs to update the VTOC. Therefore, times of low system activity are best for DEFRAG runs.

DFSMSdss erases the source location for every extent that is moved during the DEFRAG operation when you specify the ADMINISTRATOR keyword. This occurs even if the extent is not part of an erase-on-scratch data set.

DFSMSdss can use FlashCopy during a DEFRAG operation if the device is in an ESS that supports data set FlashCopy (FlashCopy Version 2). FlashCopy is much faster than traditional data movement methods, especially when you are moving large amounts of data.

DFSMSdss can also use SnapShot to quickly move the data from the source location to the target location during a defrag operation if the device is in a RAMAC Virtual Array. SnapShot is much faster than traditional methods of data movement, especially when moving large amounts of data.

Designating FlashCopy Usage

The FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword tells DFSMSdss how you want to use fast replication methods such as FlashCopy. The default is FASTREPLICATION(PREFERRED).

FASTREPLICATION(REQUIRED) specifies that DFSMSdss must use data set FlashCopy for the DEFRAG operation. If FlashCopy cannot be used for one of the following *normal* reasons, DFSMSdss issues informational message ADR946I and error message ADR938E, which indicates that the processing of the current extent failed. DEFRAG processing attempts to use FlashCopy to move the subsequent extents.

- The target tracks are already the source of a FlashCopy operation.
- The source tracks are already the target of a FlashCopy operation.
- The target tracks will exceed 12 relationships, which is the maximum relationships that are allowed for any source tracks.

If FlashCopy cannot be used for reasons other than the *normal* reasons, DFSMSdss issues message ADR945W and error message ADR938E, which indicates that the processing of the current extent failed. DFSMSdss terminates DEFRAG processing.

FASTREPLICATION(PREFERRED) specifies that DFSMSdss attempt to use data set FlashCopy before any other I/O method. If data set FlashCopy cannot be used for one of the normal reasons listed earlier, DFSMSdss issues message ADR946I and uses traditional I/O methods to move the current extent. DEFRAG processing attempts to use FlashCopy to move the subsequent extents. If FlashCopy cannot be used for reasons other than the *normal* reasons, DFSMSdss issues message ADR945W and uses traditional I/O methods to move the current extent and all the subsequent extents on the volume.

& **NOTE:** The unexpected FlashCopy failures are logged in the LOGREC by services
& that DFSMSdss initiates to perform a FlashCopy operation. The *normal* FlashCopy
& reasons are not logged in the LOGREC.

FASTREPLICATION(NONE) specifies that DFSMSdss not attempt to use data set FlashCopy during the DEFRAG operation.

Related reading: For additional information about the FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

Determining Why FlashCopy Cannot be Used

There may be times when you expect DFSMSdss to use FlashCopy to move the data but FlashCopy was not used. As far as you can tell, your volume meets all the criteria for data set FlashCopy use. Use the DEBUG(FRMSG(MINIMAL | SUMMARIZED | DETAILED)) keyword to help you resolve this situation. Include this keyword indicating the applicable fast replication message level (MIN, SUM, or DTL) in your DEFRAG command. The message level controls the type and amount of information that DFSMSdss provides.

DEBUG(FRMSG(MIN | SUM | DTL)) directs DFSMSdss to issue an informational message that indicates why data set FlashCopy was not used. When you specify FASTREPLICATION(REQUIRED), the informational message is issued in addition to the ADR938E message whether you have specified the DEBUG(FRMSG(MIN | SUM | DTL)) keyword or not.

Related reading: For additional information about the DEBUG(FRMSG(MIN | SUM | DTL)) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

Designating SnapShot Usage

The FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword tells DFSMSdss how you want to use SnapShot. The default is FASTREPLICATION(PREFERRED).

FASTREPLICATION(REQUIRED) specifies that DFSMSdss must use SnapShot for the DEFRAG operation. If SnapShot cannot be used to move an extent DFSMSdss issues error message ADR938E, which indicates that the DEFRAG operation failed.

FASTREPLICATION(PREFERRED) specifies that DFSMSdss attempt to use SnapShot before any other I/O method. If SnapShot cannot be used, DFSMSdss uses traditional I/O methods to move the current extent and all the subsequent extents on the volume.

FASTREPLICATION(NONE) specifies that DFSMSdss not attempt to use SnapShot during the DEFRAG operation.

Related reading: For additional information about the FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

Determining Why SnapShot Cannot be Used

There may be times when you expect DFSMSdss to use SnapShot to move the data but SnapShot was not used. As far as you can tell, your volume meet all the criteria for SnapShot use. Use the DEBUG(FRMSG(MINIMAL | SUMMARIZED | DETAILED)) keyword to help you resolve this situation. Include this keyword to indicate the applicable fast replication message level (MIN, SUM, or DTL) in your DEFRAG command. The message level controls the type and amount of information that DFSMSdss provides.

DEBUG(FRMSG(MIN | SUM | DTL)) directs DFSMSdss to issue an informational message that indicates why SnapShot was not used. When you specify FASTREPLICATION(REQUIRED), the informational message is issued in addition to the ADR938E message whether you have specified the DEBUG(FRMSG(MIN | SUM | DTL)) keyword or not.

Related reading: For additional information about the DEBUG(FRMSG(MIN | SUM | DTL)) keyword, see the *z/OS DFSMSdss Storage Administration Reference*.

Data Sets Excluded from DEFRAG Processing

DFSMSdss automatically excludes and does not relocate the following types of data sets in a DEFRAG operation:

- User-specified data sets (EXCLUDE)
- Data sets that do not satisfy all BY criteria
- Indexed sequential data sets
- VSAM data sets not cataloged in an integrated catalog facility catalog
- Key range VSAM data sets
- Catalogs (system and user)
- The VTOC index data set
- RACF control data sets (any data set with a name in the form SYS1.RACF*.**)
- Page, swap, and SYS1.STGINDEX data sets
- VSAM volume data sets (VVDS)
- Unmovable data sets
- Data sets allocated by absolute track
- Data sets that it cannot serialize for exclusive access
- VSAM data sets that have Record Level Sharing (RLS) information associated with them (only the first extent of this type of data set is excluded from the DEFRAG operation)

Because the DEFRAG function does not relocate these data sets, the effectiveness of a DEFRAG run is affected by their presence.

Place the following data sets in the EXCLUDE list if they are present on the volume being defragmented:

1. If you plan to defragment a volume containing the active RACF database, you must place the RACF database data sets in the EXCLUDE list.
2. Any data that has been defined as a retained DLF object for use with Hiperbatch™.

Note: Exclude system data sets that are opened and are being accessed without an enqueue.

DEFRAG Options

You can use the following keywords to make more efficient use of the DEFRAG command:

CONSolidate Perform extent reduction by consolidating multiple extent data sets when possible.

DYNALLOC The use of dynamic allocation to serialize the use of data sets, rather than enqueue, does not always provide cross-system serialization.

FRAGI(n) Perform a DEFRAG operation only if the fragmentation index is more than *n*.

MAXMOVE(n,p)
Stop the DEFRAG run when *n* contiguous free tracks are assembled. If *n* contiguous free tracks already exist, the DEFRAG function tries to further reduce the fragmentation of the volume but no more than *n* tracks are relocated. If more than *n* tracks must be relocated, no DEFRAG is performed.

n The number of free tracks that DFSMSdss is to try to assemble in a contiguous area.

p The number of passes DFSMSdss is to make in attempting to assemble the tracks.

PASSDelay Specify the time delay between the passes (p) specified in MAXMOVE(n,p) to allow access to the volume between passes.

WAIT(s,r) If the data set is unavailable, wait *s* seconds before retrying to obtain control of it and retry only *r* times.

To determine the fragmentation index of a volume without actually performing the DEFRAG operation, code the NORUN parameter on the EXEC statement in your JCL. In addition to the fragmentation index, the NORUN parameter lists the number of free cylinders, the number of free tracks, the number of free extents, the largest free extent size, and the percentage of free space on the volume. A map of the volume with the CCHH location of each data set or free space (in ascending order from cylinder 0, track 0) is also issued.

General Hints

- If you want the DEFRAG function to perform in the shortest period of time or to create the largest single freespace extent, perform only the first pass. Do so by coding the MAXMOVE(*n*) parameter, using a very high value (9999) for *n*. When the value is higher than the DEFRAG function can assemble, the process stops at the end of the first pass. For example:

  DEFRAG DYNAM(388002) MAXMOVE(9999)
- Experimenting with the DEFRAG FRAGI and MAXMOVE parameters will allow you to compare results when you operate on DASD with different fragmentation characteristics. The fragmentation index that can be specified by the DEFRAG options represents a number between 0 and 1 and can be one to three digits long. (FRAGI(333) represents 0.333 and FRAGI(3) represents 0.3. The recommended basic DEFRAG parameters are to let MAXMOVE default and use FRAGI(3). To defragment DASD volume 388001, the following could be used:

```
DEFRAG DYNAM(388001) FRAGI(3)
```

Serialization

The DEFRAG command serializes access to the VTOC. The DEFRAG command releases this serialization before it generates the ending statistics provided by message ADR213I. Therefore, the information in message ADR213I may not reflect the state of the volume at the completion of DFSMSdss processing because another job may allocate or delete data sets on the processed volume between the time the serialization is released and the ending statistics are obtained. The serialization scheme is described in *z/OS DFSMSdss Storage Administration Reference*.

The DEFRAG command does a RESERVE on the VTOC to serialize access to the VTOC. The DEFRAG command also serializes access to each data set before relocating the extent of a data set. The enqueue scheme used by the DEFRAG function ensures integrity on a single processor but does not ensure integrity for data sets on DASD shared between processors. This is due to the use of an ENQ scope of SYSTEM for the SYSDSN resource name. To ensure the integrity of data

sets on a shared DASD, you must do one of the following:

- Vary the volume offline from all processors except the one on which DEFRAG runs. After the DEFRAG function finishes, you can vary the volume back online for the other processors.
- In either a JES2 or a JES3 environment, you can use multisystem GRS (or equivalent function) to convert the scope of all enqueues with a resource name of SYSDSN from SYSTEM to SYSTEMS by placing SYSDSN in the GRS SYSTEM INCLusion resource name list (RNL). This allows all systems in the GRS ring to be made aware of all SYSDSN enqueues. The default GRS System Inclusion RNL includes SYSDSN. However, ensure that this has not been changed on your system before using the DEFRAG command on a volume shared between two or more processors.

  Note: GRS must not be used to convert the scope of any of DEFRAG function's SYSTEMS enqueues (including SYSVSAM) to SYSTEM by placing the resource names in the GRS RNL. GRS may, however, be used to convert DEFRAG function's RESERVE on SYSVTOC to a simple enqueue with a scope of SYSTEMS by including it in the GRS "RESERVE CONVERSION RNL". If you choose not to do this, you can avoid doing two global serializations on the volume's VTOC by placing SYSVTOC in the GRS Systems Exclusion RNL, thus changing RESERVE's global enqueue to a local enqueue. Refer to *z/OS MVS Planning: Global Resource Serialization*, or *OS/VS2 MVS Planning: Global Resource Serialization* to find out which restrictions apply to enqueues and dequeues. The DYNALLOC serialization mechanism of DFSMSdss does not solve all cross-system serialization problems. GRS (Global Resource Serialization) is recommended with shared DASD.
- If you are running on a system using JES2 and are not using multisystem GRS (or equivalent function), you can use DEFRAG function's BY filtering to specifically include or exclude data sets from processing. Both creation date and last-referenced date criteria are needed to ensure that only those data sets that are not in use are selected for DEFRAG function processing. For example, if you choose to defragment a volume with typical TSO or batch data sets, you could select only those data sets that were created and referenced more than two days previously. Two days should be a minimum selection age because of the level of precision of the creation date. In the following example, data set A.B.C is created two minutes before the DEFRAG function begins.

```
TIME OF DAY     ACTION
    2359          Create data set A.B.C
     .
     .
    0001          Begin DEFRAG BY(CREDT,LT,-1).  Data set A.B.C is selected
                  because it has now been one day since creation
```

However, because there is a date change between the two actions, A.B.C was selected for the DEFRAG operation. A two day delay enforces a convention that more than 24 hours must pass before a data set is eligible for DEFRAG. In an environment with TSO and batch data sets, the probability that one of these data sets will be open more than 24 hours is low. In the following example, choosing data sets that have not been manipulated within the preceding 24 hours causes DEFRAG processing only for those data sets on volume SHARE3 that were created and referenced more than two days previously and that are not temporary data sets.

```
DEFRAG BY(LIST((CREDT LT *,-2),(REFDT LT *,-2))) -
       EXCLUDE(LIST(SYS8*.T*.**)) DYNAM(SHARE3)
```

The two-day criterion is probably sufficient for TSO and batch type data sets. However, if there are applications that use the volumes being defragmented, consider setting the delay time to the maximum time that the application would have a data set open.

- If you are running on a system using JES3 with MDS enabled and are not using multisystem GRS (or equivalent function), you can use the DEFRAG command DYNALLOC keyword to provide serialization for data sets on shared DASD.

  Note: Not all data sets allocated within a JES3 environment are known to the global. The use of the DYNALLOC keyword does not provide cross-system serialization for these data sets.

  – Allocation of existing (old) data sets whose names appear in the RESDSN and DYNALDSN lists are not protected by the DYNALLOC serialization mechanism of DFSMSdss. DEFRAG processing for these data sets can be prevented by placing their names (or filters for the names) in the EXCLUDE LIST for the DEFRAG command.

  – New data sets created with nonspecific allocation (no volume serial supplied) are not protected by the DYNALLOC serialization mechanism of DFSMSdss. However, you can use BY filtering with the DEFRAG command to specifically include or exclude data sets from processing. In the following example, the DEFRAG function processes only those data sets on volume SHARE3 that were created more than two days before.

```
DEFRAG BY(LIST(CREDT,LT,*,-2)) DYNALLOC DYNAM(SHARE3)
```

You can also use the EXCLUDE parameter to avoid processing data sets that were created by long-running programs or subsystems more than two days previously but that are still allocated. In the following example, if the newly created data sets are temporary, the DEFRAG operation processes only those data sets on volume SHARE3 that were created more than two days before and are not temporary data sets.

```
DEFRAG BY(LIST(CREDT,LT,*,-2)) -
       EXCLUDE(LIST(SYS8*.T*.**)) DYNALLOC DYNAM(SHARE3)
```

You can ensure successful DEFRAG processing of volumes having a significant number of free or allocated extents by specifying appropriate SIZE and REGION parameters in the EXEC statement. If you receive a message that the region size is not large enough, specify a larger region size in the EXEC or JOB statement and rerun your job.

Note: During DEFRAG processing, a data set VTOC entry with the unique name “SYS1.DFDSS.DEFRAG.xxxxxxxx.volser.DUMMY” is allocated on the volume being defragmented. This data set is not cataloged but is automatically deleted after a successful run. If a job is canceled or abnormally ends, this data set remains on the volume. After the restart, DADSM functions might fail with message IEC602. To correct this problem

or to delete the “SYS1.DFDSS.DEFRAG.xxxxxxxx.volser.DUMMY” entry, rerun the DEFRAG function on the volume.

Security Considerations

For security purposes, the data set tracks used before the relocation are erased after relocation under these conditions:

- When the z/OS Security Server (RACF element) Version 1 Release 7 is installed and either:
  - The data set was defined to RACF with the RACF ERASE option
  - The VSAM data set has the ERASE attribute
  - The data set is password protected. (In this case, if the data set is also defined to RACF, the RACF ERASE option is taken. Table 9 provides more detail.)

Table 9. Data Set Erase Table for DEFRAG with z/OS Security Server (RACF element) Version 1 Release 7

	Password Protected	**Defined ERASE**	**RACF Protected**	**Erased on Scratch**
User Install Exit=ERASE (default)	No	No	No	No
	Yes	No	No	Yes
		Yes	No	Yes
	No	No	Yes (=ERASE)	Yes
	No	No	Yes (=NOERASE)	No
	No	Yes	Yes (=ERASE)	Yes
	No	Yes	Yes (=NOERASE)	Yes
	Yes	Yes	Yes (=ERASE)	Yes
	Yes	Yes	Yes (=NOERASE)	Yes
User Install Exit=NOERASE	No	No	No	No
Note: The catalog entry contains the ERASE attribute specified when the data set was defined (VSAM only).				

The data set tracks that were used before the relocation are also erased after relocation if you have specified the ADMINISTRATOR keyword. This occurs whether the tracks are part of the erase-on-scratch data set or not.

You can prevent the tracks from being erased by using the installation options exit routine.

The DEFRAG function does not relocate protected data sets unless:

- You have RACF DASDVOL update access to the volume.
- You have RACF DATASET read access to the data sets on the volume.
- You specify the read or update password for password-protected data sets, or the Installation Authorization Exit Routine supplied with DFSMSdss is changed to allow relocation of protected data sets.

When RACF DASDVOL class is active and a profile exists for the volume, a DASDVOL authorization failure causes the DEFRAG task to abend with a system code 913. This happens regardless of RACF data set access authority.

Related reading: For additional information about the installation options exit routine, see *z/OS DFSMS Installation Exits*.

Maximizing Track Utilization by Reblocking Data Sets

DFSMSdss provides a REBLOCK keyword that allows users to maximize the track usage by data sets during copy and restore processing. When REBLOCK is specified on a fully or partially qualified name of a sequential or partitioned data set during copy or restore processing, DFSMSdss will choose an optimal block size for the data set and the device. However, the installation reblock exit can specify that a different block size be used (except for partitioned load modules during copy operations).

REBLOCK is ignored for:
- Unmovable data sets
- Data sets with record format of U (except for partitioned load modules during copy operations), V, VB, VBS, or F
- Partitioned data sets with note lists (except for partitioned load modules during copy operations)
- Partitioned data sets that are also specified in the NOPACKING keyword

Partitioned load modules can be reblocked in copy operations, even if they have NOTELISTs.

The reblockable indicator in the data set's VTOC entry also determines whether a data set is to be reblocked or not. When the indicator is on, the data set is always reblocked to a system determined, optimal block size, except when the data set:
- Is a partitioned data set that is also specified in the NOPACKING keyword
- Is an unmovable data set
- Has a record format of V, VS, VBS, or F

The installation reblock exit is not called if the reblockable indicator is ON.

A PDSE being converted to a PDS will always be reblocked except when the data set has a record format of V, VS, VBS, or F; or when the data set's record length is '0'.

Related reading: For additional information about the installation reblock exit routine, see the *z/OS DFSMSdss Storage Administration Reference*.

Chapter 13. Introduction to Diagnostics

You might one day experience a problem with the DFSMSdss program operation that will cause you to contact IBM Service. They will ask you to describe the problem to them so that they might help solve it. This document explains how you can diagnose the problem by performing the following actions:

- Systematically develop a set of *keywords* that describe a DFSMSdss program failure.
- Describe the program failure to the IBM representative by using the keywords.

A keyword is an agreed-upon word or abbreviation that is used to describe a single aspect of a program failure.

Use the following procedure as a guide to help you find a resolution to the program failure:

1. Select your set of keywords.
2. Use the keywords to search the ServiceLink function within IBMLink™.
3. Determine whether an authorized program analysis report (APAR) has been previously recorded for the failure. A description of the problem and usually a solution (designated by an APAR number) is provided when there is a match to your set of keywords.

   An APAR is a record of a product operation discrepancy. APAR records are maintained in the IBM Software Support Facility (SSF) database.
4. Use the keywords to describe the program failure when you contact IBM for assistance.

Using Keywords

Searching SSF or the early warning system (EWS) with only the first keyword (the DFSMSdss component identifier) will detect all reported problems for the entire program product. Each keyword that you add to the search argument, however, makes the search more specific, reducing the number of problem descriptions to be considered. In some cases, a search can locate a correction for a problem with less than a full set of keywords. If for some reason it is difficult to determine a particular keyword, you can omit that keyword.

Determining the Source of the Failure: DFSMSdfp, DFSMSdss, or DFSMShsm

The interactive storage management facility (ISMF) component of DFSMSdfp provides an interactive interface to DFSMSdss and DFSMShsm. Try to determine where the error occurred. Examine the content of the error message and the error logs. It is likely that the error has occurred either in DFSMSdss or in the interface to ISMF if you were performing one of the following functions:

BUILDSA
& CGCREATE
COMPRESS
CONVERTV
COPY
COPYDUMP
DEFRAG
DUMP
PRINT
RELEASE
RESTORE

If the failure occurred while you were using a function that does not appear in the list, it is possible the failure occurred in either DFSMSdfp or in the ISMF interface to DFSMShsm.

ISMF also uses the functions provided by the DFSMSdfp common services component, which consists of the following three routines:

- Common filter services
- DASD calculation services
- Device information services

Related reading:

- For information about diagnosing DFSMSdfp function failures, see the *z/OS DFSMSdfp Diagnosis*.
- For information about diagnosing DFSMShsm function failures, see the *z/OS DFSMShsm Diagnosis*.
- For information about beginning the diagnostic procedure for a DFSMSdss failure, see Chapter 14, “Using Keywords to Identify the Problem,” on page 153.
- For information about the DFSMSdss dump data set, see Appendix C, “Format of the DFSMSdss Dump Data Set,” on page 179.
- For information about the DFSMSdss patch area, see Appendix D, “DFSMSdss Patch Area,” on page 193.

Chapter 14. Using Keywords to Identify the Problem

This section explains the keywords and their relation to the full set of keywords used in describing a DFSMSdss program failure. Keywords are made up of the following categories:

- Component identification
- Release-level
- Type-of-failure
- Function
- Module
- Maintenance-level

Component Identification Keyword

The component identification keyword is the first keyword in a set. Use this keyword when you suspect that DFSMSdss is the failing component.

Note: If you are using the ISMF panels for DFSMSdss, exit this procedure and continue your diagnosis using the*z/OS DFSMSdfp Diagnosis*.

Example:

```
5695DF175
```

Procedure: The component identification number for DFSMSdss is **5695DF175**. Go to "Release-Level Keyword."

Release-Level Keyword

Using this keyword to identify the release level of DFSMSdss is optional in the SSF or EWS search argument. However, it is required in an APAR.

Example:

```
R1k0
```

Procedure: The keyword for z/OS V1R7 DFSMSdss is **R1k0**. Go to "Type-of-Failure and Function Keywords."

Type-of-Failure and Function Keywords

Select one keyword from Table 10 on page 154 that best describes the type of failure and go to the section indicated. Most type-of-failure keywords are accompanied by a function keyword. The function keywords are described in the type-of-failure sections. If you are not certain which of two keywords to use for the type of failure, use the one that appears first on your display screen.

Table 10. Summary of Type-of-Failure Keywords

Keyword	Type of Failure	See Page
ABEND*xxx*	DFSMSdss ends abnormally because of a system-detected error. **Note:** The ABEND*xxx* keyword does not apply to the stand-alone restore program of DFSMSdss.	154
MSGADR*nnnt*	An error is related to a DFSMSdss message.	156
WAIT	The program does not seem to be doing anything.	156
LOOP	The program is doing something repetitively.	157
INCORROUT	Output from the program is incorrect or missing.	158
DOC	Documentation of the program is in error.	159
PERFM	Program performance is degraded.	160

ABEND*xxx*

Use this procedure when DFSMSdss ends abnormally. However, if the abend is user abend "0001," skip this section, and go directly to "MSGADR*nnnt*" on page 156.

Note: Because there are no abends in the stand-alone restore program of DFSMSdss, this procedure does not apply to it.

Example:

```
5695DF175 R1K0 ABENDxxx function
```

Procedure: To determine the keywords for type-of-failure and function, follow these instructions:

1. Replace the *xxx* in the ABEND*xxx* keyword with the abend code from either the ending message or the abend dump.

---

2. Find the function keyword:

   Note: When a DFSMSdss task abends, the DFSMSdss scheduler module writes message ADR013I (TASK ABENDED) to the SYSPRINT (or its equivalent) data set. If you do *not* receive the ADR013I message, use function keyword CNTRL and refer to "Module Keyword" on page 160.

   a. Record the relative task identifier. The number in parentheses (*ttt*) that immediately follows the message identifier is the relative task identifier. For example:

      ```
      ADR013I (ttt)-mmmmm(yy), date_and_time TASK ABENDED . . .
      ```

   b. Check prior messages for an ADR101I message with a task identifier that matches the one that you found in step 2a. For example:

      ```
      ADR101I (ttt)-mmmmm(yy), TASKID xxx HAS BEEN ASSIGNED TO COMMAND 'command'
      ```

      The command that is identified in this message is the function that is executing at the time of the abend. This command name is the function keyword.

You have finished when you have identified the function keyword for the ABEND*xxx* type-of-failure keyword. Refer to "Module Keyword" on page 160 to

identify the maintenance level of the module that failed.

MSGADR*nnnt*

Use this procedure for any of the following conditions:

- A DFSMSdss message indicates that an internal program error has occurred (for example, message ADR799E, "AN UNEXPECTED ERROR HAS OCCURRED").
- A message is not issued when it should have been.
- A message is issued when it should not have been.

Example:

5695DF175 R1K0 MSGADR*nnnt* **ADR***mmmmm* **OC***yy*

Procedure: To determine the keywords for type-of-failure, module, and occurrence code, follow these instructions:

1. Replace the *nnnt* in **MSGADR***nnnt* with the message number and severity code. For example, if the message is ADR503A, the MSGADR*nnnt* type-of-failure keyword is MSGADR503A.

   ---

2. Replace the *mmmmm* in *ADRmmmmm* with the module identifier from the message (*mmmmm* in the following example).

   ADR013I (*ttt*)-*mmmmm*(*yy*), date_and_time TASK ABENDED . . .

   The number in the parentheses (*ttt*) immediately following the message identifier is the relative task identifier. The resulting module keyword is ADR*mmmmm*.

   Note: For the stand-alone restore program, the module keyword is always ADRDMPRS. If you select this keyword, go to "Maintenance-Level Keyword" on page 162.

   ---

3. Replace the *yy* in **OC***yy* with the two-character occurrence code, which is in parentheses following the module identifier in the message. For example, you could have **OC01**.

You have finished when you have identified the MSGADR*nnnt* type-of-failure keyword, the module keyword, and the occurrence code for the MSGADR*nnnt* type-of-failure. To identify the maintenance level of the module, refer to "Maintenance-Level Keyword" on page 162.

WAIT

Use this procedure when DFSMSdss suspends activity while it is waiting for a condition to be satisfied, yet it has not issued a message to tell why it is waiting. Ensure that the wait is not caused by your having specified WAIT subparameters that are too large.

Example:

5695DF175 R1k0 WAIT *function*

Procedure: To determine the keywords for type-of-failure and function, follow these instructions:

1. Use WAIT as the type-of-failure keyword if all DFSMSdss tasks were in a WAIT state.
2. Obtain an abend dump of the WAIT state. Ensure that the job control language (JCL) has a SYSABEND, SYSMDUMP, or SYSMDUMP data definition (DD) statement.

---

3. Follow these steps to find the function keyword:
   a. Record the relative task identifier. The relative task identifier is the number in parentheses (*ttt*) that immediately follows the message identifier in message ADR006I. Message ADR006I is printed when a DFSMSdss task begins. For example:

      ```
      ADR006I (ttt)-mmmmm(yy), date_and_time EXECUTION BEGINS
      ```

   b. Locate the tasks that are active (message ADR006I without a matching ADR013I). Message ADR013I is printed when a function ends. For example:

      ```
      ADR013I (ttt)-mmmmm(yy), date_and_time TASK COMPLETED
      ```

   c. Use CNTRL as the function keyword if no tasks are active, or if more than one function is active. Go to "Module Keyword" on page 160.
   d. Check prior messages for an ADR101I message with a task identifier that matches the one that you found in step 3a. For example:

      ```
      ADR101I (ttt)-mmmmm(yy), TASKID xxx HAS BEEN ASSIGNED TO COMMAND 'command'
      ```

      The command identified in this message is the function that is executing at the time that the wait started. This function name is the function keyword.

You have finished when you have identified the function keyword for the WAIT type-of-failure. To identify the maintenance level of the module that failed, refer to "Module Keyword" on page 160.

Guideline: For the stand-alone restore program, the module keyword is always ADRDMPRS. If you select this keyword, go to "Maintenance-Level Keyword" on page 162.

LOOP

Use this procedure when some part of the program repeats endlessly. If a message repeats endlessly, use the MSGADRnnnt keyword.

Example:

```
5695DF175 R1k0 LOOP function
```

Procedure: To determine the keywords for type-of-failure and function, follow these instructions:

1. Use LOOP as the type-of-failure keyword if a DFSMSdss task was in a loop.
2. Obtain an abend dump of the LOOP state. Ensure that the JCL has a SYSABEND, SYSMDUMP, or SYSUDUMP DD statement.

---

3. Follow these steps to find the function keyword:
   a. Record the relative task identifier. Message ADR006I is printed when a DFSMSdss task begins. The number in parentheses (*ttt*) immediately following the message identifier is the relative task identifier. For example:
      ```
      ADR006I (ttt)-mmmmm(yy), date_and_time EXECUTION BEGINS
      ```
   b. Locate the tasks that are active (message ADR006I without a matching ADR013I). Message ADR013I is printed when a function ends. For example:
      ```
      ADR013I (ttt)-mmmmm(yy), date_and_time TASK COMPLETED
      ```
   c. Use CNTRL as the function keyword if no tasks are active. Go to "Module Keyword" on page 160.
   d. Analyze the program further to determine which task has failed if more than one function is active. You can best accomplish this by examining each function task in the dump.
   e. Check prior messages for an ADR101I message with a task identifier that matches the one that you found in step 3a. For example:
      ```
      ADR101I (ttt)-mmmmm(yy), TASKID xxx HAS BEEN ASSIGNED TO COMMAND 'command'
      ```
      The command that is identified in this message is the function that is executing at the time the loop began. This function name is the function keyword.

You have finished when you have identified the function keyword for the LOOP type-of-failure. To identify the maintenance level of the module that failed, refer to "Module Keyword" on page 160.

Guideline: For the stand-alone restore program, the module keyword is always ADRDMPRS. If you select this keyword, go to "Maintenance-Level Keyword" on page 162.

INCORROUT

Use this procedure if you expect output but do not receive it, or if the output is not what you expected.

Examples:

```
5695DF175 R1k0 INCORROUT MSGADRnnnt ADRmmmmm OCyy function
 or
5695DF175 R1k0 INCORROUT function
```

Procedure: To determine the keywords for type-of-failure, module keyword, occurrence code, and function, follow these instructions:

1. Use INCORROUT as the type-of-failure keyword.

---

2. Perform steps 1 through 3 in "MSGADR*nnnt*" on page 156, if the output is in the form of an incorrect message, and then return to this procedure.

---

3. Use the name of the DFSMSdss function that you were using as the keyword. Choose from the following list:

& BUILDSA
CGCREATE
COMPRESS
CONVERTV
COPY
COPYDUMP
DEFRAG
DUMP
PRINT
RELEASE
RESTORE

You have now finished obtaining the keywords for type-of-failure, module keyword, occurrence code, and function. Refer to Chapter 15, “Using IBM Support Center,” on page 163.

DOC

Use this procedure when you encounter incorrect or missing information in the DFSMSdss documentation. For a minor publication error, submit a readers' comment form from the back of the publication in error. If the error is serious and of general concern to other users, continue with the procedure described below.

Example:

5695DF175 R1k0 DOC *xxnnnnnnnn*

Procedure: To determine the keywords for type-of-failure and document number, follow these instructions:

1. Use DOC as the type-of-failure keyword.

---

2. Place the order number of the document after the DOC keyword, omitting the hyphens. If the suffix is one digit, precede it with a zero. For example, the keyword for a document whose order number is SC35-0424-04 is listed below:

   ```
   DOC SC35042404
   ```

---

3. Locate the page in the document where the error occurred and prepare a description of the problem. If you submit an APAR, you must include this information in the error description.

You have now obtained the keywords for type-of-failure and document number. Refer to Chapter 15, “Using IBM Support Center,” on page 163.

PERFM

Use this procedure if performance is below explicitly stated expectations and the problem cannot be corrected by system tuning.

Example:

5695DF175 R1k0 PERFM *function*

Procedure: To determine the keywords for type-of-failure and function, follow these instructions:

1. Use PERFM as the type-of-failure keyword.

   ---

2. Use the name of the function as the function keyword if the performance problem occurred during one of the following functions:

   BUILDSA
   CGCREATE
   COMPRESS
   CONVERTV
   COPY
   COPYDUMP
   DEFRAG
   DUMP
   PRINT
   RELEASE
   RESTORE

You have now obtained the keywords for type-of-failure and function. Refer to Chapter 15, “Using IBM Support Center,” on page 163.

Module Keyword

This section applies only to the ABEND*xxx*, WAIT, and LOOP type-of-failure keywords.

DFSMSdss uses subtasking to isolate functions; thus, a dump might contain a task and subtasks as follows:

- A DFSMSdss scheduler task
- A subtask for the reader/interpreter
- Subtasks for the DFSMSdss functions (for example, COPY or DUMP)
- A subtask for managing SVC services
- A subtask for managing IGWFAMS
- A subtask for attached utilities

Each task has its own task control block (TCB) and request block (RB) chain in the dump. An explanation of the subtasks follows:

- The task for the scheduler is ADRDSSU, as indicated in the program request block/contents directory entry (PRB/CDE). Its normal state is waiting for the return from the EVENTS SVC (X'7D') in ADRDSSU.
- The subtask for the reader/interpreter is ADRRI01, as indicated in the PRB/CDE. Its normal state is waiting for the return from the WAIT SVC (X'01') in ADRRI01.

& • The subtasks for the DFSMSdss functions are ADRBLDSA, ADRCGCR,
& ADRCPYD, ADRPRNT, ADREFRAG, ADRDDTFP, ADRDTDS, ADRDTDSC,
& ADRDDDS, ADRCMPR0, ADRRLSE0, ADRTDFP, ADRTDDS, or ADRKVOL, as
& indicated in the PRB/CDE.
- The subtask that manages the SVC services is ADRSVCD. Its normal state is waiting for return from the WAIT SVC (X'01') in ADRSVCD.
- The subtask that manages IGWFAMS is ADRATFMS. Its normal state is waiting for the return from the WAIT SVC (X'01') in ADRATFMS.
- The subtask that manages other attached utilities is ADRMUTIL. Its normal state is waiting for the return from the WAIT SVC (X'01') in ADRMUTIL.

Procedure:

1. Locate the failing task by comparing the task identifier in the task-related messages to the task identifier in the function blocks for the executing functions. The task identifier is in the third halfword in the function block for the function. The function block is addressed by register 1 at entry to the function. This register is in the first save area on the save area chain under the TCB for the function. The ID in the message is a decimal number; the identifier in the function block is a hexadecimal number.

   If the function keyword is CNTRL, the failing task is either the scheduler or the reader/interpreter. If both are present, inspect the program status words (PSWs) for these tasks to see if one is not in its normal state, as defined above.

---

2. Locate the active module.

   Get the PSW for the failing task. This is usually found in the PRB and might not always be the error PSW identified at the beginning of the dump.

   Starting at the address in the PSW, search backward through the dump for the module identifier. The identifier consists of the following two character strings (separated by from 20 to 80 bytes):

```
ADRxxxxx mm/dd/yyHDZ11k0 aparnum
  .
  .
  .
ptfnum,yr.day,hh:mm:ss
```

   where:

ADR*xxxxx*	= Module name
mm/dd/yy	= Maintenance date
HDZ11k0	= FMID of the release
aparnum	= Current® APAR number or “NONE”
ptfnum	= Current program temporary fix (PTF) number or “NONE”
yr.day	= Compile date
hh:mm:ss	= Compile time

   If the PSW does not point within a DFSMSdss module, use register 12 (base) or 14 (return) at the time of the error and repeat the above search. You can usually find these registers in the first supervisor request block (SVRB) after the PRB for the failing task.

Your keyword set looks like one of the following:

5695DF175 R1k0 ABEND*xxx function module*

5695DF175 R1k0 WAIT *function module*

5695DF175 R1k0 LOOP *function module*

---

3. If you already know the PTF number, see Chapter 15, “Using IBM Support Center,” on page 163. Otherwise, go to Maintenance-Level Keyword.

Maintenance-Level Keyword

Use this keyword to identify the maintenance level of the module that failed.

Procedure:

1. Use the PTF number as the maintenance level keyword. The PTF number is found when you find the module identifier (see Step 2 in “Module Keyword” on page 160).

---

2. Use the following process to find the PTF number if it is not available as part of the module identifier:
   a. If you are using preventive service tapes, you can identify the maintenance level of the last tape applied to the system.
   b. Find the maintenance level of a module by listing the SMP/E control data set (CDS).
   c. Find the name of the module that the previous steps identified as the cause of the problem, in the name column of the module entries.
   d. Find the replacement module identifier (RMID) field in the entry for the module. The RMID field contains the PTF number that identifies the maintenance level of the module. For the entire maintenance history of this module (superzaps, user modifications, and so forth), a LIST CD MOD with the keyword XREF and the module name produces a SYSMOD history of this module.

You now have all the necessary information for an effective search of known problems in the Software Support Facility (SSF) database or EWS for APARs and PTFs. Call the IBM support center or go to Chapter 15, “Using IBM Support Center,” on page 163.

Chapter 15. Using IBM Support Center

IBM Support Center personnel have access to several software support databases. They use these databases and the set of keywords that you provide as a search argument to help solve your program failure. Support Center personnel may help you improve the effectiveness of your search argument. If someone already reported the problem, the Support Center personnel review the recorded failure and provide you with a suggested method of corrective action. The types of software support databases available to the IBM Support Center personnel include the following:

- Software support facility
- IBMLink/ServiceLink
- Info/System

Using the Software Support Facility

The software support facility is an IBM online database that contains information about all authorized program analysis reports (APARs) and program temporary fixes (PTFs). IBM Support Center personnel have access to SSF and are responsible for using the set of keywords that you provide as a search argument. Support Center personnel may help you improve the effectiveness of your search argument. They can also retrieve the records of previously reported problems. These records describe the failure and the corrective actions taken.

Using IBMLink/ServiceLink

IBMLink/ServiceLink is a set of online electronic services available to customers. Some of these services are available to you free of charge as a part of the SoftwareXcel basic contract. Some of these services are available for an additional fee as part of the optional SoftwareXcel Extended contract. Contact your local IBM marketing branch office for more information on SoftwareXcel contracts and services. The following services are available to you under one of these contracts:

SRCHSERVICE
: Online database of APAR and PTF information, with extensive search capability.

PSP
: Preventive Service Planning information database. This database contains the latest information concerning the installation of IBM products, including the latest service recommendations.

SRD
: Service Request and Delivery Facility. This facility provides a means for election ordering and delivery of corrective services, including PTFs and APARS.

ASAP
: Automatic Software Alert Process. This facility alerts the user when critical service information becomes available on a list of products that are selected by the user.

ETR
: Electronic Technical Response. The user may electronically report problems and ask appropriate technical questions about IBM products through this facility. The ETR answers questions and problem reports by utilizing electronic responses. Optionally, you can request to talk to an IBM representative about your problem

report. Submit nondefect-related, nontechnical questions to the question and answer (Q&A) queue in Canada on a severity 3, priority 3 basis.

AST Automatic Status Tracking. This facility allows the user to request notification when the status of a user-selected APAR or PTF changes, or both.

VPL View Program Listings. This is an online database of module listings for non-object code only (OCO) modules distributed through PTFs.

Info/System

Info/System is an interactive online database information retrieval program product. It is available primarily for use by customers with the companion data base feature, Info/MVS. The data base divides itself into several logical files of related or similar information, such as IBM's ServiceLink.

Appendix A. ACS Routine Information

This appendix contains general-use Programming Interface and Associated Guidance Information.

This appendix lists the variables available to automatic class selection (ACS) routines during DFSMSdss copy, restore, and CONVERTV operations, and provides additional information about ACS routine processing. This information is provided for guidance purposes only. It is not associated with any interface provided by DFSMSdss. For details on writing ACS routines, refer to *z/OS DFSMSdfp Storage Administration Reference*.

Messages generated by ACS routines are not printed by DFSMSdss unless the ACS routine returns a nonzero return code.

ACS Variables Available during Copy Function

When automatic class selection (ACS) is invoked during a DFSMSdss copy function, the following variables, as shown in Table 11 are passed to the ACS routines.

Table 11. Variables Passed to ACS Routines during DFSMSdss Copy Function. The following variables are not available to the storage group ACS routine: &ACCT_JOB, &ACCT_STEP, &DD, &JOB, &PGM, and &XMODE.

Variable Name	Description
&ACCT_JOB	Accounting information from the JOB statement.
&ACCT_ STEP	Accounting information on the EXEC statement.
&ACSENVIR	Environment in which ACS was invoked. Set to 'ALLOC.'
&ALLVOL	Output volume serial numbers. References the same volume list as &ANYVOL, but when used in a comparison, returns true only if all volume serial numbers satisfy the condition. &ALLVOL is not available to the storage group ACS routine when VOLCOUNT(ANY) is specified.
&ANYVOL	Output volume serial numbers. References the same volume list as &ALLVOL, but when used in a comparison, returns true if any volume serial numbers satisfy the condition. &ANYVOL is not available to the storage group ACS routine when VOLCOUNT(ANY) is specified.
&APPLIC	Application identifier associated with the data set (available only if RACF is installed).
&DD	DDNAME
&DEF_DATACLAS	Default data class name (available only if RACF is installed).
&DEF_MGMTCLAS	Default management class name (available only if RACF is installed).
&DEF_STORCLAS	Default storage class name (available only if RACF is installed).
&DSN	Data set name.
&DSNTYPE	Data set name type (for example: EXT, HFS, LIBRARY, PDS, or null).

Table 11. Variables Passed to ACS Routines during DFSMSdss Copy Function (continued). The following variables are not available to the storage group ACS routine: &ACCT_JOB, &ACCT_STEP, &DD, &JOB, &PGM, and &XMODE.

Variable Name	Description
&DSORG	Data set organization.
&DSOWNER	RACF owner or group that is considered the data set owner (available only if RACF is installed).
&DSTYPE	Data set type (for example, GDS, PERM, or TEMP).
&EXPDT	Expiration date.
&GROUP	Group identifier from the JOB statement.
&HLQ	High-level qualifier of the data set name.
&JOB	Job name, started task name, or TSO user ID from the JOB statement.
&LLQ	Low-level qualifier of the data set name.
&MAXSIZE	Maximum size of data set in kilobytes. For non-VSAM data sets, the primary value plus 15 secondary extents, or 122 secondary extents for PDSE and extended-format sequential data sets. For VSAM data sets, the primary value plus 122 secondary extents. See “Using SIZE and MAXSIZE Variables” on page 168 for more information about this value.
&NQUAL	Number of qualifiers in the data set name.
&NVOL	Number of output volumes specified by the user.
&PGM	Program name from the EXEC card.
&RECORG	Data set record organization.
&RETPD	Retention period.
&SIZE	Size of the data set in kilobytes. See “Using SIZE and MAXSIZE Variables” on page 168 for more information about this value.
&UNIT	Actual unit name (not esoteric names).
&USER	User ID from the JOB statement or the user ID propagated from the environment when a security package, such as RACF, is active.
&XMODE	Execution mode (for example, TSO, BATCH, or TASK).

ACS Variables Available during RESTORE and CONVERTV Processing

When ACS is invoked during DFSMSdss RESTORE or CONVERTV processing, the variables shown in Table 12 are passed to the ACS routines.

Table 12. Variables Passed to ACS Routines during DFSMSdss Restore and CONVERTV Processing

Variable Name	Description
&ACSENVIR	Environment in which ACS was invoked. Set to 'RECOVER' for RESTORE. Set to 'CONVERT' for CONVERTV.

Table 12. Variables Passed to ACS Routines during DFSMSdss Restore and CONVERTV Processing (continued)

Variable Name	Description
&ALLVOL	For restore processing, the output volume serial numbers. For CONVERTV processing, the volumes on which the data set resides. References the same volume list as &ANYVOL, but when used in a comparison, returns true only if all volume serial numbers satisfy the condition. &ALLVOL is not available to the storage group ACS routine when VOLCOUNT(ANY) is specified.
&ANYVOL	For restore processing, the output volume serial numbers. For CONVERTV processing, the volumes on which the data set resides. References the same volume list as &ALLVOL, but when used in a comparison, returns true if any volume serial numbers satisfy the condition. &ANYVOL is not available to the storage group ACS routine when VOLCOUNT(ANY) is specified.
&APPLIC	Application identifier associated with the data set (available only if RACF is installed).
&DEF_DATACLAS	Default data class name (available only if RACF is installed).
&DEF_MGMTCLAS	Default management class name (available only if RACF is installed).
&DEF_STORCLAS	Default storage class name (available only if RACF is installed).
&DSN	Data set name.
&DSNTYPE	Data set name type (for example: EXT, HFS, LIBRARY, PDS, or null).
&DSORG	Data set organization.
&DSOWNER	RACF owner or group that is considered the data set owner (available only if RACF is installed).
&DSTYPE	Data set type (for example, GDS, PERM, or TEMP).
&EXPDT	Expiration date.
&GROUP	Group identifier from the JOB statement. (This variable is passed to ACS routines during processing of RESTORE only, not CONVERTV.)
&HLQ	High-level qualifier of the data set name.
&LLQ	Low-level qualifier of the data set name.
&MAXSIZE	Maximum size of data set in kilobytes. For non-VSAM data sets, the primary value plus 15 secondary extents, or 122 secondary extents for PDSE and extended-format sequential data sets. For VSAM data sets, the primary value plus 122 secondary extents. See “Using SIZE and MAXSIZE Variables” on page 168 for more information about this value.
&NQUAL	Number of qualifiers in the data set name.
&NVOL	For restore processing, number of volumes specified by the user. For CONVERTV processing, the number of volumes (including candidate volumes) on which the data set resides.
&RECORG	Data set record organization.
&RETPD	Retention period.
&SIZE	Size of the data set in kilobytes. See “Using SIZE and MAXSIZE Variables” on page 168 for more information about this value.
&UNIT	Actual unit name (not esoteric names).

Table 12. Variables Passed to ACS Routines during DFSMSdss Restore and CONVERTV Processing (continued)

Variable Name	Description
&USER	User ID from the JOB statement or the user ID propagated from the environment when a security package, such as RACF, is active. (This variable is passed to ACS routines during processing of RESTORE only, not CONVERTV.)

Using SIZE and MAXSIZE Variables

The values for SIZE and MAXSIZE (and the space units the values represent) depend on the type of allocation of the data set. For all VSAM data sets, and non-VSAM data sets allocated in device dependent units (tracks or cylinders), the value represents the number of kilobytes using the maximum block size of the device. For example, the maximum block size for a 3390 is 56664. For all other non-VSAM data sets allocated in device independent units (blocks, average block, AVGREC=U, AVGREC=K, AVGREC=M), the value represents the number of data kilobytes (based on the average block size, specified block size, or 4096 if no block size is available).

The values DFSMSdss computes for SIZE and MAXSIZE may not match that of the original allocation. The values may be different if the device type on which the data set now resides is not the same as either of the following:

- The device type used at allocation, or
- The default device type in the CDS when the original allocation was done

DFSMSdss calculates the value based on the device type where the data set currently resides. DFSMSdss has no way of "knowing" what device type was specified or used for the original allocation.

DFSMSdss calculates SIZE and MAXSIZE variables as follows:

- For **PDS and PDSE data sets** — The values DFSMSdss computes may be different from those computed by SMS because SMS adds space for the directory. Also, DFSMSdss computes MAXSIZE for PDSE data sets based on 122 secondary extents.
- For **VSAM data sets** — DFSMSdss computes the SIZE and MAXSIZE for key-sequenced data sets (KSDS) from the current size and space values of the data component. The index component size is not included. Also, DFSMSdss computes MAXSIZE for VSAM data sets based on 122 secondary extents.
- For **extended-format sequential data sets** — DFSMSdss computes MAXSIZE for extended-format sequential data sets based on 122 secondary extents.
- For data sets allocated with **AVGREC=U, K, or M** — The size values computed during the initial allocation used the specified average block value. Since the average block size value is not stored anywhere and was only available during the initial allocation, DFSMSdss uses the DCB BLKSIZE value. If the DCB BLKSIZE value is not the same as the average block size value, the values computed for SIZE and MAXSIZE may be different from those computed at initial allocation.

Appendix B. Linux-z/OS DFSMSdss Dump or Restore HOW-TO

This appendix describes how to use z/OS DFSMSdss to back up and restore Linux for OS/390 or Linux for zSeries partitions and volumes. It is for storage administrators who are familiar with Linux for OS/390 or Linux for zSeries. The storage administrators should also have root authority on Linux and are authorized to run DFSMSdss batch jobs by IBM RACF, or similar security programs. You will also learn how to use a z/OS image running on either the OS/390 or zSeries hardware to back up Linux partitions that are attached to a Linux for OS/390 or zSeries image.

LINUX is a registered trademark of Linus Torvalds. You may direct your comments to linux@de.ibm.com.

Introduction

Backup

The term *mainframe* refers to the IBM OS/390 processor or the IBM eServer™ zSeries processor.

You can include your Linux volumes into an existing z/OS backup solution that uses DFSMSdss. Linux may dump volumes to tape or other direct access storage device (DASD) volumes. The procedures in Appendix B, "Linux-z/OS DFSMSdss Dump or Restore HOW-TO" were tested by using DFSMSdss release 10. OS/390 version 2 release 10, and z/OS version 1 release 1 are packaged with this release. These procedures may or may not work with earlier versions of DFSMS.

This backup solution is to work in the following two environments`:

- A z/OS-centric environment with z/OS running on several LPARs, in tandem with a few dozen Linux servers running within the virtual image facility (VIF)
- A Linux-focused environment with VM running in BASIC or LPAR mode. Hundreds of Linux guests and one or more z/OS images will perform the DFSMSdss processing

Requirements

Hardware Environment

The following are the three modes under which any operating system can run on the mainframe:

BASIC A single operating system image that owns the entire processor or all processors.

LPAR The processor can divide into 15 logical partitions, each partition running its own operating system image.

VIRTUAL IBM offers VM and VIF for Linux systems. Both VM and VIF are hypervisors (an operating system that allows other operating systems to run). VM supports hundreds to thousands of guests. Each guest produces its own operating system image. As an example, one guest using Linux, and another using z/OS, all on the same hardware.

DASD Background

The mainframe environment can store data on DASD by utilizing an extended count-key-data (ECKD) protocol. DFSMSdss divides the DASD into logical units known as volumes. You can define a volume to any size, up to a maximum size of approximately 9 GB. Volumes divide into fixed-size tracks. Its geometry determines the size of the track. Linux supports both the 3380 and 3390 track geometries. You can reference a volume by using a 16-bit device number, and a six-character volume serial number (referred to as the *volser*).

Linux Requirements

You must format the Linux volumes in the compatible disk layout (cdl) using dasdfmt version 1.0, and partitioned using fdasd version 1.0.

Linux Disk Utilities

dasdfmt: The default disk layout for dasdfmt is cdl. Volumes formatted in the original Linux disk layout (ldl) are not compatible with the z/OS system. z/OS cannot back them up. Below is an example of how to format a disk with dasdfmt at address 0198, having a byte block size of 4096, and a *volser* of LNX200:

```
dasdfmt -n 198 -b 4096 -l lnx200
```

The screen should appear like the one below:

```
Drive Geometry: 3339 Cylinders * 15 Heads =  50085 Tracks
I am going to format the device 198 in the following way:
    Device number of device : 0x198
    Major number of device  : 94
    Minor number of device  : 8
    Labelling device        : yes
    Disk label              : VOL1
    Disk identifier         : LNX200
    Extent start (trk no)   : 0
    Extent end (trk no)     : 50084
    Compatible Disk Layout  : yes
    Blocksize               : 4096

 --->> ATTENTION! <<--- All data in the specified range of that device
will be lost. Type "yes" to continue, no will leave the disk untouched.
```

You can see a 3339 cylinder volume that is attached to Linux at address 198 in the screen capture above. The volume serial number (*volser*) is LNX200, and its block size is 4096 bytes. The *volser* must be six characters in length. The disk label, VOL1, indicates that a z/OS system can process this volume.

A classic Linux volume will have a disk label of LNX1. z/OS cannot process any volume with a disk label of LNX1.

fdasd: After you have formatted a volume with dasdfmt in the compatible disk layout, you must partition it before Linux can use it. You should use the fdasd program to partition the volume. This program is similar to the fdisk program that comes with the Linux version that runs on personal computers. One difference is that it creates partitions on ECKD DASD instead of hard drives. With fdasd, you can create up to three partitions on a volume. The partitions will appear to z/OS as data sets, and you can choose the size of a partition.

Below are some rules that you should follow when creating partitions with fdasd that you want to back up with z/OS tools.

- Create partitions by starting from the lowest possible track.
- Do not leave gaps between partitions.
- If you want to restore a partition, ***do not*** delete it with fdasd first. (You rename the partition names and the data set names when you delete a partition with fdasd.)

Authorization Requirements

You need to have root authority for Linux in order to mount and unmount the partitions, as well as to format and partition the volumes using dasdfmt and fdasd. For z/OS you need authority to run ADRDSSU, which is the program that is invoked when using DFSMSdss. z/OS sees the Linux partitions as data sets. You can prevent unauthorized access of the Linux partitions by z/OS applications and users by using a security product like the IBM RACF Security Server.

Backing up a Linux Volume with Partitions

z/OS sees a partition on the Linux side (For example: /dev/dasd/0198/part1) as a data set. The data set is named LINUX.V*volser*.PART000*x*.NATIVE for a data partition, or LINUX.V*volser*.PART000*x*.SWAP for a swap partition. The *volser* is the volume serial number given to the volume when dasdfmt formatted the volume. fdasd can change the *volser*, as well. The *volser* must be unique in order for the z/OS system to process it. The *x* in PART000*x* is most likely the partition number, minus one. For example, a Linux partition such as /dev/dasd/0198/part2 would be known to z/OS as the data set LINUX.VLNX200.PART0001.NATIVE, where LNX200 is the volume serial number of the volume.

CAUTION:
If a partition is still mounted read/write while undergoing a DUMP, you may not back up data written to the partition during the DUMP. Because Linux uses deferred writes, unmounting a partition or remounting a partition read-only also serves to flush Linux's internal memory buffers to disk. You should process the DUMP when Linux is down, or when the partitions currently being backed up are unmounted or mounted read-only. If the partitions are still mounted read/write, DFSMSdss will be able to back up your data, but the data may be inconsistent. By unmounting or remounting a partition read-only, you may ensure that all of your data has been backed up. You are not required to do this, but it provides the best copy. Below is an example of mounting a partition read-only:

```
mount -t ext2 -r /dev/dasd/019b/part1 /mntpoint
```

The data sets and the partitions they represent will have the following naming convention:

Data Set Names	Partition Names
LINUX.V*volser*.PART0000.type	/dev/dasd/yyyy/part1
LINUX.V*volser*.PART0001.type	/dev/dasd/yyyy/part2
LINUX.V*volser*.PART0002.type	/dev/dasd/yyyy/part3

where *volser* is the volume serial number of the volume where the data set resides. *yyyy* is the device number of the volume in the Linux environment, *type* can be NATIVE or SWAP. You might decide that you need to back up only certain NATIVE partitions. SWAP partitions are the Linux equivalent to z/OS page packs.

You should not rename the data sets. fdasd expects the 24th character to be an 'N' or an 'S'. If it is something else, fdasd will be unable to recognize the partition type.

DFSMSdss Commands: For DFSMSdss, a Time Sharing Option (TSO) user ID can submit JCL to the z/OS system, or through ftp from the Linux system. You can find JCL rules and information in the MVS JCL manual online. IBM recommends using only the ADRDSSU keywords shown in the screen captures below. The keywords are between //SYSIN and /*.

Note: You must specify ALLEXCP when specifying the keywords for any DFSMSdss DUMP batch job or COPY batch job that will process Linux cdl volumes.

Example 1. DUMP FULL

The DFSMSdss DUMP FULL command dumps the entire contents of a volume to tape or DASD. DFSMSdss can restore this dump later. After you have a dump of the boot volume, you can restore that volume to multiple volumes as a way of making identical copies of the basic Linux system. The DFSMSdss keywords that you can use are DUMP FULL and ALLEXCP. You must specify ALLEXCP because it tells DFSMSdss to process all of the data set/partition even though it looks like it might be unused. Your data is not backed up if you do not specify ALLEXCP. The Linux volume has *volser* LNX200. Some sample JCL is below:

```
//LXD2D1BB JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT02  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//TARGET   DD UNIT=TAPE,VOL=(PRIVAT,SER=111111),DISP=(NEW,CATLG),
//            DSN=TDS.DUMP200,LABEL=(1,SL)
//SYSIN    DD *
 DUMP FULL INDDNAME(SOURCE) OUTDDNAME(TARGET) -
       ALLEXCP
/*
```

You can also dump to two or more output tapes at the same time. The example JCL is below. You might use this feature if you wanted to make a backup and a copy of the backup for storage off-site.

```
//LXD2D1XX JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT02  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//TARGET1  DD UNIT=TAPE,VOL=(PRIVAT,SER=111111),DISP=(NEW,CATLG),
//            DSN=TDS.DUMP200,LABEL=(1,SL)
//TARGET2  DD UNIT=TAPE,VOL=(PRIVAT,SER=222222),DISP=(NEW,KEEP),
//            DSN=TDS.DUMP200,LABEL=(1,SL)
//TARGET3  DD UNIT=3390,VOL=SER=WRKVOL,DISP=(NEW,KEEP),
//            DSN=TDS.DUMP200
//SYSIN    DD *  DUMP FULL INDDNAME(SOURCE) OUTDDNAME(TARGET1,TARGET2,TARGET3) -
               ALLEXCP
/*
```

Example 2. DUMP FULL with CONCURRENT COPY

DUMP FULL with CONCURRENT COPY allows you to use the volume being dumped without the dump being affected, much sooner than with the plain full volume dump. You might want to use this when the Linux partitions that are being backed up need to be available in read/write mode. Notice that the JCL specified ALLEXCP. Once you see message ADR734I (below), you can remount the partitions to the Linux system and continue using them.

```
ADR734I (001)-T0MI (03), 2001.168 14:38:22 CONCURRENT COPY INITIALIZATION
SUCCESSFUL FOR VOLUME LNX200. SERIALIZATION FOR THIS DATA IS RELEASED IF DFSMSDSS
HELD IT. THE INTERMEDIATE RETURN CODE IS 0000.
```

The dump will reflect whatever data was present when the concurrent copy job started. An example of DUMP FULL with CONCURRENT COPY JCL is below:

```
//LXD2D2BB JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT02  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//TARGET   DD UNIT=TAPE,VOL=(PRIVAT,SER=111111),DISP=(NEW,CATLG),
//            DSN=TDS.DUMP200,LABEL=(1,SL)
//SYSIN    DD *
 DUMP FULL INDDNAME(SOURCE) OUTDDNAME(TARGET) -
       CONCURRENT ALLEXCP
/*
```

Example 3. DUMP DATASET

You can dump individual partitions by using physical processing. This may help if you have a swap partition on a particular volume and you only want to back up the native, data holding partitions. There is probably no reason to back up a swap partition. An example of DUMP DATASET JCL is below:

```
//LXD2J1BB JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT03  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//TARGET   DD UNIT=TAPE,VOL=(PRIVAT,SER=111111),DISP=(NEW,CATLG),
//            DSN=TDS.DUMP200,LABEL=(1,SL)
//SYSIN    DD *
 DUMP INDDNAME(SOURCE) OUTDDNAME(TARGET) -
       DATASET(INCLUDE(LINUX.**.NATIVE)) ALLEXCP
/*
```

You can also dump all of the Linux partitions with the following JCL:

```
//LXD2J2BB JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=0M,
//      MSGCLASS=H,CLASS=A
//STEPT03  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DASDIN   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//DASDOUT  DD UNIT=3390,VOL=SER=D9BIG1,DISP=(NEW,CATLG),
//            SPACE=(CYL,(4300,1000),RLSE),DSN=TDS.DUMP200
//SYSIN    DD *
 DUMP INDDNAME(DASDIN) OUTDDNAME(DASDOUT) -
       DATASET(INCLUDE(LINUX.**)) CONCURRENT ALLEXCP
/*
```

Example 4. COPY FULL

You can also make a full volume copy of your volume. You may want to use this if you want to populate a new server with a standard configuration.

The COPY FULL function can also be useful because of SnapShot (for RAMAC Virtual Array devices [RVA]), or FlashCopy (for Enterprise Storage Server devices [ESS]). DFSMSdss attempts to use the fastest copy method possible before using the traditional data movement methods. SnapShot or FlashCopy make a virtually instantaneous copy of the volume, allowing you to continue using the volume.

In order for SnapShot to work, the volume (to which you make a copy) must be within the same subsystem as the source volume. In an RVA box, the four SSIDs that are defined to the RVA are considered to be within the same subsystem. Likewise for ESS boxes, one of the following conditions must be met for FlashCopy to be used:

- The volumes must be in the same ESS that supports FlashCopy Version 2.
- The volumes must be in the same logical subsystem in an ESS that supports FlashCopy Version 1 but does not support FlashCopy Version 2 or later functions.

After the plain COPY FULL command, the data on LNX900 will be the same as LNX200; except that the *volser* will still be LNX900.

```
//LXD2C1BB JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT02  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//TARGET   DD UNIT=3390,VOL=SER=LNX900,DISP=OLD
//SYSIN    DD *
 COPY  FULL INDDNAME(SOURCE) OUTDDNAME(TARGET) -
       ALLEXCP
/*
```

Example 5. COPY FULL COPYVOLID ALLEXCP

The following example shows how you can create a backup copy of a Linux volume. COPYVOLID specifies that DFSMSdss should copy the volume serial number (*volser*) to the new volume. After the copy, the two volumes are identical, including the *volser*. z/OS only allows one volume with a *volser* to be online to z/OS at a time. DFSMSdss varies the volume that was the target of the copy, offline, after making the copy. You must specify ALLEXCP because it tells

DFSMSdss to process all of the data set/partition even though it looks like it may be unused. Your data will not be backed up if you do not specify ALLEXCP.

```
//LXD2C2BB JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT02  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//TARGET   DD UNIT=3390,VOL=SER=LNX900,DISP=OLD
//SYSIN    DD *
 COPY  FULL INDDNAME(SOURCE) OUTDDNAME(TARGET) -
       COPYVOLID ALLEXCP
/*
```

Example 6. RESTORE FULL

You can restore a full volume from a full volume dump taken by DFSMSdss. This restores the entire contents from the original volume. You can use this command when you want a Linux volume from a DFSMSdss dump that was created earlier. You must specify the RESTORE command with the FULL keyword. You can find sample JCL below the next paragraph.

The Linux volume is *volser* LNX200. The dump that was stored in data set TDS.DUMP200 is being restored to LNX200. The SOURCE DD statement describes the place where DFSMSdss should find the information to restore. The TARGET DD statement describes to where DFSMSdss should restore the volume information and data. You may overwrite any data sets that are on the LNX200 volume currently with unexpired dates when you specify PURGE. All Linux partitions are permanent, and thus have “never expire” dates.

```
//LXD2D1BB JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=A,CLASS=A
//STEPT03  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,DISP=OLD,DSN=TDS.DUMP200
//TARGET   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//SYSIN    DD *
 RESTORE FULL INDDNAME(SOURCE) OUTDDNAME(TARGET) PURGE
/*
```

Example 7. RESTORE DATASET

You can restore individual partitions/data sets that were part of a full volume dump or that were from a data set level dump previously taken by DFSMSdss. You might want to do this to restore those particular partitions that were corrupted. Restoration of data sets from a data set level dump is similar to restoring full volumes. Notice now that the keyword REPLACE is specified instead of PURGE. For data set level restores, REPLACE says to replace any data sets existing on the volume with the restored versions. If the data set you are restoring exists and you do not specify REPLACE, the restore will fail and you will not get the backup version. The INCLUDE statement says to restore any data sets that start with LINUX and end with NATIVE (and have anything in-between). The ’**’ means any number of eight-letter qualifiers. Be careful to use two asterisks (one asterisk has a different meaning).

When preparing to restore a partition, you should not use fdasd to delete the partition, before running DFSMSdss to restore it. When fdasd deletes a partition, it

reorders and renames the remaining partitions on the same volume. A subsequent restore can result in the wrong partition being overlaid.

If you want to restore a deleted partition or a partition that never existed, use fdasd. fdasd can create a new partition that is exactly the same size and in the same location as the deleted partition. Use the same starting and ending track. fdasd will create the correct names for the data sets. When you restore that data set/partition, the data will be put in the right place and you will not lose any partitions. That is because the name of the second partition was the same as the restored first partition.

For example, if you delete /dev/dasd/xxxx/part1 (known to z/OS as LINUX.VLNX200.PART0000.NATIVE), fdasd renames the other partitions (fdasd subtracts one from the former name). part2 becomes part1 and part3 becomes part2. The data set names change as well. After fdasd deletes LINUX.VLNX200.PART0000.NATIVE, it renames LINUX.VLNX200.PART0001.NATIVE to LINUX.VLNX200.PART0000.NATIVE. If you then use DFSMSdss to restore the first partition (named LINUX.VLNX200.PART0000.NATIVE), you will lose the second partition.

```
//LXD2S1BB JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=0M,
//      MSGCLASS=A,CLASS=A
//STEPT03  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,DISP=OLD,DSN=TDS.DUMP200
//TARGET   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//SYSIN    DD *
 RESTORE INDDNAME(SOURCE) OUTDDNAME(TARGET) -
         DATASET(INCLUDE(LINUX.**.NATIVE)) REPLACE
/*
```

You can also use the RENAMEUNCONDITIONAL keyword to change the names of the data sets that you are restoring. If you are changing the data set, make sure that you only change the *volser* or the last character of the partition name (PART000*x*). If you change anything else, Linux may not recognize the partition.

```
//LXD2S1XX JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=0M,
//      MSGCLASS=A,CLASS=A
//STEPT03  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,DISP=OLD,DSN=TDS.DUMP200
//TARGET   DD UNIT=3390,VOL=SER=LNX300,DISP=OLD
//SYSIN    DD *  RESTORE INDDNAME(SOURCE) OUTDDNAME(TARGET) -
           DATASET(INCLUDE(LINUX.VLNX200.PART0001.NATIVE))
           RENAMEUNCONDITIONAL(LINUX.VLNX200.PART0001.NATIVE,LINUX.VLNX300.PART0001.NATIVE)
/*
```

Consider the following example:

You have three partitions on a volume where partition 1 has programs, partition 2 has data, and partition 3 is a swap partition. Perhaps someone who had root authority deleted most of the programs on partition 1. You want to restore the backup versions of those programs, but leave the data partition (partition 2) alone. You may not have to restore Partition 3 because it is swap space. The JCL below restores only the first partition, LINUX.VLNX200.PART0000.NATIVE.

```
//LXD2S1BB JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=0M,
//      MSGCLASS=A,CLASS=A
//STEPT03  EXEC PGM=ADRDSSU //SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,DISP=OLD,DSN=TDS.DUMP200
//TARGET   DD UNIT=3390,VOL=SER=LNX200,DISP=OLD
//SYSIN    DD *  RESTORE INDDNAME(SOURCE) OUTDDNAME(TARGET) -
           DATASET(INCLUDE(LINUX.VLNX200.PART0000.NATIVE)) REPLACE
/*
```

Example 8. COPYDUMP

DFSMSdss also supports the copying of Linux volume dumps. You can copy a DUMP immediately after creating the DUMP, or you can make a copy of an old DUMP. You might want to do this if you need to have copies of your DUMP tapes. Some installations will have an on-site backup tape as well as an off-site copy of the DUMP, in case of disaster. DFSMSdss uses the COPYDUMP keyword to copy the dumps. Below is some sample JCL:

```
//LXDRP1AA JOB ,'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=A,CLASS=A
//STEPT03  EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DASDIN   DD UNIT=3390,VOL=SER=D9XWRK,DISP=SHR,
//            DSN=TDS.BACKUP.DUMP200
//DASDOUT  DD UNIT=TAPE,VOL=SER=D9XWRK,DISP=(NEW,CATLG),
//            DSN=TDS.DUMP200,SPACE=(CYL,(225,10),RLSE)
//SYSIN    DD *
 COPYDUMP INDDNAME(DASDIN) OUTDDNAME(DASDOUT)
/*
```

Example 9. BUILD STAND-ALONE

Stand-Alone is a DFSMSdss function that lets you create an ipl-able (bootable) image on tape that can be used to restore dumps created by DFSMSdss. You can use the ipl-able image to bring up DFSMSdss Stand-Alone Restore to restore a Linux volume that was backed up by DFSMSdss. You can do this without having to bring up OS/390 or z/OS. Refer to "DFSMSdss Stand-Alone Services" in the *z/OS DFSMSdss Storage Administration Reference* for a list of devices you can use with the Stand-Alone DFSMSdss restore program.

Submitting JCL batch jobs to a z/OS image using ftp: You can submit your JCL batch job to a z/OS image from a Linux image. When you ftp to the z/OS image, you must issue the `site file=jes` command. The ftp server on the z/OS image should rout any file that it receives to the Job Entry Subsystem (JES) for execution. Of course, your login id must have sufficient authority to run DFSMSdss batch jobs. Please see "Authorization Requirements" on page 171.

Finally, "put" your JCL jobs to the z/OS image. You should save your jobs as text files that do not exceed 80 characters in length.

JCL Information and Rules: You can find JCL information in the MVS JCL manual online.
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/iea2b600/CCONTENTS

Volume Serial Rules: When choosing a Volume Serial number (*volser*) for a volume, follow these rules:

- The *volser* should be six characters long. z/OS will accept fewer than six characters for the *volser*, but does not support that for Linux volumes that you want to back up using z/OS tools.
- The *volser* can be uppercase, alphanumeric characters (A-Z, 0-9), and the national characters ($, #, @).

Appendix C. Format of the DFSMSdss Dump Data Set

This appendix describes the format of the DFSMSdss dump data set. It also contains two data area descriptions that correspond to data areas ADRBMB (Table 13 on page 181) and ADRTAPB (Table 14 on page 181).

Related reading

- For information about ADRUFO, see *z/OS DFSMS Installation Exits*.
- For information about ADREID0, see the *z/OS DFSMSdss Storage Administration Reference*.

Format of the DFSMSdss Dump Data Set

& • For a physical dump, the Volume record also contains an Encryption record
& following the volume header when encryption was performed during the dump.
& • For a logical dump, the Tape header record also contains an Encryption record
& following the DFSMSdss tape header when encryption was performed during
& the dump.
& • If hardware-assisted compression was used (HWCOMPRESS keyword), a data
& track record may be preceded by an expansion dictionary record.

For a **physical dump**, each logical volume of a DFSMSdss dump data set contains the following data in the following sequence:

1. Volume header record, which identifies and contains data pertinent to the whole volume and identifies the type of operation that created the dump.
2. Map record or records, which map the tracks that were dumped. The data in this record is described by the "ADRBMB Data Area" on page 181.
3. Track 0: dump of track 0 of cylinder 0.
4. VSAM volume data set (VVDS) track records, if VSAM data sets exist on the volume and were dumped.
5. Volume table of contents (VTOC) track records.
6. Data track records, which include VVDS if it is part of the dump. This item is repeated for all tracks being dumped.
7. Two volume trailer records, which identify the end of the data for the DASD volume.

For a **logical dump**, the format of the DFSMSdss dump data set contains the following data in the following sequence:

1. Tape header record.
2. List of potential data sets record.
3. Data set header record.
4. Volume header record (one for non-VSAM data sets, one for each VSAM component).
5. Sphere information record (if the SPHERE keyword was specified and this data set is part of a sphere).
6. Data track records (one or more for each track of the data set on this volume).
7. Two data set trailer records.

Notes:

1. For each data set, repeat items 3 through 7.
2. For each volume of the data set, repeat items 4 and 6.

ADRTAPB Data Area

Every record on a DFSMSdss dump data set is described by the “ADRTAPB Data Area” on page 181.

ADRBMB Data Area

Table 13. ADRBMB Mapping Macro

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	CHARACTER	*	ADRBMB	
0	(0)	CHARACTER	12	BMHDR	BITMAP HEADER
0	(0)	CHARACTER	4	BMID	BLK IDENTIFIER EBCDIC “BMBB”
4	(4)	ADDRESS	4	BMNPTR	ADDR OF NEXT BITMAP SEGMENT/0
8	(8)	UNSIGNED	4	BMNTRK	NUMBER OF TRACKS MAPPED BY SEGMENT
8	(8)	UNSIGNED	2	BMNTRKTP	NUMBER OF TRACKS MAPPED BY SEGMENT
10	(A)	CHARACTER	2	*	RESERVED
12	(C)	BITSTRING	0	BMBITMAP	BITMAP LAYOUT, CONTIGUOUS BITS FOR EACH TRACK. 1=OPERATE ON CORRESPONDING TRACK, 0=SKIP TRACK

ADRBMB Cross-Reference

Name	Hex Offset
ADRBMB	0
BMBITMAP	12
BMHDR	0
BMID	0
BMNPTR	4
BMNTRK	8

ADRTAPB Data Area

Table 14. ADRTAPB Mapping Macro

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	16	ADRTAPB	
0	(0)	CHARACTER	6	DTPSCHK	
0	(0)	SIGNED	4	DTPSEQNO	SEGMENT SEQUENCE NUMBER
4	(4)	UNSIGNED	1	DTPNOSEG	NUMBER OF SEGMENTS PER RECORD
5	(5)	UNSIGNED	1	DTPSEGNO	SEGMENT NUMBER OF RECORD
6	(6)	UNSIGNED	2	DTPSEGLN	SEGMENT LENGTH INCLUDING PREFIX
8	(8)	UNSIGNED	1	DTPPFXLN	LENGTH OF PREFIX(CONSTANT 16)
9	(9)	CHARACTER	1	DTPDMPID	TYPE OF DUMP ID
		1...		DTPFULD	80 - FULL DUMP
		.1..		DTPPARTD	40 - PARTIAL DUMP
		..1.		DTPDASDD	20 - DATASET DUMP
		...1		DTPLOGCL	10 - LOGICAL DUMP - THIS BIT IS CALLED DTPCATDD IN V2.1, V2.2 - DTPLOGCL MUST BE 4TH BIT IN STRUCTURE TO MAINTAIN COEXISTENCE BETWEEN V2.1, V2.2, AND V2.3.
		 1111		*	RESERVED - DO NOT USE LAST 4 BITS IN DTPDMPID TO ENSURE DOWNWARD COEXISTENCE.
10	(A)	BITSTRING	1	DTPRCID1	RECORD IDENTIFIER 1
		1...		DTPVHDR	80 - VOLUME HEADER (SEE DTVOL)
		1...		DTPTHDR	- TAPE HEADER (SEE DTHDR)
		.1..		DTPBTM	40 - MAP OF DUMPED TRACKS (SEE DTBTMR)

& *Table 14. ADRTAPB Mapping Macro (continued)*

& Offsets					
& Dec	Hex	Type	Len	Name (Dim)	Description
&		.1..		DTPDSNL	- DS NAME/CATALOG LST (SEE DTLDSN)
&		..1.		DTPTRK0	20 - TRACK 0 (SEE DTTTRK)
&		..1.		DTPDSHDR	- DATA SET™ HEADER (SEE DTDSHDR)
&		...1		DTPVTOC	10 - VTOC TRACK (SEE DTTTRK)
&		...1		DTPVOLD	- VOLUME DEFINITION (SEE DTMVOL)
&		.... 1...		DTPDATA	08 - DATA TRACK (SEE DTTTRK)
&		.... .1..		DTPVTRLR	04 - VOLUME TRAILER (SEE DTRTLR)
&		.... .1..		DTPDTRLR	- DS TRAILER (SEE DTRTLR)
&		.... ..1.		DTPVVDS	02 - VVDS TRACK (SEE DTTTRK)
&		1		DTPSPHDR	01 - SPHERE RECORD HEADER (SEE DTSPHERE)
& 11	(B)	BITSTRING	1	DTPRCFL1	FLAG BYTE
&		1...		DTPDDISP	IF ON, DATA EQUAL TO LENGTH OF ADRTAPB HAS BEEN DISPLACED FROM THIS SEGMENT TO THE LAST SEGMENT OF THE TRACK.
&		.1..		DTPDDDSP	IF ON, LAST SEGMENT DISPLACED DATA MUST BE REPLACED FIRST (SEE DTPDDISP).
&		..1.		DTPENDNT	1=END OF NONTRACK DATA SET (FOR EXAMPLE, CDF DATA SET OR VSAM DATA SET DUMPED BY VSAM I/O)
&		...1		DTPENDKR	1=END OF A KEY RANGE FOR DS DUMPED BY VSAM I/O
&		.... 1...		DTPBWOE	1=BWODSN ONLY USED FOR DUMP OF DSET (OPEN DSET)
&		.... .1..		DTPDUMPC	SOURCE FROM THE DUMPCONDITIONING VOLUME
&		.... ..1.		DTPSELFD	SELF DESCRIBING RECORDS EXIST
&		1		*	RESERVED
& 12	(C)	CHARACTER	4	*	RESERVED
& 16	(10)	CHARACTER		DTPBODY	START OF REMAINDER OF RECORD
&					
& **Note:** REST OF VOLUME HEADER RECORD					
& 0	(0)	STRUCTURE	46	DTVOL	
& 0	(0)	CHARACTER	4	DTVTOCB	VTOC BEGINNING CCHH
& 4	(4)	CHARACTER	4	DTVTOCE	VTOC ENDING CCHH
& 8	(8)	CHARACTER	4	DTVOLSZ	VOLUME SIZE(DS4DEVSZ)
& 8	(8)	UNSIGNED	2	DTVLOGCY	NUMBER OF CYLINDERS IN VOLUME
& 10	(A)	UNSIGNED	2	DTVTRKCY	NUMBER OF TRACKS PER CYLINDER
& 12	(C)	UNSIGNED	2	DTVBLKSZ	MAXIMUM BLOCKSIZE
& 14	(E)	UNSIGNED	2	DTVMAXCB	MAXIMUM COMPRESS BUFFER IN WORDS
& 16	(10)	CHARACTER	6	DTVSERL	VOLUME SERIAL OF SOURCE VOL
& 22	(16)	CHARACTER	1	DTVVTOCI	VTOC INDICATORS
& 23	(17)	CHARACTER	8	DTVTIMD	DATE AND TIME OF DAY OF DUMP
& 23	(17)	CHARACTER	4	DTVDAY	DATE - 00YYDDDC
& 27	(1B)	CHARACTER	4	DTVTIME	TIME OF DAY IN DECIMAL
& 31	(1F)	CHARACTER	4	DTVDEVTY	DEVTYP OF VOLUME(UCBTBYT4)
& 35	(23)	UNSIGNED	1	DTVMODNO	MODEL NUMBER
& 36	(24)	BITSTRING	1	DTVIND1	VOLUME TYPE INDICATORS
&		1...		DTVVIRT	80 - VIRTUAL VOLUME
&		.1..		DTVMINI	40 - MINI DISK
&		..1.		DTVCVAF	20 - VOLUME HAS INDEXED VTOC
&		...1		DTVCPVOL	10 - CP VOLUME FORMAT
&		.... 1...		DTVFCMP	08 - FILE COMPRESSED
&		.... .1..		DTVUNLCD	04 - UNALLOCATED SPACE DUMPED
&		.... ..1.		DTVLVF	02 - OTHER LOGICAL VOLUMES MAY FOLLOW THIS LOGICAL VOLUME

Table 14. ADRTAPB Mapping Macro (continued)

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
		1		DTVVVDS	01 - VVDS DATA SET DUMPED BEFORE VTOC
37	(25)	UNSIGNED	2	DTVBMSZ	BITMAP SIZE IN WORDS
39	(27)	BITSTRING	1	DTVIND2	VOLUME TYPE INDICATORS
		1...		DTVLNVI	80-NONVSAM DATA SETS NOT ON VOLUME
		.1..		DTVLVI	40-VSAM DATA SETS NOT ON VOLUME
		..1.		DTVGNVI	20-NONVSAM DATA SETS NOT ON ANY VOLUME
		...1		DTVGVI	10-VSAM DATA SETS NOT ON ANY VOLUME
		.... 1...		DTVSMS	SOURCE VOLUME IS SMS MANAGED
		.... .1..		DTVSMSI	SOURCE VOL IS SMS INITIAL STAGE
		.... ..1.		DTVFHCMP	HARDWARE COMPRESSION USED
		1		*	UNUSED
40	(28)	UNSIGNED	2	DTVLEN	LEN OF HEADER (SEGMENT LEN - PREFIX
42	(2A)	UNSIGNED	1	DTVVERNO	DFSMSdss VERSION NUMBER
43	ebb	UNSIGNED	1	DTVLVLNO	DFSMSdss MODIFICATION NUMBER
44	(2C)	UNSIGNED	1	DTVVXTNO	NUMBER OF VVDS EXTENTS
45	(2D)	UNSIGNED	1	DTVVXTOF	OFFSET TO VVDS EXTENT
0	(0)	STRUCTURE	10	DTVVXTNT (*)	VVDS EXTENTS, IN DSCB FORMAT
0	(0)	CHARACTER	10	DTVVXLEN	
Note: REST OF TAPE HEADER RECORD (CATALOG FILTERING)					
0	(0)	STRUCTURE	18	DTHDR	
0	(0)	CHARACTER	8	DTHTIMD	DATE AND TIME/DAY OF DUMP
0	(0)	CHARACTER	4	DTHDAY	DAY
4	(4)	CHARACTER	4	DTHTIME	TIME
8	(8)	CHARACTER	1	DTHIND2	DATA SET TYPE INDICATORS
		1...		DTHGNVI	NO NON VSAM DATA SETS
		.1..		DTHGVI	NO VSAM DATA SETS
		..1.		DTHGT64K	ON=MORE THAN 65535 DATA SETS ON VOLUME
		...1 1111		*	RESERVED
9	(9)	UNSIGNED	2	DTHLEN	HEADER LEN
11	(B)	UNSIGNED	1	DTHVERNO	DFSMSdss VERSION NUMBER
12	(C)	UNSIGNED	1	DTHLVLNO	DFSMSdss MODIFICATION NUMBER
13	(D)	UNSIGNED	2	DTHBLKSZ	MAXIMUM BLKSIZE
15	(F)	UNSIGNED	2	DTHNDS	NUMBER OF DATA SETS IN LIST
17	(11)	CHARACTER	1	DTHIND1	INDICATORS
		1...		DTHFCMP	FILE COMPRESSED
		.1..		DTHUNLCD	UNALLOCATED SPACE DUMPED
		..1.		DTHSFER	SPHERE OPTION
		...1		DTHFHCMP	HARDWARE COMPRESSION USED
		.... 1...		DTFEFSAM	EXTENDED FMT SAM DS HAVE DTTTRK
		.... .111		*	RESERVED
Note: REST OF DATA SET NAME/CATALOG LIST RECORD					
0	(0)	STRUCTURE	45	DTLDSN	
0	(0)	UNSIGNED	1	DTLLEN	LENGTH OF DATA SET NAME
1	(1)	CHARACTER	44	DTLCAT	CATALOG NAME
Note: REST OF DATA SET HEADER RECORD					
0	(0)	STRUCTURE	106	DTDSHDR	
0	(0)	UNSIGNED	1	DTDLEN	LENGTH OF DATA SET NAME
1	(1)	UNSIGNED	1	DTDCATLN	LENGTH OF CATALOG NAME
2	(2)	CHARACTER	2	DTDDSORG	DATA SET ORGA (FROM F1)
4	(4)	CHARACTER	1	DTDOPTCD	DS OPTION CODE (FROM F1)
5	(5)	CHARACTER	1	DTDNVOL	NUMBER OF VOLUMES FOR DATA SET
6	(6)	BITSTRING	1	DTDIND	DATA SET INDICATOR
		1...		DTDIPWD	PASSWORD SUPPLIED 1=YES

& *Table 14. ADRTAPB Mapping Macro (continued)*

& Offsets					
& Dec	Hex	Type	Len	Name (Dim)	Description
&		.1..		DTDTPWD	TYPE PASSWORD 0=D.S. 1=CATALOG
&		..1.		DTDIRACF	RACF PROFILE
&		...1 1...		DTDRACFP	RACF PROFILE FLAGS
&		...1		DTDRACFD	1 = RACF DISCRETE PROFILE
&		.... 1...		DTDRACFG	1 = RACF GENERIC PROFILE
&		.... .1..		DTDALIAS	1 = USER CATALOG ALIAS
&		.... ..1.		DTDSPER	1 = SPHERE RECORD FOLLOWS
&		1		DTDSMS	1=SMS-MANAGED DATA SET
& 7	(7)	CHARACTER	8	DTDPWD	PASSWORD
& 15	(F)	CHARACTER	44	DTDCAT	CATALOG NAME
& 59	(3B)	CHARACTER	44	DTDDSN	DATA SET NAME
& 103	(67)	CHARACTER	2	DTDVOLCT	SMS VOLUME COUNT (FROM BCS)
& 103	(67)	UNSIGNED	1	DTDVCTD	VOLCOUNT FOR DATA COMPONENT OR NONVSAM DATA SET
& 104	(68)	UNSIGNED	1	DTDVCTI	VOLCOUNT FOR INDEX COMPONENT OR ZERO FOR NO INDEX
& 105	(69)	CHARACTER	1	DTDIND2	DATA SET INDICATOR 2
&		1...		DTDAIXSP	AIX AND PART OF A SPHERE
&		.1..		DTDCDF	1=COMMON DATA FORMAT DSET
&		..1.		DTDPDSE	1=PDSE DATA SET
&		...1		DTDNTALL	1=DUMPED WITHOUT USING ALLD OR ALLX
&		.... 1...		DTDSAI	1=DS ADDITIONAL INFORMATION
&		.... .1..		DTDNOIDX	1=VSAM INDEXED DATA SET DUMPED USING VALIDATE OPTION (INDEX NOT DUMPED, DATA CI'S IN ORDER)
&		.... ..1.		DTDPDSET	1=PDSE DUMPED AS TRACK IMAGES
&		1		DTDSDM	USE SYSTEM DATA MOVER.
& **Note:** SPHERE RECORD FOR CATALOG FILTER DUMP					
& 0	(0)	STRUCTURE	4	DTSPHERE	
& 0	(0)	SIGNED	4	DTSLEN	LENGTH OF SPHERE RECORD
& 0	(0)	STRUCTURE	102	DTSINFO	SPHERE INFORMATION
& 0	(0)	CHARACTER	44	DTSAIXNM	AIX NAME
& 44	(2C)	CHARACTER	44	DTSPATHN	PATH NAME
& 88	(58)	CHARACTER	1	DTSPATHA	PATH ATTRIBUTE
& 89	(59)	BITSTRING	1	DTSPATHF	PATH INFO FLAG
&		1...		DTSPATHI	IF SET, PATH OWNERID IS CONTAINED IN THIS BLOCK (SEE DTSPTHON FOR DESCRIPTION)
&		.1..		DTSPATHE	PATH EXPIRATION DATE IS CONTAINED IN THIS BLOCK (SEE DTSPTHEP FOR DESCRIPTION). IF NOT SET, IGNORE THE EXPIR FIELDS.
&		..1.		DTSPATHP	PATH HAS PASSWORD THAT IS CONTAINED IN THIS BLOCK (SEE DTSEXPIR FOR DESCRIPTION). IF EXISTS, BEGINS AT DTSPATHD.
&		...1 1...		DTSPRACF	PATH RACF FLAGS
&		...1		DTSRACFD	1=DISCRETE PROFILE
&		.... 1...		DTSRACFG	1=GENERIC PROFILE
&		.... .111		*	NOT USED
& 90	(5A)	CHARACTER	8	DTSPTHON	FORMAT OF OWNERID
& 98	(62)	CHARACTER	4	DTSPTHEP	EXPIRATION DATE FORMAT:
& 98	(62)	CHARACTER	3	DTSEXPIR	YYDDDF FORMAT
& 98	(62)	UNSIGNED	1	DTSEYEAR	YY 8 BIT IN DECIMAL
& 99	(63)	UNSIGNED	2	DTSERDAY	12-BITS DAY OF IN DECIMAL AND 4-BIT SIGN CHARACTER THAT ALLOWS TO BE UNPACKED
& 101	(65)	UNSIGNED	1	DTSEXCNY	CENTURY BYTE IN DECIMAL
& 0	(0)	STRUCTURE	52	DTSPASSW	SECURITY INFO TABLE

Table 14. ADRTAPB Mapping Macro (continued)

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	CHARACTER	8	DTSMSTRP	MASTER PASSWORD. MUST HAVE IN ORDER TO HAVE ANY OTHER PASSWORDS.
8	(8)	CHARACTER	8	DTSCNTLP	CONTROL INTERVAL PASSWORD
16	(10)	CHARACTER	8	DTSUPDAT	UPDATE PASSWORD
24	(18)	CHARACTER	8	DTSREADP	READ PASSWORD
32	(20)	CHARACTER	8	DTSCODEN	CODE NAME
40	(28)	SIGNED	2	DTSNUMAT	NUMBER OF ATTEMPTS
42	(2A)	CHARACTER	8	DTSAUTNM	ADDRESS OF AUTHORIZATION MOD
50	(32)	SIGNED	2	DTSAUTHR	AUTHORIZATION RECORD LENGTH
Note: COMMON DATA FORMAT DATA SET ATTRIBUTE RECORD					
0	(0)	STRUCTURE	64	DTCDFATT	CDF ATTRIBUTE
0	(0)	SIGNED	4	DTCHURPN	HIGH USED PAGE NUMBER
4	(4)	CHARACTER	1	DTCDFIND	CDF FLAGS
		1...		DTDPDSEX	PDSEX FLAG
5	(5)	CHARACTER	3	*	RESERVED FOR FUTURE
8	(8)	UNSIGNED	4	DTC#DIRB	DIRECTORY BLOCK CNT
12	(C)	CHARACTER	52	*	RESERVED FOR FUTURE
64	(40)	CHARACTER		*	LAST
0	(0)	STRUCTURE	*	DTDSAIR	
0	(0)	SIGNED	4	DTDSAIDL	ADDITIONAL DATA LENGTH
4	(4)	CHARACTER	*	DTDSAID	ADDITIONAL DATA FOLLOWS
Note: REST OF DATA SET VOLUME DEFINITION RECORD - 1/VOLUME					
0	(0)	STRUCTURE	28	DTMVOL	
0	(0)	CHARACTER	6	DTMVSERL	VOLUME SERIAL ID
6	(6)	CHARACTER	4	DTMDEVTY	DEVTYPE (UCBTBYT4)
10	(A)	CHARACTER	2	*	SLACK FILLER
12	(C)	CHARACTER	8	DTMVOLSZ	VOLUME SIZE (DS4DEVSZ)
12	(C)	UNSIGNED	4	DTMTRKCP	NUMBER OF BYTES PER TRACK (TRACK CAPACITY WITH OVERHEAD).
16	(10)	UNSIGNED	2	DTMLOGCY	NUMBER OF CYLINDERS PER VOLUME
18	(12)	UNSIGNED	2	DTMTRKCY	NUMBER OF TRACKS PER VOLUME
20	(14)	UNSIGNED	2	DTMMAXCB	MAXIMUM COMPRESS BUFFER IN WORDS
22	(16)	CHARACTER	2	DTMIND	VOLUME INDICAT
		1...		DTMVIRT	VIRTUAL VOLUME
		.1..		DTMMINI	MINI VOLUME
		..1.		DTMCVAF	VOLUME HAS INDEXED VTOC
		...1		DTMTIME	TIME STAMP FOLLOWS VVR
		.... 1...		DTMBWOT	RLS TIME STAMPS ARE BWO
22	(16)	BITSTRING	1	*	UNUSED
24	(18)	UNSIGNED	1	DTM#VVRS	NUMBER OF VVRS/NVRS DUMPED
25	(19)	UNSIGNED	1	DTM#DSCB	NUMBER OF DSCBS DUMPED
26	(1A)	UNSIGNED	1	DTM#EXT	NUMBER OF EXTENTS DUMPED
27	(1B)	CHARACTER	1	DTMMODNO	MODEL NUMBER
Note: REST OF RECORD THAT MAPS THE TRACKS DUMPED					
0	(0)	STRUCTURE	12	DTBTMR	
0	(0)	CHARACTER	12	DTBHDR	RESERVED
0	(0)	CHARACTER	4	DTBBMID	ID OF THE BLOCK = BMBB
4	(4)	SIGNED	4	DTBADDR	@ OF NEXT BMBB SEGMENT COPIED FROM ADRBMB BLOCK
8	(8)	UNSIGNED	4	DTBTRK#	NUMBER OF TRACKS MAPPED IN BMBB SEGMENT
8	(8)	UNSIGNED	2	DTBTRK#P	NUMBER OF TRACKS MAPPED IN BMBB SEGMENT
10	(A)	CHARACTER	2	*	RESERVED
Note: REST OF TRACK RECORD FIRST SEGMENT OF THE TRACK					

Table 14. ADRTAPB Mapping Macro (continued)

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	24	DTTTRK	MAPS THE TRACK
0	(0)	CHARACTER	16	DTTHDR	HEADER FOR EACH TRACK SEGMENT
0	(0)	UNSIGNED	2	DTTTRKLN	LENGTH OF DATA ON TRACK
2	(2)	CHARACTER	1	DTTTRKID	TRACK INDICATORS
		1...		DTTIOER	I/O ERROR ON TRACK
		.1..		DTTTROVF	LAST RECORD ON TRACK IS OVERFLOW RECORD
		..1.		DTTTCMP	IF ON, TRACK COMPRESSED
		...1		DTTVFRST	FIRST VVDS RECORD
		.... 1...		DTTINVT	INVALID TRACK FORMAT
		.... .1..		DTTSTAT	USER STATISTICAL RECORD
3	(3)	CHARACTER	4	DTTCCHH	CCHH OF TRACK
3	(3)	UNSIGNED	2	DTTCC	CYLINDER NUMBER
5	(5)	UNSIGNED	2	DTTHH	TRACK NUMBER
7	(7)	SIGNED	4	DTTLRCNT	LR COUNT FOR THE DS (THIS FIELD IS FILLED IN ON THE LAST DATA CA TRACKS FOR A VS DUMPED BY VSAM I/O)
11	(B)	SIGNED	1	DTTXDLEN	EXTRA ENCRYPTED DATA LENGTH
12	(C)	CHARACTER	4	*	RESERVED
16	(10)	CHARACTER	8	DTTR0DAT	RCRD 0 DATA, ONLY ON FIRST SEG
24	(18)	CHARACTER		DTTBODY	R1- RN RECORDS ON TRACK
0	(0)	STRUCTURE	8	DTTCKD	CNT, KEY AND DATA FIELDS ON TRACK
0	(0)	CHARACTER	8	DTTCNT	COUNT FIELD
0	(0)	CHARACTER	5	DTTCCHHR	CCHHR OF RECORD
0	(0)	CHARACTER	4	DTTCCCHH	CCHH OF RECORD
0	(0)	UNSIGNED	2	DTTCCC	CC OF RECORD
2	(2)	UNSIGNED	2	DTTCHH	HH OF RECORD
4	(4)	UNSIGNED	1	DTTCRCRD	R OF RECORD
5	(5)	UNSIGNED	1	DTTKELEN	KEY LENGTH
6	(6)	UNSIGNED	2	DTTDATLN	DATA LENGTH

Note: OTHER SEGMENTS OF TRACK: THE REST OF THE TRACK SEGMENTS DO NOT HAVE THE TRACK HEADER AS SPECIFIED IN DTTTRK BUT HAVE THE COUNT, KEY AND DATA FIELDS FOLLOWING THE SEGMENT PREFIX. REST OF RECORD THAT MAPS THE TRAILER RECORD

0	(0)	STRUCTURE	6	DTRTLR	
0	(0)	CHARACTER	6	DTRSERL	VOLUME SERIAL OF FILE (USED FOR PHYSICAL DUMP ONLY) V26
0	(0)	SIGNED	4	DTRRECNT	NUMBER OF LOGICAL RECORDS DUMPED (USED FOR LOGICAL DATA SET DUMP ONLY) V26
4	(4)	CHARACTER	2	*	RESERVED - LOGICAL DUMP

Note: SELF-DESCRIBING RECORD

0	(0)	STRUCTURE	4	DTSDHDR	
0	(0)	SIGNED	2	DTSDLEN	LENGTH OF SELF DESCRIBING REC
2	(2)	UNSIGNED	1	DTSDTYPE	TYPE OF SELF DESCRIBING REC
3	(3)	CHARACTER	1	DTSDIND1	SELF DESCRIGING RECORD FLAGS
		1...		DTSDLAST	LAST SELF DESCRIBING RECORD
		.111 1111		*	RESERVED
4	(4)			DTSDEND	REST OF RECORD

Note: REST OF SELF-DESCRIBING CIPHER BLOCK RECORD

0	(0)	STRUCTURE	352	DTCIPHB	CIPHER BLOCK
0	(0)	CHARACTER	1	DTCFLAG1	ENCRYPTION TYPES
		1...		DTCCTDES	CLRTDES
		.1..		DTCCA128	CLRAES128
		..1.		DTCETDES	ENCTDES
		...1 1111		*	RESERVED
1	(1)	CHARACTER	3	*	RESERVED

& *Table 14. ADRTAPB Mapping Macro (continued)*

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
4	(4)	CHARACTER	4	DTCDATAS	SAMPLE DATA
8	(8)	CHARACTER	64	DTCRSAL	RSA LABEL
72	(48)	CHARACTER	256	DTCRDKL	RSA ENCIPHED DATA LABEL
328	(148)	SIGNED	4	DTCICNT	ITERATION COUNT
332	(14C)	CHARACTER	8	DTCSALT	SALT FOR ADRPWKEY
Note: REST OF SELF-DESCRIBING COMPRESSION DICTIONARY BLOCK RECORD					
0	(0)	STRUCTURE	16	DTCDCT	
0	(0)	STRUCTURE		DTCFLDS	COMPRESSION DICTIONARY FIELDS
0	(0)	SIGNED	4	DTCMAXD	MAX DICTIONARY SIZE
0	(4)	CHARACTER	12	*	RESERVED
Note: EXPANSION DICTIONARY IMMEDIATELY FOLLOWS COMPRESS DICTIONARY BLOCK					
0	(0)	CHARACTER	*	DTCDICT	EXPANSION DICTIONARY

ADRTAPB Constants

Table 15. ADRTAPB Mapping Macro

Length	Type	Value	Name	Description
1	CONSTANT	X'01'	DTSDCIPH	CIPHER BLOCK SELF DESCRIBING RECORD CONSTANT
1	CONSTANT	X'02'	DTSDCDCT	COMPRESSION DICTIONARY SELF DESCRIBING RECORD CONSTANT

ADRTAPB Cross-Reference

Name	Hex Offset	Hex Value
ADRTAPB	0	
BFRPREFX	0	
BPBODY	40	
BPHWMRK	4	
BPSCOC	0	
BPSCOD	2	
BPWORKA	8	
DTBADDR	4	
DTBBMID	0	
DTBHDR	0	
DTBTMR	0	
DTBTRK#	8	
DTBTRK#P	8	
DTC#DIRB	8	
DTCCA128	0	40
DTCDATAS	4	20
DTCCTDES	0	80
DTCDCT	0	
DTCDFATT	0	
DTCDFIND	4	
DTCDICT	4	
DTCETDES	0	
DTCFLAG1	0	

	Name	Hex Offset	Hex Value
&	DTCFLDS	0	
	DTCHURPN	0	
&	DTCICNT	148	
&	DTCIPHB	0	
&	DTCMAXD	0	
&	DTCRDKL	48	
&	DTCRSAL	8	
&	DTCSALT	14C	
	DTDAIXSP	69	80
	DTDALIAS	6	04
	DTDCAT	F	
	DTDCATLN	1	
	DTDCDF	69	40
	DTDDSN	3B	
	DTDDSORG	2	
	DTDIND	6	
	DTDIND2	69	
	DTDIPWD	6	80
	DTDIRACF	6	20
	DTDLEN	0	
	DTDNOIDX	69	04
	DTDNTALL	69	10
	DTDNVOL	5	
	DTDOPTCD	4	
	DTDPDSE	69	20
	DTDPDSET	69	02
	DTDPDSEX	4	80
	DTDPWD	7	
	DTDRACFD	6	10
	DTDRACFG	6	08
	DTDRACFP	6	18
	DTDSAI	69	08
	DTDSAID	4	
	DTDSAIDL	0	
	DTDSAIR	0	
	DTDSDM	69	01
	DTDSHDR	0	
	DTDSMS	6	01
	DTDSPER	6	02
	DTDTPWD	6	40
	DTDVCTD	67	
	DTDVCTI	68	
	DTDVOLCT	67	
	DTHBLKSZ	D	
	DTHDAY	0	
	DTHDR	0	
	DTHFCMP	11	80
	DTHGNVI	8	80
	DTHGT64K	8	20
	DTHGVI	8	40
	DTHIND1	11	
	DTHIND2	8	
	DTHLEN	9	
	DTHLVLNO	C	
	DTHNDS	F	
	DTHSFER	11	20
	DTHTIMD	0	

Name	Hex Offset	Hex Value
DTHTIME	4	
DTHUNLCD	11	40
DTHVERNO	B	
DTLCAT	1	
DTLDSN	0	
DTLLEN	0	
DTM#DSCB	19	
DTM#EXT	1A	
DTM#VVRS	18	
DTMBWOT	16	08
DTMCVAF	16	20
DTMDEVTY	6	
DTMIND	16	
DTMLOGCY	10	
DTMMAXCB	14	
DTMMINI	16	40
DTMMODNO	1B	
DTMTIME	16	10
DTMTRKCP	C	
DTMTRKCY	12	
DTMVIRT	16	80
DTMVOL	0	
DTMVOLSZ	C	
DTMVSERL	0	
DTPBODY	10	
DTPBTM	A	40
DTPBWOE	B	08
DTPDASDD	9	20
DTPDATA	A	08
DTPDDDSP	B	40
DTPDDISP	B	80
DTPDMPID	9	
DTPDUMPC	B	
DTPDSHDR	A	20
DTPDSNL	A	40
DTPDTRLR	A	04
DTPENDKR	B	10
DTPENDNT	B	20
DTPFULD	9	80
DTPLOGCL	9	10
DTPNOSEG	4	
DTPPARTD	9	40
DTPPFXLN	8	
DTPRCFL1	B	
DTPRCID1	A	
DTPSCHK	0	
DTPSEGLN	6	
DTPSEGNO	5	
DTPSEQNO	0	
DTPSPHDR	A	01
DTPTHDR	A	80
DTPTRK0	A	20
DTPVHDR	A	80
DTPVOLD	A	10
DTPVTOC	A	10
DTPVTRLR	A	04
DTPVVDS	A	02

	Name	Hex Offset	Hex Value
	DTRRECNT	0	
	DTRSERL	0	
	DTRTLR	0	
	DTSAIXNM	0	
	DTSAUTHR	32	
	DTSAUTNM	2A	
	DTSCNTLP	8	
	DTSCODEN	20	
&	DTSDCIPH	14C	
&	DTSDHDR	0	
&	DTSDLEN	0	
&	DTSDTYPE	2	
&	DTSDIND1	3	
&	DTSDLAST	3	80
&	DTSDEND	4	
	DTSERDAY	63	
	DTSEXCNY	65	
	DTSEXPIR	62	
	DTSEYEAR	62	
	DTSINFO	0	
	DTSLEN	0	
	DTSMSTRP	0	
	DTSNUMAT	28	
	DTSPASSW	0	
	DTSPATHA	58	
	DTSPATHE	59	40
	DTSPATHF	59	
	DTSPATHI	59	80
	DTSPATHN	2C	
	DTSPATHP	59	20
	DTSPHERE	0	
	DTSPRACF	59	18
	DTSPTHEP	62	
	DTSPTHON	5A	
	DTSRACFD	59	10
	DTSRACFG	59	08
	DTSREADP	18	
	DTSUPDAT	10	
	DTTBODY	18	
	DTTCC	3	
	DTTCCC	0	
	DTTCCCHH	0	
	DTTCCHH	3	
	DTTCCHHR	0	
	DTTCHH	2	
	DTTCKD	0	
	DTTCNT	0	
	DTTCRCRD	4	
	DTTDATLN	6	
	DTTHDR	0	
	DTTHH	5	
	DTTINVT	2	08
	DTTIOER	2	80
	DTTKELEN	5	
	DTTLRCNT	7	
	DTTR0DAT	10	
	DTTSTAT	2	04

Name	Hex Offset	Hex Value
DTTTCMP	2	20
DTTTRK	0	
DTTTRKID	2	
DTTTRKLN	0	
DTTTROVF	2	40
DTTVFRST	2	10
DTVBLKSZ	C	
DTVBMSZ	25	
DTPBTM	A	40
DTVCPVOL	24	10
DTVCVAF	24	20
DTVDAY	17	
DTVDEVTY	1F	
DTVFCMP	24	08
DTVGNVI	27	20
DTVGVI	27	10
DTVIND1	24	
DTVIND2	27	
DTVLEN	28	
DTVLNVI	27	80
DTVLOGCY	8	
DTVLVF	24	02
DTVLVI	27	40
DTVLVLNO	2B	
DTVMAXCB	E	
DTVMINI	24	40
DTVMODNO	23	
DTVOL	0	
DTVOLSZ	8	
DTVSERL	10	
DTVSMS	27	08
DTVSMSI	27	04
DTVTIMD	17	
DTVTIME	1B	
DTVTOCB	0	
DTVTOCE	4	
DTVTRKCY	A	
DTVUNLCD	24	04
DTVVERNO	2A	
DTVVIRT	24	80
DTVVTOCI	16	
DTVVVDS	24	01
DTVVXLEN	0	
DTVVXTNO	2C	
DTVVXTNT	0	
DTVVXTOF	2D	

Appendix D. DFSMSdss Patch Area

This appendix explains how to customize certain DFSMSdss functions by setting flags in ADRPATCH, a module in the DFSMSdss load module (ADRDSSU). You can set the flags in ADRPATCH dynamically by using the DFSMSdss SET PATCH auxiliary command. You can also permanently set the flags in ADRPATCH by using the AMASPZAP program.

The SET PATCH offset=value command specifies that DFSMSdss set the patch byte, at the specified offset to the specified value. Your installation can limit use of the SET PATCH command with the RACF FACILITY-class profile STGADMIN.ADR.PATCH. You must have READ access authorization to that profile to use the SET PATCH command.

The following sample JCL sets the flags in ADRPATCH. The specific offsets and values of the flags are explained in the following sections. To see the mapping of the flags in ADRPATCH, go to the ADRPTCHB data area description inTable 16 on page 214.

SAMPLE JCL:

```
//PATCH    JOB...
//*
//*****************************************************************
//*                                                               *
//* SAMPLE JCL TO SET THE FLAGS IN ADRPATCH.                      *
//*                                                               *
//*****************************************************************
//ZAP      EXEC PGM=AMASPZAP,PARM='IGNIDRFULL'
//SYSPRINT DD   SYSOUT=*
//SYSLIB   DD   DISP=SHR,DSN=LIBNAME.LINKLIB
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   offset    value
   REP   offset    value
/*
```

As an alternative to using the above JCL to set flags in ADRPATCH, you can customize certain DFSMSdss functions by temporarily setting patch bytes during DFSMSdss processing through a SET PATCH command. Refer to *z/OS DFSMSdss Storage Administration Reference* for details on the SET command.

Forcing the Use of Preallocated VSAM Data Sets (PN04574)

You can force the use of preallocated VSAM data sets by setting the flag at offset X'08' in ADRPATCH. The settings are:

X'00'	DFSMSdss functions normally, deleting and reallocating VSAM data sets, as necessary, before restoring data to them.
Any setting other than X'00'	During a logical RESTORE of VSAM data sets to preallocated targets, DFSMSdss unconditionally uses the preallocated data sets without deleting and reallocating them. DFSMSdss also assumes the targets are reusable and resets the high use RBA to

0 during OPEN if the target data sets are not empty. You can ensure that the preallocated targets are defined using the REUSE attribute.

Note: If you want DFSMSdss to restore a single-volume VSAM data set to multiple volumes, the preallocated target data set must have the reusable attribute and have data allocated across multiple primary volumes. DFSMSdss will not recognize candidate volumes.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on (for example, X'FF') permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   08        00
   REP   08        FF
```

Ignoring VSAM Duplicate Key Errors (PN05529)

If you set the flag at offset X'09' in ADRPATCH on (any value other than X'00'), DFSMSdss provides a serviceability aid that determines which records have duplicate keys. DFSMSdss restores all records of a keyed VSAM data set, ignoring any duplicate key error conditions that have occurred.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on (for example, X'FF') permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   09        00
   REP   09        FF
```

You can set a slip trap and use generalized trace facility (GTF) as follows:

1. Use AMBLIST to locate the displacement (xxxx) of label ADRSLIP1 within ADRDSSU. The displacement of label ADRSLIP1 is used in setting the slip trap.

```
//AMBLIST  JOB...
//*
//*************************************************************
//*                                                           *
//* SAMPLE JCL TO LOCATE THE DISPLACEMENT OF LABEL ADRSLIP1   *
//* WITHIN ADRDSSU.  THE DISPLACEMENT OF ADRSLIP1 IS NEEDED   *
//* TO SET THE SLIP TRAP.                                     *
//*                                                           *
//*************************************************************
//LISTIT   EXEC PGM=AMBLIST
//SYSPRINT DD   SYSOUT*
//SYSLIB   DD   DISP=SHR,DSN=LIBNAME.LINKLIB
//SYSIN DD *
/*
```

2. Set a slip trap from the system console. The displacement label of ADRSLIP1 obtained with AMBLIST is 'xxxx'. Registers 7 and 8 must be used as shown below:

   ```
   SLIP SET,IF,ACTION=TRACE,TRDATA=(STD,7R?,8R?),
   PVTMOD=(ADRDSSU,xxxx),JOBNAME=nnnnnnnn,END
   ```

3. Start the GTF trace at the system console:
   a. Enter **S GTF**.

b. Respond **TRACE=SLIP,USR** to message AHL125A (“SPECIFY TRACE OPTIONS”).
c. Respond **U** to message AHL125A (“RESPECIFY TRACE OPTIONS OR REPLY U”).
d. Run the DFSMSdss job.
e. Use the Interactive Problem Control System (IPCS) to analyze the GTF trace output to determine which records are in error.

Notes:

1. The duplicate key error occurs only after the first record with a duplicate key gets written to the data set. Second and subsequent records with the same key are recognized as duplicates.
2. Only the first 65 535 bytes of each duplicate record can be written to the trace data set.

Modifying the Timeout Period for Enqueue Lockout Detection (PL84514)

The timeout period for enqueue lockout detection is 90 seconds. You can customize the timeout period by setting the halfword at offset X'0A' in ADRPATCH to the number of seconds to wait. Setting the field to X'FFFF' disables enqueue lockout detection.

To set the timeout period dynamically, use the SET PATCH command. To set the timeout period permanently, (for example, 120 seconds), modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   0A      0000
   REP   0A      0078
```

To disable lockout detection, use the SET PATCH command or modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   0A      0000
   REP   0A      FFFF
```

Controlling the Wait/Retry Time for Serialization of System Resources (PN11523)

For system resources (such as the VTOC or VVDS), the default wait time is 3 seconds and the default retry count is 30. Therefore, the total default wait time is 90 seconds. To modify these defaults, you must set the byte at each of the following offsets:

X'0D' An indicator that new wait/retry values are specified. Any nonzero value is valid.

X'0E' The new wait time in seconds. Valid values are X'00' through X'FF' (0 - 255).

X'0F' The new retry count. Valid values are X'00' through X'FF' (0 - 255).

To have DFSMSdss use a wait time of 60 seconds and a retry count of 10 (for a total wait time of ten minutes), do one of the following:

- Use the SET PATCH command to set the wait/retry time dynamically

- Modify the sample JCL on page 193 as follows:

```
//SYSIN    DD *
   NAME ADRDSSU ADRPATCH
   VER    000D    000000      /* Verify all three bytes are zero */
   REP    000D    FF          /* Set indicator flag to nonzero   */
   REP    000E    3C          /* Set wait time to 60 seconds     */
   REP    000F    0A          /* Set retry count to 10           */
```

Using CONVERTV on Data Sets with a Revoked User ID in the RESOWNER Field (OY59957)

DFSMSdss lets you perform CONVERTV operations on data sets that have a revoked user ID, in the RESOWNER field in the profile. The byte at offset X'11' in ADRPATCH can be used to give that resource owner authority to the SMS constructs.

To use ADRPATCH to set the revoked RESOWNER on, set the byte at offset X'11' in module ADRPATCH to a X'FF'. You can use the SET PATCH command to set the patch byte dynamically or modify the sample JCL on page 193 as follows to set the patch byte permanently:

```
//SYSIN    DD *
   NAME ADRDSSU ADRPATCH
   VER   11      00
   REP   11      FF
```

Notes:

1. Catalog APAR OY56724 must be applied for this fix to work for VSAM data sets.
2. Normal DFSMSdss COPY and RESTORE processing is not changed to bypass MGMTCLAS and STORCLAS authorization checking. See “Bypassing Storage and Management Class Authorization Checking during RESTORE (OY65348)” on page 198.

Restoring Inconsistent PDSE Data Sets (OY60301)

If during a logical restore operation, DFSMSdss cannot tell if the data set is a PDS or a PDSE, it issues message ADR793E, indicating that the data set is an inconsistent PDSE. Setting the flag at offset X'12' in ADRPATCH tells DFSMSdss how to process these data sets during the restore. The settings are listed below:

X'01' DFSMSdss tries to restore the data set as a PDSE. Before the attempt, DFSMSdss issues message ADR794W to warn you that DFSMSdss assumes that the data set to be restored is a PDSE.

Notes:

1. If the data set is a PDS and you try to restore it as a PDSE, it will be unusable.
2. If you try to restore the data set to an unlike device, the restore fails and DFSMSdss issues message ADR792E, indicating to which device type the data set must be restored.
3. If you try to restore the data set to a preallocated target data set, the restore fails because DFSMSdss does not recognize the preallocated data set as usable.

X'02' DFSMSdss tries to restore the data set as a PDS. Before the attempt, DFSMSdss issues message ADR794W to warn you that DFSMSdss assumes that the data set to be restored is a PDS.

The data set is restored only if it is a PDS.

Any setting other than X'01' or X'02'
DFSMSdss does not restore the data set and issues message ADR793E.

To set the patch byte value dynamically, use the SET PATCH command. To set the patch byte value permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   12        00
   REP   12        01
```

Changing Default Protection Status during RESTORE (PN37489)

By default, DFSMSdss maintains the protection status of the data set during a logical RESTORE. If the source data set was protected by RACF®, a component of the Security Server for z/OS, at dump time, the target data set is RACF-indicated at restore time.

This function is affected by setting the flag at offset X'13' in ADRPATCH. The settings are listed below:

X'00'	DFSMSdss functions as previously described.
Any setting other than X'00'	DFSMSdss does not RACF-indicate the target data set during a logical RESTORE, even if the source data set was RACF-indicated at dump time, the MENTITY keyword was not specified, and the target data set was protected by a generic profile.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   13        00
   REP   13        FF
```

Restoring or Copying Undefined, Multivolume SMS-Managed Data Sets (OY63818)

Messages ADR709E and IGD17040I might be received when copying or restoring the following types of non-VSAM, multivolume data sets:

- Empty, multivolume data sets dumped with the ALLEXCP keyword
- Undefined data sets
- TTR-organized BDAM data sets
- Data sets with BLKSIZE=0

This function is affected by setting the flag at offset X'14' in ADRPATCH. The settings are listed below:

X'00'	DFSMSdss functions normally; these data sets are allocated as multivolume data sets.
Any setting other than X'00'	These data sets are allocated as single volume data sets.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   14        00
   REP   14        FF
```

Bypassing Backup-While-Open Processing (OY63531)

When DFSMSdss encounters a data set marked as backup-while-open (BWO) eligible, DFSMSdss processes it accordingly. If the BWO indication is on erroneously, the BWO processing may be bypassed for a logical data set dump.

This function is affected by setting the flag at offset X'15' in ADRPATCH. The settings are listed below:

X'00'	BWO processing is performed.
Any setting other than X'00'	BWO processing is bypassed.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   15        00
   REP   15        FF
```

Bypassing Storage and Management Class Authorization Checking during RESTORE (OY65348)

For a logical RESTORE and a physical data set restore, storage class and management class authorization checking is normally conducted. This checking can be bypassed to let the storage administrator restore data sets that are protected by data set profiles whose RESOWNER field contains a user ID that has been revoked.

This function is affected by setting the flag at offset X'16' in ADRPATCH. The settings are listed below:

X'00'	DFSMSdss functions normally and conducts authorization checking.
Any setting other than X'00'	Authorization checking is bypassed.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   16        00
   REP   16        FF
```

Issuing Notification for Tape and Migrated Data Sets (OY66092)

On a logical data set copy or dump, a data set can be specified with a partially qualified name. If that name resolves to a data set cataloged to tape or to a migrated data set, DFSMSdss does not issue a warning message that the data set was not selected.

This function is affected by setting the flag at offset X'17' in ADRPATCH. The settings are listed below:

X'00'	DFSMSdss functions normally and issues no warning message.
Any setting other than X'00'	DFSMSdss issues a warning message.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   17        00
   REP   17        FF
```

Using RESET with Concurrent Copy (OY65555)

The RESET keyword specifies that DFSMSdss resets the data-set-changed indicator for data sets after the dump is complete. By default, the RESET keyword is ignored when the CONCURRENT keyword is also specified with the DUMP command.

Setting the byte at offset X'18' in ADRPATCH to a nonzero value causes DFSMSdss to perform RESET processing when both the RESET and CONCURRENT keywords are specified with the DUMP command. In this case, the data-set-changed indicator is reset after concurrent copy initialization is complete and before the data set is written to the output.

To dynamically enable RESET processing with concurrent copy, use the SET PATCH command. To enable RESET processing with concurrent copy permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN    DD *
   NAME ADRDSSU ADRPATCH
   VER   18      00
   REP   18      FF
```

After the data-set-changed indicator has been reset, if the concurrent copy dump is unsuccessful for any reason, the data-set-changed indicator for the data set remains reset even though you might not have a usable dump of the data. For this reason, you must carefully evaluate the risks of using this patch to enable RESET processing with concurrent copy. Rather than using this patch to accomplish incremental backup in DFSMSdss with concurrent copy, use DFSMShsm instead.

In the event that the data-set-changed indicators have been reset and the dump subsequently fails, your recovery action can be either of the following:

- Turn the data-set-changed indicators back on by using AMASPZAP.
- Dump the data again. Because the data-set-changed indicators are still off, you must not specify “BY(DSCHA,EQ,YES)” in the dump command.

Forcing RESTORE after Message ADR482E (OY67532)

If message ADR727E was received while DFSMSdss was dumping an SMS-managed user catalog with aliases, the dump tape might be unusable and message ADR482E might be received during RESTORE. If you set the flag at offset X'19' in ADRPATCH on, DFSMSdss forces the restore to continue after receiving message ADR482E.

Note: If you use the AMASPZAP program to set the patch permanently, turn off this patch after restoring the problem dump tape, so that legitimate message ADR482E condition are detected.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   19        00
   REP   19        FF
```

Restoring VSAM KSDS or VRRDS after Messages ADR789W, ADR364W, and ADR417W (OY67942)

Prior to the fix of APAR OY67724, DFSMSdss sometimes (when using the CONCURRENT and VALIDATE options) created dumps with empty track records for imbedded sequence set tracks and with zero in the total record count dumped for the data set. The sequence set records are not necessary when restoring data sets dumped with VALIDATE. Without patch byte X'1A' on the following messages are generated:

- ADR364W and ADR417W (because of the empty tracks)
- ADR789W (because of the missing record count)

and the restore fails.

If message ADR789W, or message ADR364W and message ADR417W, or all three messages are received during a logical restore of a KSDS or VRRDS from a dump created while using CONCURRENT and VALIDATE either by keyword or by default, you can retrieve your data sets by setting the flag at offset X'1A' in ADRPATCH and rerunning the restore job. The settings are listed below:

X'00' DFSMSdss functions normally. It fails to restore data sets that have errors in the dump.

Any setting other than X'00' During a logical RESTORE of a KSDS or VRRDS, if DFSMSdss detects an empty record in the dump or a missing count of the total number of records dumped for the data set, or both, it restores all of the data for the data set in the dump and keeps the resulting data set. You still get message ADR789W with a valid output record count. You can compare this valid output record count to the record count from message ADR788I from the dump job to ensure DFSMSdss has retrieved all the data.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   1A        00
   REP   1A        FF
```

Restoring VSAM Data Sets with Expiration Date of 1999365 (OW00780)

Prior to the fix of APAR OW00780, DFSMSdss sometimes mishandled VSAM expiration dates during logical data set DUMP and RESTORE operations. Non-SMS VSAM data sets that had an expiration date of 1999365 were dumped and restored with no expiration date (an expiration date of 0000000).

The byte at offset X'1B' of module ADRPATCH has been defined to allow special processing for VSAM data sets with an expiration date of 1999365 that were dumped without the fix for OW00780. The possible settings for the flag byte are listed below:

X'00' Non-SMS VSAM data sets that had an expiration date of 1999365 and that were dumped without the fix for OW00780 are restored with no expiration date.

X'FF' Any non-SMS VSAM data set that had an expiration date of 1999365 and that was dumped without the fix for OW00780 are restored with an expiration date of 1999365. However, non-SMS VSAM data sets that originally had no expiration date are also restored with an expiration date of 1999365.

Note: This flag byte should only be used when restoring VSAM data sets dumped without the APAR OW00780 fix and with an expiration date of 1999365.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   1B        00
   REP   1B        FF
```

Restoring VSAM Data Sets with Expiration Dates beyond 2000 (OW00780)

Prior to the fix of APAR OW00780, DFSMSdss sometimes mishandled VSAM expiration dates during logical data set DUMP and RESTORE operations. Data sets that had expiration dates in the year 2000 or beyond were dumped and restored as though they had expired in the 1900s.

The byte of offset X'1C' of module ADRPATCH has been defined to allow special processing for VSAM data sets with expiration dates in the year 2000 or beyond that were dumped without the fix of OW00780. The possible settings for the flag byte are listed below:

X'00' Any VSAM data sets that expired in the years 20*nn* or 21*nn* and that were dumped without the fix for OW00780 are restored with an expiration date of 19*nn*.

X'FF' Any VSAM data set that was dumped with an expiration date less than 1980 will have 100 years added to the expiration date. This includes not only data sets that were dumped without the fix for OW00780 that had expiration dates for 2000 to 2080 or from 2100 to 2180, but also any VSAM data set that actually had an expiration date less than 1980.

Note: This patch does not yield the correct expiration dates for data sets that expire beyond the year 2100. However, it at least causes the data sets to be restored with an expiration date that has not already passed. Users then have several years to correct the date.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   1C        00
   REP   1C        FF
```

Changing Default Insertion of EOF Track during COPY with ALLDATA Specified (OW15003)

When multivolume sequential data sets are copied to a single-volume like device and ALLDATA is specified, the default action inserts an explicit EOF track in the target data set, when required. A patch is provided that allows an installation to change this default. If the patch byte at offset X'1D' of module ADRPATCH is set to X'FF', no explicit EOF track is added to the target data set.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   1D        00
   REP   1D        FF
```

Using RESET or UNCATALOG in a Logical Data Set Dump (PN60114)

For a logical data set dump operation, use of the RESET or UNCATALOG keyword causes the enqueue on a data set to be held until all data sets are dumped. DFSMSdss does not reset the data-set-changed indicator or uncatalog the data set until after all data sets are dumped.

This function is affected by setting the flag at offset X'1E' in ADRPATCH. The settings are listed below:

X'00'	DFSMSdss functions normally as described above.
Any setting other than X'00'	DFSMSdss resets the data-set-changed indicator or uncatalog the data set when the data set is dumped. The enqueue on a data set is released after DFSMSdss has completed resetting the data-set-changed indicator or uncataloging the data set.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   1E        00
   REP   1E        FF
```

Changing Secondary Allocation Quantity in Format 1 DSCB for PDSE Data Sets (OW07755)

PTFs UW06768 and UW06769 for APAR OW04199 caused the target PDSE secondary allocation quantity (DS1SCAL3) in the Format 1 DSCB to increase or decrease by a ratio of “blocks per track of blocksize” divided by “blocks per track of 4096” during data set COPY and RESTORE processing. As part of the fix to this problem, three pairs of 4-byte fields at offset X'20' through X'37' in ADRPATCH are provided for installations to change the incorrect secondary allocation quantity of

an unpreallocated target PDSE during COPY and RESTORE processing. The secondary allocation quantity of the source PDSE is not affected by the patch.

Note: Use this patch to change the incorrect secondary allocation quantity resulting from PTFs UW06768 and UW06769. Set the high threshold value with caution to avoid changing the secondary allocation quantity of PDSE data sets that are not affected by PTFs UW06768 and UW06769.

To use this function when the source PDSE is allocated in cylinders, you must modify the 4-byte field at each of the following offsets:

X'20' A nonzero, 4-byte value indicates a threshold in cylinders. DFSMSdss compares the secondary allocation quantity of the source PDSE to this threshold. If the secondary allocation quantity is greater than this threshold, the value set at offset X'24' is used as the secondary allocation quantity in the calculation for the target PDSE.

A zero, 4-byte value indicates this function is not activated.

X'24' This value is used as the secondary allocation quantity in cylinders, if applicable.

To use this function when the source PDSE is allocated in tracks, you must modify the 4-byte field at each of the following offsets:

X'28' A nonzero, 4-byte value indicates a threshold in tracks. DFSMSdss compares the secondary allocation quantity of the source PDSE to this threshold. If the secondary allocation quantity is greater than this threshold, the value set at offset X'2C' is used as the secondary allocation quantity in the calculation for the target PDSE.

A zero, 4-byte value indicates this function is not activated.

X'2C' This value is used as the secondary allocation quantity in tracks, if applicable.

To use this function when the source PDSE is allocated in blocks, you must modify the 4-byte field at each of the following offsets:

X'30' A nonzero, 4-byte value indicates a threshold in blocks. DFSMSdss compares the secondary allocation quantity of the source PDSE to this threshold. If the secondary allocation quantity is greater than this threshold, the value set at offset X'34' is used as the secondary allocation quantity in the calculation for the target PDSE.

A zero, 4-byte value indicates this function is not activated.

X'34' This value is used as the secondary allocation quantity in blocks, if applicable.

You can specify one, two, or all three pairs of values. For example, you should make the modifications shown in the JCL example below to the sample JCL on page 193 to cause DFSMSdss to do the following:

- Use 250 cylinders as the secondary allocation quantity when the source PDSE is allocated in cylinders and has a secondary allocation quantity greater than 5000 cylinders.
- Use 2500 tracks as the secondary allocation quantity when the source PDSE is allocated in tracks and has a secondary allocation quantity greater than 50 000 tracks.

- Use 12 500 blocks as the secondary allocation quantity when the source PDSE is allocated in blocks and has a secondary allocation quantity greater than 250 000 blocks.

```
//SYSIN   DD  *
   NAME ADRDSSU ADRPATCH
   VER   20       00000000       /* Verify value is 0      */
   REP   20       00001388       /*   If > 5000  cylinders */
   VER   24       00000000       /*   Verify value is 0    */
   REP   24       000000FA       /*   Then use 250 cyls    */
   VER   28       00000000       /* Verify value is 0      */
   REP   28       0000C350       /*   If > 50000 tracks    */
   VER   2C       00000000       /*   Verify value is 0    */
   REP   2C       000009C4       /*   Then use 2500 tracks */
   VER   30       00000000       /* Verify value is 0      */
   REP   30       0003D090       /*   If > 250000 blocks   */
   VER   34       00000000       /*   Verify value is 0    */
   REP   34       000030D4       /*   Then use 12500 blks  */
```

NOTE: As an alternative, you can use the SET PATCH command to set the patch bytes dynamically

Changing Reference Date Default Settings during Data Set COPY and RESTORE Processing (OW12011)

During RESTORE and COPY operations, source and target data set reference dates remain the same, unless the data set is renamed. However, if you rename the restored or copied data set, the current date (TODAY) replaces the reference date in the target data set's DS1REFD field.

DFSMSdss provides four patch bytes that allow you to change the method of setting the DS1REFD field. Below are the patch bytes:

- If the patch byte at offset X'38' of module ADRPATCH is set to X'FF', the reference date of the target data set that is restored without a rename is set to the current date.
- If the patch byte at offset X'39' of module ADRPATCH is set to X'FF', the reference date of the target data set that is restored and renamed is set to the source data set's reference date.
- If the patch byte at offset X'3A' of module ADRPATCH is set to X'FF', the reference date of the target data set that is copied without a rename is set to the current date.
- If the patch byte at offset X'3B' of module ADRPATCH is set to X'FF', the reference date of the target data set that is copied and renamed is set to the source data set's reference date.

To cause DFSMSdss to set the reference date to the current date for all data sets being copied and restored and not renamed, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   38       00
   REP   38       FF
   VER   3A       00
   REP   3A       FF
```

NOTE: As an alternative, you can use the SET PATCH command to set the patch bytes dynamically

Changing Default Protection Processing during COPY (OW10314)

By default, DFSMSdss issues message ADR757E during COPY with DELETE specified if the data set is RACF-indicated, but a discrete profile is not associated with the data set.

This function is affected by setting the flag at offset X'3C' in ADRPATCH. The settings are listed below:

X'00' DFSMSdss functions normally as described above.

Any setting other than X'00' DFSMSdss does not fail the COPY operation with message ADR757E, but allows the data set to be processed and issues either message ADR759W or ADR771W, depending on the SAF return codes.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   3C        00
   REP   3C        FF
```

Bypassing Management and Storage Class Access Checks during COPY (PN72592)

A COPY operation might not be successful when the owner of the data set does not have read access to the STORCLAS or MGMTCLAS routines that allocate the target data set, even though the ADMINISTRATOR keyword is specified. The ADMINISTRATOR keyword gives access to the data set, not to the automatic class selection (ACS) routines that are used for STORCLAS or MGMTCLAS processing.

This function is affected by setting the flag at offset X'3D' in ADRPATCH. The settings are listed below:

X'00' DFSMSdss functions normally as described above.

Any setting other than X'00' When the ADMINISTRATOR keyword is specified, DFSMSdss causes SMS to bypass access authorization checking to the ACS routines for STORCLAS and MGMTCLAS when allocating the target data set during a COPY operation.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   3D        00
   REP   3D        FF
```

Changing Default Handling of Invalid Tracks Created during Data Set COPY and RESTORE Processing (OW08174)

When invalid tracks are created during COPY and RESTORE processing, message ADR367E is issued and the invalid tracks are erased from the target volume, regardless of whether CANCELERROR is specified. The copy or restore of the data set ends and the target data set is deleted. The COPY or RESTORE operation continues with the next data set.

A patch byte is provided that allows you to change the default handling of invalid tracks created during COPY and RESTORE processing. If the patch byte at offset X'3E' of module ADRPATCH is set to X'FF', CANCELERROR specification will determine the action taken. When CANCELERROR is specified, the action is to issue message ADR367E and erase the invalid track from the target volume. The restore or copy of the data set receiving the error is terminated and the target data set is deleted. The RESTORE or COPY processing continues with the next data set. When CANCELERROR is not specified, the action is to issue message ADR366W and leave the invalid track on the volume. The RESTORE or COPY operation continues.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   3E        00
   REP   3E        FF
```

Forcing RESTORE to the Same Volumes as the Source VSAM Data Set (OW07077)

If a user does not specify output volumes, you can force the RESTORE operation to attempt to return a VSAM data set assigned to an SMS storage class with guaranteed space to the same set of primary volumes that the source data set had occupied when dumped.

This function is affected by setting the flag at offset X'3F' in ADRPATCH. The settings are listed below:

X'00' Unless the user specifies output volumes, RESTORE processing does not pass the source volume list to SMS, and the restored data set probably does not occupy the same primary volumes that the source data set occupied when it was dumped. Because this allows the use of any volumes in the storage group, there is less chance of a failure due to lack of space on volumes.

Any setting other than X'00' Unless the user specifies output volumes, RESTORE processing passes the source volume list to SMS, forcing the restored data set to occupy the same primary volumes that the source data set had occupied when it was dumped, provided that it is assigned to an SMS storage class with guaranteed space. If it will not fit, RESTORE fails processing.

Notes:

1. DFSMSdss cannot ensure that the order of volumes are maintained during a restore of a multivolume data set. SMS determines volume order.
2. This patch does not support data sets with candidate space volumes. The restore of such a data set will fail with an ADR709E message.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   3F        00
   REP   3F        FF
```

Modifying Number of Volumes Allocated for SMS Data Sets during Logical RESTORE and COPY (OW15880)

When processing SMS data sets, the total number of volumes DFSMSdss allocates for the target data set is the same as the total number of volumes that were allocated for the source data set, unless the source data set was multivolume and output volumes are specified. When the source data set was multivolume and a list of output volumes is supplied with OUTDDNAME or OUTDYNAM, the number of volumes DFSMSdss allocates is the number of volumes in the volume output list.

The number of output volumes allocated for SMS data sets can be modified by setting an enabling flag at offset X'40' in ADRPATCH and a count value at offset X'41' in ADRPATCH. After the enabling flag is set, the following are true:

- When an output volume list is specified and the patch count value is not zero, the allocated number of volumes is the lesser of either the number of volumes in the output list or the patch count value.
- When an output volume list is specified and the patch count value is zero, the allocated number of volumes is the lesser of either the number of volumes in the output list, or the allocated number of volumes that were for the source data set.
- When no output volume list is specified, the allocated number of volumes is the greater of either the number of volumes that were allocated for the source data set, or the patch count value.

The patch bytes are defined as follows:

- Offset X'40'

  X'00' Use current DFSMSdss SMS allocation rules

  X'20' Modify DFSMSdss SMS allocation rules

- Offset X'41'

  The hexadecimal representation of the number of volumes to be used for SMS data set allocation. Any value between 0 and 59 (X'00' through X'3B') can be specified. If a value larger than 59 is specified, the value of 59 (X'3B') is used.

Notes:

1. For any keyed VSAM data set, there must be sufficient space for a primary extent on each volume that is to contain data. No space is allocated on candidate volumes.
2. For any guaranteed space keyed VSAM data set, there also must be sufficient volumes in the target storage group to provide volume serial numbers for each primary and candidate volume.
3. This patch does not modify the number of volumes allocated for data sets whose DSORG is PO (for example, PDS, PDSE and HFS), and does not modify the number of volumes allocated for VSAM linear data sets.
4. When this patch is activated, the MAKEMULTI keyword is ignored.

To cause DFSMSdss to use ten volumes for SMS data set allocation, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   40        00
   REP   40        20
   VER   41        00
   REP   41        0A
```

NOTE: As an alternative, you can use the SET PATCH command to set the patch bytes dynamically

Dumping a Keyed VSAM Data Set that has Data CAs without Corresponding Index CIs (OW17877)

When a KSDS is logically dumped with DFSMSdss using the VALIDATE option, a check is performed to determine if there are data control areas (CAs)) without corresponding index control intervals (CIs). If there are missing index CIs, ADR970E is issued and the dump of this data set fails.

This function is affected by setting the flag at offset X'42' in ADRPATCH. The settings are listed below:

X'00' DFSMSdss functions normally as described above.

X'01' DFSMSdss issues ADR985W instead of ADR970E when a possible missing index CI condition is detected during logical data set DUMP using the VALIDATE option.

Any setting other thanX'00' or X'01'
DFSMSdss issues ADR974I instead of ADR970E when a possible missing index CI condition is detected during logical data set DUMP using the VALIDATE option.

Note: This patch byte should only be used when you know that the KSDS being dumped has incomplete CA spits. It should be used with caution since if the data set is actually broken, the backup copy could be incomplete. Also, use caution when specifying the DELETE keyword in conjunction with setting this patch byte. If this patch byte is used to allow a KSDS with more data CAs than index CIs to be successfully dumped, the source KSDS will be deleted if the DELETE keyword is also specified.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   42        00
   REP   42        FF
```

Changing the Default DEFRAG Processing of Checkpointed Data Sets (OW20285)

Note: This patch is not supported by DFSMS/MVS DFSMSdss V1R4 (R1D0) , where IMS™ GSAM checkpointed data sets can be processed by DEFRAG. MVS™ checkpoint/restart checkpointed data sets cannot be processed by DEFRAG unless the parameter FORCECP(days) with an appropriate days value is specified on the DEFRAG command.

When DEFRAG selects an extent for movement and the associated data set's FORMAT 1 DSCB in the VTOC has the DS1CPOIT flag set indicating that a checkpoint was taken while the data set was open, DEFRAG cannot relocate the

data set extent. Message ADR211I is issued to indicate when this occurs. A patch byte is provided to allow an installation to change the default DEFRAG processing of checkpointed data set extents.

This function is affected by setting the flag at offset X'43' in ADRPATCH. The settings are listed below:

X'00'	DFSMSdss functions normally as described above.
Any setting other than X'00'	DEFRAG moves selected extents, even when the data set DS1CPOIT flag is set on. Message ADR252I is issued when the patch byte is set on to indicate that the installation is overriding normal DEFRAG processing.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   43        00
   REP   43        FF
```

Setting the Percentage to Overallocate Target Data Set Space (OW27837)

When restoring an extended-format non-VSAM data set to a device which does not support extended-format, the data set is converted to a nonextended-format. During this type of conversion, DFSMSdss is not able to calculate the exact amount of space that the target data set will require. Therefore, it is possible for the target data set to be underallocated, causing RESTORE to fail with message ADR910E, return code 40000004, reason code 0000000D. To avoid underallocating the target data set, a new patch byte has been created. The hex value set in this patch byte is the percentage to overallocate by. This function is affected by setting the flag at offset X'44' in ADRPATCH.

To use ADRPATCH to overallocate the target data set by a decimal 10%, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   44        00
   REP   44        0A
```

Notes:

1. As an alternative, you can use the SET PATCH command to set the patch byte dynamically.
2. Set this patch only for those data sets that have failed to convert. Reset the patch to zero after the failing data set is restored, so that other converted data sets are not needlessly overallocated. Also, the required percentage of overallocated space can vary by data set. In other words, some data sets might require an overallocation of 10%, while some require additional space. Adjust the overallocation percentage, as needed.

Bypassing RLS Processing (OW32817)

For SMS-managed VSAM data sets, DFSMSdss performs serialization that provides data integrity during logical dump and copy in both the RLS and the non-RLS environments. This serialization includes communications with VSAM RLS to determine the type of enqueues to be performed, based on the type of access that the data set is currently being used for by other jobs.

You can bypass the communication between DFSMSdss and VSAM RLS, which results in DFSMSdss performing non-RLS serialization, regardless of how the data set is currently being accessed. Use of this function can compromise data integrity when the data set is being accessed through RLS by some other job while the dump or copy is being performed. Use this function only if you are willing to tolerate the exposure of not having data integrity in order to force the successful completion of the operation or to prevent updates to the data set from failing during the operation.

When the communication between DFSMSdss and VSAM RLS is bypassed, communication with CICS is also bypassed. Therefore, use this function should only in an environment where forward recovery logging and forward recovery is managed entirely by the application.

This function is affected by setting the flag at offset X'45' in ADRPATCH. The settings are listed below:

X'00'	DFSMSdss functions normally. The communications with VSAM RLS are performed and DFSMSdss provides RLS or non-RLS serialization accordingly.
Any setting other than X'00'	DFSMSdss bypasses RLS processing. The communications with VSAM RLS are not performed and DFSMSdss performs non-RLS serialization.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   45        00
   REP   45        FF
```

Changing Creation Date Default Settings during Logical Data Set COPY and RESTORE (OW19618)

During RESTORE and COPY operations, source and target data set creation dates remain the same, unless the data set is renamed. However, if you rename the restored or copied data set, the current date (TODAY) replaces the creation date in the target data set's DS1CREDT field.

DFSMSdss provides four patch bytes that allow you to change the method of setting the DS1CREDT field. These patch bytes apply only to data sets that are allocated by DFSMSdss (not prealocated data sets). The patch bytes are below:

- If the patch byte at offset X'46' of module ADRPATCH is set to X'FF', the creation date of the target data set that is restored without a rename is set to the current date.

- If the patch byte at offset X'47' of module ADRPATCH is set to X'FF', the creation date of the target data set that is restored and renamed is set to the source data set's creation date.
- If the patch byte at offset X'48' of module ADRPATCH is set to X'FF', the creation date of the target data set that is copied without a rename is set to the current date.
- If the patch byte at offset X'49' of module ADRPATCH is set to X'FF', the creation date of the target data set that is copied and renamed is set to the source data set's creation date.

To cause DFSMSdss to set the creation date to the current date for all data sets being copied and restored and not renamed, modify the sample JCL on page 193 as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER   46        00
   REP   46        FF
   VER   48        00
   REP   48        FF
```

NOTE: As an alternative, you can use the SET PATCH command to set the patch bytes dynamically

Copying and Dumping a PDSE Data Set using the VALIDATE PDSE Option (OW48074)

DFSMSdss sometimes invokes the file and attribute management services (FAMS) to process PDSE data sets in logical data set COPY and DUMP operations. By default, DFSMSdss does not enable the FAMS validation option because validation can slow the processing time. However, a DFSMSdss logical COPY or logical DUMP operation with the FAMS validation disabled might not detect a potentially broken PDSE. It might copy or dump the invalid PDSE with return code zero—without processing all members.

To change the VALIDATE PDSE option, set the flag at offset X'4B' in ADRPATCH. Use the following settings:

X'00'	DFSMSdss functions without using the FAMS VALIDATE PDSE option.
Any setting other than X'00'	DFSMSdss functions using the FAMS VALIDATE PDSE option.

Note: This patch byte is only valid for logical COPY and DUMP operations when you do not specify the CONCURRENT keyword.

You can use the SET PATCH command to set the patch byte at offset X'4B' to X'FF', or you can modify the job control language (JCL), as shown below:

```
//SYSIN DD *
 NAME ADRDSSU ADRPATCH
 VER   4B        00
 REP   4B        FF
```

Changing the Default DEFRAG Processing of LINKLIST-Indicated Data Sets (OW43874)

During DEFRAG operations, DFSMSdss does not relocate extents for data sets that are LINKLIST-indicated. Installations must be able to indicate to DEFRAG that it must move such extents. This capability is most likely to be needed for volumes that have been cloned and where the linklist data sets contained on the cloned volume are not being used in a production environment.

DFSMSdss provides a patch byte to allow you to change the DEFRAG command default processing of LINKLIST-indicated data set extents. This patch byte is honored only when the SET PATCH command sets it on, dynamically.

This function is affected by setting the flag at offset X'4E' using the SETPATCH command. The settings are:

X'00'	DFSMSdss functions normally. DEFRAG does not relocate extents for data sets that are LINKLIST-indicated.
Any setting other than X'00'	DEFRAG command processing then moves as needed, any selected extents of a LINKLIST-indicated data set that are contained on the volume. This occurs, even if you cannot obtain serialization for the data set.

DFSMSdss issues message ADR254I during function task start-up. Message ADR254I indicates that the DEFRAG command is using the installation patch byte to override the normal default processing of LINKLIST-indicated data sets.

Recommendation: Due to the possible misuse of this capability, you might want to restrict its use. The patch byte is honored only when the SET PATCH command sets it on, dynamically. You can restrict who can dynamically set patch bytes with the SET PATCH command. To do this, use a RACF facility class that requires read access to STGADMIN.ADR.PATCH. In addition, use a RACF facility class that requires read access to STGADMIN.ADR.DEFRAG to restrict the use of the DEFRAG command.

& Changing the FASTREPLICATION Default Setting During Copy and & Defrag (OA11637)

& New Patch Description

& During DEFRAG and COPY operations, fast replication method is used when it
& can be; this is considered FASTREPLICATION(PREFERRED) which is the
& DFSMSdss default setting when the FASTREPLICATION keyword is not specified.
& The FASTREPLICATION keyword overrides this default.

& DFSMSdss now provides a patch byte that allows you to change this default
& setting to not use fast replication unless you specify
& FASTREPLICATION(PREFERRED) or FASTREPLICATION(REQUIRED).

& The FASTREPLICATION default is affected by setting the flag at offset X'4F' in
& ADRPATCH. The settings are:

& **X'00'**	If the FASTREPLICATION keyword is not

specified, thenDFSMSdss processes as if the FASTREPLICATION(PREFERRED) keyword was specified.

Any setting other than X'00' If the FASTREPLICATION keyword is not specified, then DFSMSdss processes as if the FASTREPLICATION(NONE) keyword was specified.

Note: The Options Installation Exit Routine, ADRUIXIT, allows the final override to the FASTREPLICATION setting. In this exit there is no indication that the FASTREPLICATION setting is the default or if the keyword is specified.

To set the flag on (for example, X'FF'), modify the sample JCL as follows:

```
//SYSIN DD *
    NAME ADRDSSU ADRPATCH
    VER   4F        00
    REP   4F        FF
```

Tuning Hardware Assisted Compression (OA13300)

You may tune hardware assisted compression to improve the performance of DUMP processing. With hardware assisted compression, a compression dictionary is built using user data. This dictionary is then used to compress that user data and subsequent user data. Unfortunately, building the compression dictionary is expensive in terms of performance, so it is preferable to avoid building the compression dictionary too often. However, data could be compressed better, resulting in smaller output dump data sets, if the dictionary was rebuilt sooner.

During a physical dump process, the quality of compression is recorded and used in deciding when to rebuild the compression dictionary. Two measurements are used.

The first measurement is the quality of compression achieved. This is a percentage, calculated by dividing the compressed size of the data by the original size of the data. A compression is considered poor when the percentage is greater than a threshold value. The default threshold value is 94%.

The second measurement is how many poor compressions are allowed before rebuilding the compression dictionary. The default value for the number of poor compressions allowed before rebuilding the compression dictionary is 15.

DFSMSdss provides patch bytes to change the number of poor compressions and threshold value.

To change the number of poor compressions allowed before rebuilding the compression, the byte at offset X’50’ can be set to any value between X’01’ and X’FF’. When the byte at offset X’50’ is X’00’ then DFSMSdss uses the default value of 15.

To change the threshold value that is considered an acceptable compression, the byte at offset X’51’ can be set to one of the following values: X’00’ DFSMSdss uses the default value of 94. X’01’-X’64’ DFSMSdss uses the specified value. X’65’-X’FF’ DFSMSdss uses the default value of 94.

The number of poor compressions allowed is affected by setting the flag at offset X'50' using the SETPATCH command. The settings are:

X'00'	DFSMSdss functions normally. The default value of 15 poor compressions is used to decide when to rebuild the compression dictionary.
Any setting other than X'00'	DFSMSdss uses the value set as the number of poor compressions to allow before rebuilding the compression dictionary.

The target compression threshold is affected by setting the flag at offset X'51' using the SETPATCH command. The settings are:

X'00'	DFSMSdss functions normally. The default value of 94% is used as a threshold to make a decision whether to count the compression as poor or not.
X'01' - X'64'	DFSMSdss uses this value (1%-100%) as the target compression threshold.
X'65' - X'FF'	These values are invalid as a compression threshold percentage and will be ignored. When ignored, the default value of 94% is used as the threshold.

ADRPTCHB Data Area

Table 4 lists the mapping of flags in ADRPTCHB.

Table 16. ADRPTCHB Mapping Macro

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	4096	ADRPTCHB	
0	(0)	CHARACTER	8	PTCHEYE	
8	(8)	UNSIGNED	1	PTCHARRY (4088)	
8	(8)	UNSIGNED	1	PBUVSPRE	USE PREALLOCATED VSAM
9	(9)	UNSIGNED	1	PBDUPKEY	DUPLICATE VSAM KEY
10	(A)	UNSIGNED	2	PBTIMOUT	TIMEOUT CONSTANT
12	(C)	UNSIGNED	1	PBADINFO	RESERVED
13	(D)	UNSIGNED	1	PBWAITFG	WAIT/RETRY FLAG FOR VTOC/VVDS ENQ OR RESERVE
14	(E)	UNSIGNED	1	PBWAIT#	WAIT TIME FOR VTOC/VVDS ENQ OR RESERVE
15	(F)	UNSIGNED	1	PBRETRY#	RETRY COUNT FOR VTOC/VVDS ENQ OR RESERVE
16	(10)	UNSIGNED	1	PBRESERV	RESERVE BYTE TO ACCOUNT FOR OFFSET ERRORS
17	(11)	UNSIGNED	1	PBREVOKE	GIVE REVOKED DS OWNER ACCESS TO SMS CONSTRUCTS
18	(12)	UNSIGNED	1	PBBPDSE	BROKEN PDSE RESTORE FLAG
19	(13)	UNSIGNED	1	PBNRACFI	NO RACF INDIC. FOR LOG. REST.
20	(14)	UNSIGNED	1	PBNMVNCV	NO MV ALLOC IF ALSO UNDEFINED DATA SET
21	(15)	UNSIGNED	1	PBBYPBWO	BYPASS BWO
22	(16)	UNSIGNED	1	PBNACSAU	NO ACS AUTH. CHECKING DURING RESTORE
23	(17)	UNSIGNED	1	PBNSLECT	WARNING MESSAGE FOR DATA SETS NOT SELECTED.
24	(18)	UNSIGNED	1	PBCCRESE	DO RESET WITH CONCURRENT COPY
25	(19)	UNSIGNED	1	PBNO482E	RESTORE DS FROM DUMP TAPE WHICH HAD MSGADR727E
26	(1A)	UNSIGNED	1	PBMISCNT	RESTORE DS FROM DUMP TAPE DUMPED WITH CC PRIOR TO INSTALLATION OF FIX FOR OY67724

Table 16. ADRPTCHB Mapping Macro (continued)

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
27	(1B)	UNSIGNED	1	PBX99365	TREAT ALL F1 DSCB EXP DATE 1999.365 FROM DUMP AS ACTUAL EXP DATE WHEN DEFINING TARGET
28	(1C)	UNSIGNED	1	PBEX2000	TREAT ALL F1 DSCB SMALL EXP DATES FROM DUMP AS 2NNN EXP DATES WHEN DEFINING TARGET
29	(1D)	UNSIGNED	1	PBEOFNO	DO NOT INSERT EOF DURING COPY ALLDATA FROM MULTIVOLUME TO SINGLE-VOLUME DATASET
30	(1E)	UNSIGNED	1	PBBYLENQ	BYPASS PN27748 CODE WHICH DELAYS RESET UNCAT AND HOLDS LONGER ENQUEUES
31	(1F)	UNSIGNED	1	*	RESERVED BYTE
Note: The following 6 fields starting at offset X'20' are intended for installations to “adjust” target PDSE DS1SCAL3 which may have been incorrectly altered by PE-OW04199. The logic is: IF 2nd_qty > PBCYLHI THEN 2nd_qty = PBCYLQTY. Any non-zero value in PBCYLHI, PBTRKHI, or PBBLKHI will activate the “adjustment code” for the corresponding allocation type in ADRNEWDS.					
32	(20)	UNSIGNED	4	PBCYLHI	HI THRESHOLD FOR CHECKING DS1SCAL3 IN CYL ALLOCATION
36	(24)	UNSIGNED	4	PBCYLQTY	CHANGE DS1SCAL3 TO THIS VALUE IF IT IS > PBCYLHI
40	(28)	UNSIGNED	4	PBTRKHI	HI THRESHOLD FOR CHECKING DS1SCAL3 IN TRK ALLOCATION
44	(2C)	UNSIGNED	4	PBTRKQTY	CHANGE DS1SCAL3 TO THIS VALUE IF IT IS > PBTRKHI
48	(30)	UNSIGNED	4	PBBLKHI	HI THRESHOLD FOR CHECKING DS1SCAL3 IN BLK ALLOCATION
52	(34)	UNSIGNED	4	PBBLKQTY	CHANGE DS1SCAL3 TO THIS VALUE IF IT IS > PBBLKHI
56	(38)	UNSIGNED	1	PBROREFD	DS1REFD RESTORE OLD 0 = USE OLD DATE 1 = USE TODAY’S DATE
57	(39)	UNSIGNED	1	PBRNREFD	DS1REFD RESTORE NEW 1 = USE OLD DATE 0 = USE TODAY’S DATE
58	(3A)	UNSIGNED	1	PBCOREFD	DS1REFD COPY OLD 0 = USE OLD DATE 1 = USE TODAY’S DATE
59	(3B)	UNSIGNED	1	PBCNREFD	DS1REFD COPY NEW 1 = USE OLD DATE 0 = USE TODAY’S DATE
60	(3C)	UNSIGNED	1	PBCDELNP	DEGRADE ’E’ MSG TO ’W’ IF COPY DEL NO TGT PROFILE
61	(3D)	UNSIGNED	1	PBNACSAC	NO ACS AUTHORIZATION CHECKING DURING COPY
62	(3E)	UNSIGNED	1	PBINVTOK	DON’T ERASE INVALID TRACK DURING COPY/RESTORE
63	(3F)	UNSIGNED	1	PBSRCVOL	PASS SOURCE PRIMARY VOLUMES FROM WHICH DS WAS DUMPED
64	(40)	UNSIGNED	1	PBVOLOPT	’PATCHABLE’ VOLCOUNT OPTION FLAG DATA SET
		1...		PBVCCUR	VOLCOUNT(*)
		.1..		PBVCSRC	VOLCOUNT(SRC)
		..1.		PBVCNUM	VOLCOUNT(N(NN))
		...1		PBVCANY	VOLCOUNT(ANY)
		 1111		*	UNUSED
65	(41)	UNSIGNED	1	PBVCVAL	NUMBER OF VOLUMES TO USE FOR OUTPUT DATA SET
66	(42)	UNSIGNED	1	PBMSCIOK	OK TO HAVE MISSING INDEX CI IN KEYED VSAM DATA SET - VALIDATE
67	(43)	UNSIGNED	1	PBRESV60	WAS FOR DEFRAG TO MOVE CHECKPOINT INDICATED DATA SET

Table 16. ADRPTCHB Mapping Macro (continued)

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
68	(44)	UNSIGNED	1	PBFUDGE	OVER ALLOCATE DATA SET BY % IN FUDGE FACTOR
69	(45)	UNSIGNED	1	PBNORLS	DO NOT DO RLS QUIESCE
70	(46)	UNSIGNED	1	PBROCRED	DS1CREDT RESTORE OLD 0 = USE OLD DATE 1 = USE TODAY'S DATE
71	(47)	UNSIGNED	1	PBRNCRED	DS1CREDT RESTORE NEW 1 = USE OLD DATE 0 = USE TODAY'S DATE
72	(48)	UNSIGNED	1	PBCOCRED	DS1CREDT COPY OLD 0 = USE OLD DATE 1 = USE TODAY'S DATE
73	(49)	UNSIGNED	1	PBCNCRED	DS1CREDT COPY NEW 1 = USE OLD DATE 0 = USE TODAY'S DATE
74	(4A)	UNSIGNED	1	*	RESERVED
75	(4B)	UNSIGNED	1	PBVDPDSE	VALIDATE PDSE 1 = ENABLE VALIDATION 0 = DISABLE VALIDATION
76	(4C)	UNSIGNED	1	*	RESERVED
77	(4D)	UNSIGNED	1	*	UNUSED
78	(4E)	UNSIGNED	1	PBMOVLL	OVERRIDE DEFRAG TO MOVE LINKLIST-INDICATED DATA SET
79	(4F)	UNSIGNED	1	PBFCDEF	CHANGE FASTREPLICATION DEFAULT 1 = DEFAULT FASTREPLICATION(NONE) 0 = DEFAULT FASTREPLICATION(PREFERRED)
80	(50)	UNSIGNED	1	PBDCTBD	NUMBER OF POOR COMPRESSIONS TO ALLOW BEFORE REBUILDING COMPRESSION DICTIONARY
81	(51)	UNSIGNED	1	PBCMPTH	THRESHOLD OVER WHICH TO CONSIDER A COMPRESSION POOR

ADRPTCHB Cross-Reference

	Name	Hex Offset	Hex Value
	ADRPTCHB	0	
	PBADINFO	C	
	PBBLKHI	30	
	PBBLKQTY	34	
	PBBPDSE	12	
	PBBYLENQ	1E	
	PBBYPBWO	15	
	PBCCRESE	18	
	PBCDELNP	3C	
&	PBCMPTH	51	
	PBCNREFD	3B	
	PBCOCRED	48	
	PBCOREFD	3A	
	PBCYLHI	20	
	PBCYLQTY	24	
&	PBDCTBD	50	
	PBDUPKEY	9	
	PBEOFNO	1D	
	PBEX2000	1C	
&	PBFCDEF	4F	
	PBFUDGE	44	
	PBINVTOK	3E	
	PBMISCNT	1A	
	PBMOVLL	4E	
	PBMSCIOK	42	
	PBNACSAC	3D	
	PBNACSAU	16	
	PBNMVNCV	14	
	PBNO482E	19	
	PBNORLS	45	
	PBNRACFI	13	
	PBNSLECT	17	
	PBRESERV	10	
	PBRESV60	43	
	PBRETRY#	F	
	PBREVOKE	11	
	PBRNCRED	47	
	PBRNREFD	39	
	PBROCRED	46	
	PBROREFD	38	
	PBSRCVOL	3F	
	PBTIMOUT	A	
	PBTRKHI	28	
	PBTRKQTY	2C	
	PBUVSPRE	8	
	PBVCANY	40	10
	PBVCCUR	40	80
	PBVCNUM	40	20
	PBVCSRC	40	40
	PBVCVAL	41	
	PBVDPDSE	4B	
	PBVOLOPT	40	
	PBWAIT#	E	
	PBWAITFG	D	
	PBX99365	1B	

DFSMSdss Patch Area

Name	Hex Offset	Hex Value
PTCHARRY	8	
PTCHEYE	0	

Appendix E. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I*for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`www.ibm.com/servers/eserver/zseries/zos/bkserv/`

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.

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, 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.

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
USA

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 information 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.

Programming interface information

This publication primarily documents information that is NOT intended to be used as a Programming Interface of DFSMSdss.

This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of DFSMSdss. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Programming Interface information

End of Programming Interface information

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
Information Enabling Requests
Dept. DZWA
5600 Cottle Road
San Jose, CA 95193
U.S.A.
```

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

Trademarks

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

IBM	IBMLink
AIX	IMS
CICS	MVS
DB2	MVS/ESA
DFSMSdfp	RACF
DFSMSdss	RAMAC
DFSMShsm	OS/390
DFSMSrmm	System 370
DFSORT	z/OS

Enterprise Storage Server	ZSeries
FlashCopy	z/VM

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft 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.

Glossary

This glossary defines technical terms and abbreviations used in DFSMSdss documentation. If you do not find the term you are looking for, refer to the index of the appropriate DFSMSdss manual or view *IBM Dictionary of Computing*, located at http://www.ibm.com/networking/nsg/nsgmain.htm

A

ABARS. Aggregate backup and recovery support.

ABEND. Abnormal end of task. End of a task, a job, or a subsystem because of an error condition that cannot be resolved by recovery facilities while the task is performed.

ABENDxxx. The keyword that identifies the abnormal end of DFSMSdss because of a system-detected error.

ABSTR. A subparameter of the SPACE parameter in a DD statement. It indicates that specified tracks be assigned to a data set.

ACCEPT processing. An SMP/E process necessary for installing the FMIDs. SMP/E ACCEPT processing uses JCL to accept the modules and macros necessary to run the FMIDs. The FMIDs are accepted into the DLIBs from the temporary data sets.

access method services. A multifunction service program that is used to manage both VSAM and non-VSAM data sets and integrated catalog facility or the ICF catalog. It is used to define data sets and allocate space for them; convert indexed-sequential data sets to key-sequenced data sets; modify data set attributes in the catalog; reorganize data sets; facilitate data portability between operating systems; create backup copies of data sets, data set records, and catalog entries; help make inaccessible data sets accessible; list the records of data sets and catalogs; define and build alternate indexes; and convert OS CVOLs and the ICF catalog to integrated catalog facility catalogs.

ACDS. Active control data set.

ACS. Automatic class selection.

ADSP. Automatic data set protection.

alias. An alternate name for a member of a partitioned data set.

ALLOC. A space allocation parameter that indicates type, such as cylinders or tracks.

alternate index. In systems with VSAM, a key-sequenced data set containing index entries organized by the alternate keys of its associated base data records. It provides an alternate means of locating records in the data component of a cluster on which the alternate index is based.

alternate index cluster. In VSAM, the data and index components of an alternate index.

APAR. Authorized program analysis report.

APF. Authorized program facility.

application interface. An interface used to invoke DFSMSdss from another program.

apply processing. In SMP and SMP/E, the process, initiated by the APPLY command, that places system modifications (SYSMODS) into the target system libraries.

attach. In programming, to create a task that can be performed asynchronously with the performance of the mainline code.

authorization. (1) The right granted to a user to communicate with or make use of a computer system. (2) The process of giving a user either complete or restricted access to an object, resource, or function.

authorized program analysis report (APAR). A request for correction of a problem caused by a suspected defect in a current unaltered release of a program.

automatic class selection (ACS). A mechanism for assigning SMS classes and storage groups.

automatic data set protection (ADSP). A system function, enabled by the SETROPTS ADSP specification and the assignment of the ADSP attribute to a user with ADDUSER or ALTUSER, that causes all permanent data sets created by the user to be automatically defined to RACF with a discrete RACF profile..

B

backout. The CICSVR function that you can use if CICS fails in the attempt to back out uncommitted changes on a VSAM sphere. Using information from the RCDS, CICSVR constructs a job to back out uncommitted changes on a VSAM data set as indicated on the log.

backup. The process of creating a copy of a data set to ensure against accidental loss.

backup while open. DFSMSdss can perform backup of data sets that are open for update for a long period of time (like CICS). DFSMSdss can perform a logical data set dump of these data sets even if another application has them serialized.

base cluster. In systems with VSAM, a key-sequenced or entry-sequenced data set over which one or more alternate indexes are built.

basic catalog structure (BCS). The name of the catalog structure in the integrated catalog facility environment. An integrated catalog facility catalog consists of a BCS and its related VSAM volume data sets (VVDSs).

basic direct access method (BDAM). An access method used to directly retrieve or update particular blocks of a data set on a direct access device.

basic partitioned access method (BPAM). An access method that can be applied to create program libraries in direct access storage for convenient storage and retrieval of programs.

basic sequential access method (BSAM). An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or a direct access device.

BCS. Basic catalog structure.

BDAM. Basic direct access method.

BLK. A subparameter of the SPACE parameter in a DD statement. It specifies that space is allocated by blocks.

block length. Synonym for block size.

block size. (1) The number of data elements in a block. (2) A measure of the size of a block, usually specified in units such as records, words, computer words, or characters. (3) Synonymous with block length. (4) Synonymous with physical record size.

BPAM. Basic partitioned access method.

bpi. Bits per inch.

Broken data set. Data sets which do not conform to IBM data set standards are referred to as *broken*. Broken data sets are either missing catalog entries, VTOC entries, or VVDS entries; or, have invalid catalog entries, VTOC entries, or VVDS entries.

BSAM. Basic sequential access method.

C

CA. Control area.

call. (ISO) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point.

card image. A one-to-one representation of the hole patterns of a punched card; for example, a matrix in which a one represents a punch and a zero represents the absence of a punch.

CC-compatible SnapShot. See *virtual concurrent copy*.

CCHHR. Cylinder, cylinder, head, head, record.

CCW. Channel command word.

CDE. Contents directory entry.

CDS. Control data set.

channel command word (CCW). A doubleword at the location in main storage specified by the channel address word. One or more CCWs make up the channel program that directs data channel operations.

CI. Control interval.

CICS. Customer Information Control System.

CICSVR. CICS VSAM Recovery.

CICS VSAM Recovery (CICSVR). CICS VSAM Recovery is an IBM product that recovers your lost or damaged VSAM data. CICSVR V3.1 recovers VSAM data in the following environments:

- CICSVR VSAM batch logging (when the VSAM data sets are not accessed in record level sharing mode)
- CICS TS
- CICS V4

CLIST. Command list.

complete recovery. The CICSVR function that consists of forward recovery followed by backout, if needed. In CICSVR complete recovery, CICSVR restores a DFSMShsm or DFSMSdss backup for you.

component identification keyword. The first keyword, represented as a number, in a set of keywords used to describe a DFSMSdss program failure.

compress. (1) To reduce the amount of storage required for a given data set by having the system replace identical words or phrases with a shorter token associated with the word or phrase. (2) To reclaim the unused and unavailable space in a partitioned data set that results from deleting or modifying members by moving all unused space to the end of the data set.

COMPRESS command. The DFSMSdss function that reduces partitioned data sets by taking unused space and consolidating it at the end of the data set.

compressed format. A particular type of extended-format data set specified with the (COMPACTION) parameter of data class. VSAM can compress individual records in a compressed-format data set. SAM can compress individual locks in a compressed-format data set. See compress.

concatenation. An operation that joins two characters or strings in the order specified, forming one string whose length is equal to the sum of the lengths of the two characters or strings.

concurrent copy. A function to increase the accessibility of data by letting you make a consistent backup or copy of data concurrent with normal application program processing.

concurrent copy-compatible (CC-compatible) SnapShot. See *virtual concurrent copy*.

conditioned volume.. The target volume from a previous FULL volume COPY operation which specified DUMPCONDITIONING.

control area (CA). A group of control intervals used as a unit for formatting a data set before adding records to it. Also, in a key-sequenced data set, the set of control intervals, pointed to by a sequence-set index record, that is used by VSAM for distributing freespace and for placing a sequence-set index record adjacent to its data.

control interval (CI). A fixed-length area of auxiliary storage space in which VSAM stores records. It is the unit of information transmitted to or from auxiliary storage by VSAM.

control volume (CVOL). A volume that contains one or more indexes of the catalog.

constructs. A collective name for data class, storage class, management class, and storage group.

CONVERTV command. The DFSMSdss function that converts volumes to and from Storage Management Subsystem management without data movement.

COPY command. The DFSMSdss function that performs data set, volume, and track movement.

CP. Control program.

CREDT. Creation date.

CSW. Channel status word.

CVAF. Common VTOC access facility.

CVOL. Control volume.

CVT. Communication vector table.

CYL. A subparameter of the SPACE parameter in a DD statement. It specifies that space is allocated by cylinders.

D

DADSM. The direct access space management program that maintains the VTOC, VTOCIX, and space on a volume.

DAM. Direct access method.

DASD. Direct access storage device.

DASD ERP. DASD error recovery procedure.

DASD volume. A DASD space identified by a common label and accessed by a set of related addresses.

data class. A list of data set allocation parameters and the values that are used when allocating a new SMS-managed data set.

data compression (run-length). A method of encoding repetitive series of identical characters so that they occupy less space on a dump tape. Data compression is supported by both physical dump and logical dump processing.

Data Facility Storage Management Subsystem/MVS (DFSMS/MVS). The complementary functions of DFSMSdfp, DFSMSdss, DFSMShsm, and DFSMSrmm which, together with RACF provide a system-managed, administrator-controlled storage environment.

data set backup. Backup to protect against the loss of individual data sets.

data set change indicator. A bit that is set by OPEN when the data set is opened for processing other than input. This flag is supported on MVS systems that have data-set-changed flag support installed.

data set FlashCopy. One of the FlashCopy Version 2 functions. See also *FlashCopy Version 2.*

DAU. Direct access unmovable.

DB2. IBM DATABASE 2.

DCB. Data control block.

DEFRAG command. The DFSMSdss function that consolidates the free space on a volume to help prevent out-of-space abends on new allocations.

DEQ. An assembler language macro instruction used to remove control of one or more serially reusable resources from the active task.

DFSMS. Data Facility Storage Management Subsystem.

DFSMS environment. An environment that helps automate and centralize the management of storage. This is achieved through a combination of hardware, software, and policies. See also *system-managed storage*.

DFSMSdfp. A DFSMS/MVS functional component that provides functions for storage management, data management, program management, device management, and distributed data access.

DFSMSdss. A DFSMS/MVS functional component used to copy, move, dump, and restore data sets and volumes. DFSMSdss is the primary data mover of DFSMS/MVS.

DFSMShsm. A DFSMS/MVS functional component used for backing up and recovering data, and managing space on volumes in the storage hierarchy.

DFSMS/MVS. Data Facility Storage Management Subsystem/MVS.

DFSORT. Data Facility Sort.

DIAGNOSE. An access method services command that scans an integrated catalog facility basic catalog structure (BCS) or a VSAM volume data set (VVDS) to validate the data structure.

DIRF. DADSM interrupt recording facility. If a system fails, or a permanent I/O error occurs during allocation of space or during performance of a routine that updates the VTOC, the VTOC may be in error. To ensure that an error is recorded, the DADSM routines turn on a bit in the VTOC upon entry to a DADSM function, and, if no errors occur during processing, turn off that bit upon exiting from that function.

distribution libraries. IBM-supplied partitioned data sets on tape containing one or more components that the user restores to disk for subsequent inclusion in a new system.

DLIB. Distribution library.

DOC. In diagnosing program failures, the keyword that identifies an error in the documentation of a program.

DOS. Disk Operating System.

DOS bit. On a volume without an indexed VTOC, a bit that indicates that the free space map is invalid.

DOS/VSE. DOS/Virtual Storage Extended.

DSCB. Data set control block.

DSCHA. A DFSMSdss keyword that is used in BY filtering. It indicates that the data set is to be selected if the data set has been changed.

dsname. Data set name.

DSORG. Data set organization. It is specified in the JCL as "DSORG=".

DUMP command. The DFSMSdss function used to back up data sets, tracks, and volumes.

dynamic allocation. Assignment of system resources to a program when the program is performed rather than when it is loaded main storage.

E

early warning system (EWS). A microfiche copy of the information contained in the software support facility (SSF), organized by component identification number, and indexed by APAR symptom code. EWS is published monthly and available to customers of IBM licensed programs.

ECB. Event control block.

EC mode. Engineering change mode.

empty data set. A data set in which the pointer to the last-used block is 0.

ENQ. An assembler language macro instruction that requests the control program to assign control of one or more serially reusable resources to the active task. It is also used to determine the status of a resource; that is, whether it is immediately available or in use, and whether control has been previously requested for the active task in another ENQ macro instruction.

entry-sequenced data set (ESDS). In VSAM, a data set whose records are loaded without respect to their contents and whose RBAs cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

EOF. End-of-file.

EOJ. End of job.

erase-on-scratch. The physical erasure of data on a DASD data set when the data set is deleted (scratched).

ESA. Enterprise Systems Architecture.

ESS. Enterprise Storage Server.

ESDS. Entry-sequenced data set.

ESTAE. Extended specify task abnormal exit.

EQ. Equal to.

EWS. Early warning system.

EXCP. Execute channel program.

execute channel program (EXCP). A macro used to access a data set without specifying the organization.

EXPDT. Expiration date.

extended format. The format of a data set that has a data set name type (DSNTYPE) of EXTENDED. The data set is structured logically the same as a data set that is not in extended format but the physical format is different. Data sets in extended format can be striped or compressed. Data in an extended format VSAM KSDS can be compressed. See also *striped data set* and *compressed format*.

extended specify task abnormal exit (ESTAE). A task recovery routine that provides recovery for those programs that run enabled, unlocked, and in task mode.

extent. A continuous space on a DASD volume occupied by a data set or portion of a data set. An extent of a data set contains a whole number of control areas.

F

FC. CVAF function code.

FCEC. CVAF function-error code.

filtering. The process of selecting data sets based on specified criteria. These criteria consist of fully- or partially-qualified data set names, or of certain data set characteristics, or of both.

FlashCopy. A function of the Enterprise Storage Server (ESS) and DFSMSdss that provides instant data copying. When resources allow, DFSMSdss automatically selects FlashCopy.

FlashCopy V1. FlashCopy Version 1.

FlashCopy V2. FlashCopy Version 2.

FlashCopy Version 1. The initial FlashCopy feature provided by ESS. FlashCopy Version 1 is supported at the volume level. Both the source and target volumes must reside on the same logical subsystem (LSS). Each volume can be in one FlashCopy relationship.

FlashCopy Version 2. FlashCopy Version 2 provides enhancements to the existing FlashCopy Version 1 feature of ESS. These enhancements include data set FlashCopy, multiple relationship FlashCopy, incremental FlashCopy, improvement in FlashCopy establish time, elimination of LSS constraint, and consistency group support. The source and target volumes must reside in the ame ESS. DFSMS exploits data set FlashCopy.

FMID. Function modification identifier.

forward recovery. The CICSVR function that reapplies all changes to the VSAM sphere since the last backup. CICSVR gets the information it needs to construct the recovery job from the RCDS. The contents of the logs are applied to the VSAM sphere to return it to its exact state before the data was lost. With CICSVR forward recovery, CICSVR restores a DFSMShsm or DFSMSdss backup for you.

fragmentation index. The qualitative measure of the scattered free space on a volume.

fully-qualified data set name. A data set in which all the qualifiers are completely spelled out.

function modification identifier (FMID). A code that identifies the release levels of a program product.

FVL. Function vector list.

G

GDG. Generation data group.

GDS. Generation data set.

generalized trace facility (GTF). An optional OS/VS service program that records significant systems events, such as supervisor calls and start I/O operations, for the purpose of problem determination.

generation data group (GDG). A collection of historically related non-VSAM data sets that are arranged in chronological order; each data set is a generation data set.

generation data set. One generation of a generation data group.

global resource serialization (GRS). A component of z/OS used for serializing use of system resources and for converting hardware reserves on DASD volumes to data set enqueues.

GT. Greater than.

GTF. Generalized trace facility.

H

HFS. Hierarchical file system.

I

ICKDSF. Device Support Facilities.

IDCAMS. Access Method Services.

IDRC. Improved data recording capability.

IMS/VS. Information Management System/Virtual Storage.

INCORROUT. In diagnosing program failures, the keyword that identifies incorrect or missing program output.

incremental backup. A process in which data sets are backed up only if they have changed since their last backup.

installation exit. The means specifically described in an IBM software product's documentation by which an IBM software product may be modified by a customer's system programmers to change or extend the functions of the IBM software product. Such modifications consist of exit routines written to replace one or more existing modules of an IBM software product, or to add one or more modules or subroutines to an IBM software product, for the purpose of modifying (including extending) the functions of the IBM software.

integrated catalog facility. A facility by which VSAM data set volume-related fields are separated from the catalog and maintained in the VVDS on the volume on which the data set resides.

integrated catalog facility catalog. A catalog that is composed of a basic catalog structure (BCS) and its related volume table of contents (VTOC) and VSAM volume data sets (VVDSs).

Interactive Problem Control System (IPCS). A component of MVS that permits online problem management, interactive problem diagnosis, problem tracking, and problem reporting.

Interactive Storage Management Facility (ISMF). An interactive interface of DFSMS/MVS that allows users and storage administrators access to the storage management functions.

Interactive System Productivity Facility (ISPF). An IBM licensed program used to develop, test, and run application programs interactively. ISPF is the interactive interface for all storage management functions.

I/O. Input/output.

IPCS. Interactive Problem Control System.

IPL. Initial program load.

ISMF. Interactive Storage Management Facility.

ISAM. Indexed sequential access method.

ISMF. Interactive Storage Management Facility.

ISPF. Interactive Systems Productivity Facility.

ISPF/PDF. Interactive Systems Productivity Facility/Program Development Facility.

J

JCL. Job control language.

JES. Job entry subsystem.

JES2. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for operation, processes their output, and purges them from the system. In an installation site with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing.

JES3. An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for operation, processes their output, and purges them from the system. In complexes that have several loosely coupled processing units, the JES3 program manages processors so that the global processor exercises centralized control over the local processors and distributes jobs to them via a common job queue.

JFCB. Job file control block.

job control language (JCL). A problem-oriented language used to identify the job or describe its requirements to an operating system.

job entry subsystem (JES). A system facility for spooling, job queuing, and managing I/O.

JSCB. Job step control block.

K

K. Kilobyte: 1 024 bytes.

key-sequenced data set. A VSAM file or data set whose records are loaded in ascending key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in key sequence by means of distributed free space. Relative byte addresses can change because of control interval or control area splits.

keyword. A symptom that describes one aspect of a program failure.

KRDS. Keyrange data set. Also known as a key-sequenced data set with key ranges.

KSDS. Key-sequenced data set.

L

LASTCC. Last condition code.

LDS. Linear data set.

like devices. Devices that have the same track capacity and number of tracks per cylinder (for example, 3380 Model D, Model E, and Model K).

LINK. An assembler language macro instruction that causes control to be passed to a specified entry point.

The linkage relationship established is the same as that created by a basic assembler language (BAL) instruction.

link-pack area (LPA). An area of virtual storage that contains reenterable routines that are loaded at IPL (initial program load) time and can be used concurrently by all tasks in the system.

load module. A computer program in a form suitable for loading into main storage for operation.

load module library. A partitioned data set used to store and retrieve load modules.

logical DUMP operation (data set). A DUMP operation in which logical processing is performed.

logical processing (data set). Processing that treats each data set and its associated information as a logical entity. As an example, DFSMSdss processes an entire data set before beginning with the next one.

logical storage subsystem (LSS). Used internally by ESS to manage a set of logical volumes which are associated with an individual device adapter, e.g., a physical ESS subsystem may be partitioned into multiple logical storage subsystems.

logical RESTORE operation (data set). A RESTORE operation that uses as input a data set produced by a logical DUMP operation.

logical volume. The output produced from a physical DUMP operation, for which all data is derived from a single DASD volume.

LOOP. In diagnosing program failures, the keyword that identifies a program failure in which some part of the program repeats endlessly.

LPA. Link-pack area.

LSS.. Logical storage subsystem.

LT. Less than.

LRECL. Logical record length.

LVOL. Logical volume.

M

Mb. Megabit; 1 048 576 bits.

MB. Megabyte; 1 048 576 bytes.

maintenance-level keyword. In diagnosing program failures, a keyword that identifies the maintenance level of DFSMSdss.

management class. A list of the migration, backup, and retention parameters and the values for an SMS-managed data set.

map record. The record that maps the tracks dumped by DFSMSdss.

MAXCC. Maximum condition code.

MCS. Multiple console support.

MENTITY. Model entity.

minivolume. In an MVS system running on VM/370, an OS/VS-formatted VM/370 minidisk whose size is equal to or less than that of the real volume. DFSMSdss uses the device size specified in the VTOC. Minivolumes are supported only by the system version of DFSMSdss.

MSGADRnnnt. In diagnosing program failures, the DFSMSdss message keyword that tells of an error, or seems itself to be in error.

MVS. Multiple Virtual Storage.

N

NVR. Non-VSAM volume record.

O

operating system (OS). Software that controls the execution of programs; an operating system may provide services such resource allocation, scheduling, input/output control, and data management.

OS. Operating system.

P

pageable link-pack area (PLPA). Link-pack area.

PAM. Partitioned access method.

partially qualified data set name. A data set name in which the qualifiers are not spelled out. Asterisks and percent signs are used in place of the undefined qualifiers.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

partitioned data set extended (PDSE). A system-managed, page-formatted data set on direct access storage. A PDSE contains an indexed directory and members similar to the directory and members of partitioned data sets. A PDSE can be used instead of a partitioned data set.

PDS. Partitioned data set.

PDSE. Partitioned data set extended.

PERFM. In diagnosing program failures, the keyword that identifies degradation in program performance.

physical DUMP operation (data set). A DUMP operation in which physical processing is performed.

physical processing (data set). Processing that moves data at the track-image level and can operate against volumes, tracks, and data sets. As an example, DFSMSdss may only process one volume of a multivolume data set.

PLPA. Pageable link-pack area.

POU. Partitioned organization unmovable.

PRB. Program request block.

private library. A user-owned library that is separate and distinct from the system library.

PSU. Physical sequential unmovable.

PSW. Program status word.

PTF. Program temporary fix.

Q

QSAM. Queued sequential access method.

qualified name. A data set name consisting of a string of names separated by periods; for example, "TREE.FRUIT.APPLE" is a qualified name.

qualifier. Each component name in a qualified name other than the rightmost name. For example, "TREE" and "FRUIT" are qualifiers in "TREE.FRUIT.APPLE."

queued sequential access method (QSAM). An extended version of the basic sequential access method (BSAM). Input data blocks awaiting processing or output data blocks awaiting transfer to auxiliary storage are queued on the system to minimize delays in I/O operations.

R

RACF. Resource Access Control Facility.

RAMAC Virtual Array (RVA). A DASD that uses a virtual array architecture.

RB. Request block.

RBA. Relative byte address.

RCDS. Recovery Control Data Set.

RDJFCB. Read job file control block.

RECEIVE processing. An SMP/E process necessary to install new product libraries. During this process, the code, organized as unloaded partition data sets, is loaded into temporary SMPTLIB data sets. SMP/E RECEIVE processing automatically allocates the temporary partitioned data sets that correspond to the files on the tape, and loads them from the tape.

RECFM. Record format.

recovery. The process of rebuilding data after it has been damaged or destroyed, often by restoring a backup version of the data or by reapplying transactions recorded in a log.

REFDT. A DFSMSdss keyword used in BY filtering. It indicates the last-referenced date.

relative byte address (RBA). The displacement (expressed as a fullword binary integer) of a data record or a control interval from the beginning of the data set to which it belongs, independent of the manner in which the data set is stored.

relative record data set (RRDS). A VSAM data set whose records are loaded into fixed-length slots.

RELEASE command. The DFSMSdss function that releases the unused space in sequential and partitioned data sets for use by other data sets.

RESERVE. A method of serializing DADSM update accesses to the VTOC. It is also a method of serializing processor accesses to a shared DASD volume.

Resource Access Control Facility (RACF). An IBM program product that provides for access control by identifying and verifying users to the system, authorizing access to DASD data sets, logging the detected unauthorized attempts to enter the system, and logging the detected accesses to protected data sets.

RESTORE command. The DFSMSdss function used to recover data sets, tracks, and volumes.

RMID. Replacement module identifier.

RNL. Resource name list.

RRDS. Relative record data set.

RVA. RAMAC Virtual Array.

run-length data compression. Data compression (run-length).

S

SAF. System authorization facility.

SAM. Sequential access method.

scheduler task. A DFSMSdss subtask that interprets and schedules commands.

SCP. System control program.

SEQ. Sequential or sequential processing.

sequential data striping. A software implementation of a disk array that distributes data sets across multiple volumes to improve performance.

SEREP. System environmental recording, editing, and printing

SMF. System management facilities.

SML. MVS Storage Management Library.

SMP. System Modification Program.

SMP/E. System Modification Program/Extended.

SMPE. A cataloged procedure that includes the required DD statements for running SMP/E and is used in the RECEIVE, APPLY, and ACCEPT steps of SMP/E processing.

SMS. Storage Management Subsystem.

SnapShot. A function of the RAMAC Virtual Array (RVA) that allows an instantaneous copy to be made of data sets using DFSMS software.

software support facility (SSF). An IBM online database that allows for storage and retrieval of information about all current APARs and PTFs.

SP. System Product.

sphere. A VSAM cluster with one or more associated alternate indexes and paths. The VSAM cluster (sometimes called the base cluster), alternate indexes, and paths are sometimes referred to as sphere components.

SSF. Software support facility.

Stand-Alone restore. One of two DFSMSdss programs. The Stand-Alone restore program runs independently of the MVS system environment and is limited to one function—a full or partial (tracks) RESTORE from a dump tape.

storage class. A named list of data set storage attributes that identify performance goals and availability requirements, defined by the storage administrator, used to select a device that can meet those goals and requirements.

storage constructs. The group of predefined models (data class, management class, storage class, and storage group) that are used to classify storage management needs and procedures for data sets under the Storage Management Subsystem. Each data set has construct names associated with it by explicit specification or defaulting.

storage group. A named collection of DASD volumes that have been grouped to meet a defined service strategy.

storage management. The task of managing auxiliary storage resources for an installation.

Storage Management Subsystem (SMS). An MVS subsystem that helps automate and centralize the management of storage. To manage storage, the storage management subsystem provides the storage administrator with control over data class, storage class, management class, storage group, and automatic class selection routine definitions.

stripe. In DFSMS, the portion of a striped data set, such as an extended format data set, that resides on one volume. The records in that portion are not always logically consecutive. The system distributes records among the stripes such that the volumes can be read from or written to simultaneously to gain better performance. Whether it is striped is not apparent to the application program.

striped data set. An extended format data set that occupies multiple volumes. A software implementation of sequential data striping.

striping. A software implementation of a disk array that distributes a data set across multiple volumes to improve performance.

subtask. A task initiated and ended by a higher order task.

SVC. Supervisor call instruction.

SVRB. Supervisor request block.

SYSRES. System residence disk

system data. The data sets required by MVS or its subsystems for initialization.

system-managed data set. A data set that has been assigned a storage class.

system-managed storage. Storage managed by the Storage Management Subsystem. SMS attempts to deliver required services for availability, performance, space, and security to applications.

system library. A collection of data sets or files in which the parts of an operating system are stored.

system link library. System library.

System Modification Program (SMP). A program used to install software and software changes on the MVS system.

System Modification Program Extended (SMP/E). An IBM licensed program used to install software and software changes on the MVS system. In addition to

providing the services of SMP, SMP/E consolidates installation data, allows more flexibility in selecting changes to be installed, provides a dialog interface, and supports dynamic allocation of data sets.

T

TCB. Task control block.

Time sharing option (TSO). An option on the operating system for a System/370 that provides interactive time sharing from remote terminals.

TIOT. Task input/output table.

TLIB. Target library.

track packing. A technique used by DFSMSdss that builds target tracks for any DASD device using input physical record information.

TRK. A subparameter of the SPACE parameter in a DD statement. It specifies that space is to be allocated by tracks.

TSO. Time sharing option.

TSO/E. TSO/Extensions.

TTR. Track-track-record.

type-of-failure keyword. In diagnosing program failures, a keyword that identifies the type of program failure that has occurred in DFSMSdss.

U

UACC. Universal access authority.

UCB. Unit control block.

UIM. User interaction module.

unlike devices. Devices that have different track capacities or a different number of tracks per cylinder.

used tracks. Tracks from the beginning of data sets to the last-used track.

user exit. A programming service provided by an IBM software product that may be requested by an application program for the service of transferring control back to the application program upon the later occurrence of a user-specified event.

V

VDRL. Volume restore limits.

VDSS. VTOC/Data Set Services.

virtual concurrent copy. An operation that uses SnapShot to provide a concurrent copy-like function when the source volume supports SnapShot, but not concurrent copy. (Also called CC-compatible SnapShot.)

virtual storage access method (VSAM). An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by the relative-record number.

VM. Virtual machine.

VOLID. Volume ID.

VOLSER. Volume serial number.

volume. The storage space on DASD, tape or optical devices, which is identified by a volume label.

volume backup. Backup of an entire volume to protect against the loss of the volume.

volume header record. The record in the DFSMSdss dump tape that identifies and contains data pertinent to the whole volume, and identifies the type of operation that created a dump.

volume trailer record. The record in the DFSMSdss dump tape that identifies the end of the data for a DASD volume.

VRRDS. A VSAM variable record RRDS.

VSAM. Virtual storage access method.

VSAM volume data set (VVDS). A data set that describes the VSAM and SMS-managed non-VSAM data sets on a volume. The name of the data set is SYS1.VVDS.Vvolser.

VSE. Virtual storage extended.

VTOC. Volume table of contents.

VTOCIX. The data set on which the location of the data set VTOC entries are kept in an index for quick access by DADSM. The name of the data set is SYS1.VTOCIX.Vvolser.

VVDS. VSAM volume data set.

VVR. VSAM volume record.

W

WAIT. In diagnosing program failures, the keyword that identifies DFSMSdss suspended activity, while waiting for some condition to be satisfied. DFSMSdss does not issue a message to tell why it is waiting.

WTO. Write to operator.

X

XA. Extended Architecture.

Z

zFS. See *zSeries File System*.

zSeries File System (zFS). A z/OS UNIX file system that can be used in addition to the hierarchical file system (HFS). zFS stores files in VSAM linear data sets. z/OS provides support for zFS in its Distributed File Service element.

Index

Special characters

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

S

T

U

V

W

Readers' Comments — We'd Like to Hear from You

z/OS
DFSMSdss Storage Administration Guide

Publication No. SC35-0423-06

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

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

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC35-0423-06

IBM®

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

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

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5694-A01

Printed in USA

SC35-0423-06