

z/OS
3.2

DFSMSdss Storage Administration



Note

Before using this information and the product it supports, read the information in [“Notices” on page 651.](#)

This edition applies to IBM® z/OS® 3.2 (5655-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-09-30

© **Copyright International Business Machines Corporation 1984, 2025.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures.....	xv
Tables.....	xvii
About this document.....	xix
Required product knowledge.....	xix
z/OS information.....	xxi
How to provide feedback to IBM.....	xxiii
Summary of changes.....	xxv
Summary of changes for z/OS 3.2.....	xxv
Summary of changes for z/OS 3.1.....	xxv
Part 1. DFSMSdss Storage Administration Guide.....	1
Chapter 1. Introduction to the DFSMSdss component of DFSMS.....	3
Understanding the role of DFSMSdss.....	3
Managing user data with SMS.....	3
Sequential data striping.....	4
Record counting.....	4
Installation exit routines.....	4
Authorization checking.....	5
Managing availability with DFSMSdss.....	5
Backing up and restoring individual UNIX files.....	6
Using DFSMSHsm for backup.....	7
Using concurrent copy.....	7
Using the stand-alone restore program of DFSMSdss.....	8
Using DSS cloud solutions.....	8
Defining an object storage cloud.....	12
Using an object storage cloud.....	14
Managing data movement with DFSMSdss.....	15
Moving data.....	15
Moving data in an SMS-managed environment.....	16
Moving data with concurrent copy.....	16
Moving data with FlashCopy.....	16
Moving data with SnapShot.....	16
Converting data to and from SMS management.....	17
Converting data sets with data movement.....	17
Converting volumes without data movement.....	17
Managing space with DFSMSdss.....	17
JCL examples.....	18
Chapter 2. Requirements for running DFSMSdss.....	19
Understanding the operating environment.....	19
Storage requirements.....	19
Hardware requirements.....	21
Volume formats.....	21

Indexed VTOC.....	22
Data set organizations.....	22
Temporary data set names.....	22
Chapter 3. Logical and physical processing and data set filtering.....	25
UNIX file filtering.....	25
Defining logical and physical processing.....	25
Logical processing.....	25
Physical processing.....	26
Data integrity considerations.....	27
Choosing data sets for processing—filtering.....	27
Filtering by data set names.....	28
Filtering by data set characteristics.....	29
The FILTERDD keyword.....	30
Uses of filtering.....	30
Chapter 4. Invoking DFSMSDss.....	33
Invoking DFSMSDss with ISMF.....	33
How to invoke ISMF.....	33
Invoking DFSMSDss with JCL.....	33
Invoking DFSMSDss with the application interface.....	33
User interaction module exit functions.....	34
Chapter 5. Protecting DFSMSDss functions.....	35
Protecting DFSMSDss and ISMF functions with RACF.....	35
ISMF functions you might want to protect.....	35
Setting up the authorization structure.....	35
Protecting DFSMSDss functions with RACF FACILITY class profiles.....	37
Name-hiding.....	38
Chapter 6. Managing availability with DFSMSDss.....	39
Planning an availability strategy.....	39
Backup and recovery.....	39
Disaster recovery.....	40
Maintaining vital records.....	41
Archiving data sets.....	42
Backing up data sets.....	42
Logical data set dump.....	43
Physical data set dump.....	44
Renaming data sets during dump processing.....	44
Backup with concurrent copy.....	45
Backing up data sets to an object storage cloud.....	47
Using DFSMSDss as a backup utility for CICSVR.....	47
A backup scenario.....	48
Backing up data sets with special requirements.....	48
Dumping HFS data sets.....	49
Dumping zFS data sets.....	49
Dumping multivolume data sets.....	49
Dumping integrated catalog facility user catalogs.....	51
Dumping non-VSAM data sets that have aliases.....	51
Dumping VSAM spheres.....	52
Dumping indexed VSAM data sets.....	52
Dumping SYS1 system data sets.....	52
Dumping data sets containing records past the last-used-block pointer.....	52
Backing up SMS-managed data sets.....	53
Backing up data sets being accessed with record level sharing.....	53
Backing up data sets with extended attributes.....	54
Backing up UNIX files.....	54

Hard links.....	54
Sparse files.....	54
Last backup date.....	54
Backup with CLONE processing.....	55
Backing up volumes.....	55
Logical volume DUMP.....	55
Physical volume dump.....	55
Backing up system volumes.....	56
Backing up VM-format volumes.....	56
Dumping data efficiently.....	56
Combining volume copy and volume dump to reduce your backup window.....	57
Space considerations.....	59
Performance considerations.....	59
Shared DASD considerations.....	63
Backing up and restoring volumes with incremental FlashCopy.....	64
Securing your tape backups.....	66
Using host-based encryption to secure backups.....	67
DFSMSdss processing of dump encryption requests.....	71
Restoring data sets.....	73
Logical data set restore.....	74
DFSMSdss handling of the expiration date during logical restore.....	77
DFSMSdss handling of the data-set-changed indicator during restore	77
Physical data set restore.....	77
Coexistence considerations.....	78
Restoring data sets with special requirements.....	79
Restoring multivolume data sets and restoring data sets using multiple target volumes (spill volumes).....	79
Restoring integrated catalog facility catalogs.....	80
Restoring non-VSAM data sets that have aliases.....	81
Restoring indexed sequential, unmovable, direct, and absolute track data sets.....	81
Restoring an undefined DSORG data set.....	83
Restoring an extended-format VSAM data set with stripe count of one.....	83
Restoring a VSAM sphere.....	83
Restoring a preallocated VSAM cluster.....	84
Restoring the VVDS and the VTOCIX.....	84
Restoring a PDSE.....	84
Restoring a damaged PDS.....	84
Restoring data sets in an SMS-managed environment.....	85
Converting non-VSAM data sets to multivolume.....	85
Restoring SMS-managed data sets.....	86
Restoring GDG data sets.....	88
Restoring non-SMS-managed data sets.....	89
Logical restore of data sets with phantom catalog entries.....	89
Logical restore of preformatted empty VSAM data sets.....	90
Restoring UNIX files.....	90
Determining what files are in a backup.....	90
Restoring to preexisting files.....	91
Hard links.....	91
Last backup date.....	92
Restoring volumes.....	92
Specifying output volumes.....	93
Recovering VM-format volumes.....	94
Coexistence considerations.....	95
Chapter 7. Managing data movement with DFSMSdss.....	97
Preparing for data movement.....	97
Evaluating the use of logical and physical copy.....	97
Controlling what DFSMSdss copies.....	98

Moving data sets.....	98
Moving volumes.....	98
Logical data set copy.....	98
Physical data set copy.....	99
Specifying input volumes.....	100
Selecting output volumes.....	100
Renaming data sets.....	101
Expiration date handling.....	103
SMS to SMS.....	103
SMS to non-SMS.....	103
Non-SMS to SMS.....	104
Non-SMS to non-SMS.....	104
Defining RACF profiles.....	104
Moving data sets with utilities.....	104
Moving data sets with concurrent copy.....	106
Specifying concurrent copy for COPY requests.....	106
Moving data sets with FlashCopy.....	107
Designating FlashCopy usage.....	108
Moving data sets with SnapShot.....	110
Moving data sets with special requirements.....	111
Moving undefined DSORG and empty non-VSAM data sets.....	111
Moving system data sets.....	112
Moving catalogs.....	112
Moving non-VSAM data sets that have aliases.....	113
Moving multivolume data sets.....	113
Converting VSAM and non-VSAM data sets to multivolume.....	114
Moving VSAM data sets.....	114
Moving a PDSE.....	116
Moving a damaged PDS.....	116
Moving unmovable data sets.....	116
Moving data sets to unlike devices.....	117
Moving indexed sequential data sets.....	117
Moving direct access data sets.....	117
Moving GDG data sets.....	117
Moving SMS-managed data sets.....	119
Moving non-SMS-managed data sets.....	121
Moving to preallocated data sets.....	121
Moving data sets being accessed with record level sharing.....	124
Moving preformatted empty VSAM data sets.....	124
VTOC considerations for moving volumes.....	124
Logical volume copy operation.....	125
Physical volume copy operation.....	125
Moving volumes with FlashCopy.....	126
Designating FlashCopy usage.....	126
Determining why FlashCopy cannot be used.....	127
Freeing subsystem resources.....	127
Choosing space efficient FlashCopy	128
Initializing the volume with the FCWITHDRAW keyword.....	129
Backing up volumes with FlashCopy consistency group.....	129
Moving volumes with SnapShot.....	131
Designating SnapShot usage.....	131
Determining why SnapShot cannot be used.....	132
Moving volumes to like devices of equal capacity.....	132
Moving volumes to like devices of greater capacity.....	132
Moving volumes to unlike devices.....	132
Moving VM-format volumes.....	133
Chapter 8. Converting data to and from SMS management.....	135

Evaluating conversion to SMS management.....	135
Data sets ineligible for conversion to SMS.....	135
Data sets ineligible for conversion from SMS.....	136
Volumes eligible for conversion to SMS.....	136
Conversion by data movement.....	136
Converting to SMS management by data movement.....	136
Conversion from SMS management by data movement.....	137
Conversion without data movement.....	137
Simulating conversion.....	137
Preparing a volume for conversion.....	138
Converting to SMS management without data movement.....	138
SMS report.....	139
Special data set requirements for conversion to SMS.....	139
VSAM sphere eligibility.....	139
Multivolume data sets.....	140
GDG data sets.....	140
Temporary data sets.....	141
VTOC and VVDS.....	141
Converting from SMS management without data movement.....	141
Special data set requirements for conversion from SMS.....	141
Multivolume data sets.....	141
GDG data sets.....	142
Temporary data sets.....	142
VTOC and VVDS.....	142
Special considerations for using non-SMS-managed targets.....	142
 Chapter 9. Managing space with DFSMSdss.....	 143
Reclaiming DASD space.....	143
Releasing unused space in data sets.....	143
Compressing a PDS.....	144
Deleting unwanted data sets.....	144
Combining data set extents.....	145
Consolidating free space and extents on volumes.....	146
When to run DEFRAG and CONSOLIDATE functions.....	146
Designating FlashCopy usage.....	147
Preserve Mirror FlashCopy.....	147
Determining why FlashCopy cannot be used.....	148
Designating SnapShot usage.....	148
Determining why SnapShot cannot be used.....	149
Data sets excluded from DEFRAG or CONSOLIDATE processing.....	149
DEFRAG options.....	149
Serialization.....	151
Security considerations.....	153
Maximizing track utilization by reblocking data sets.....	153
 Chapter 10. Diagnosing problems in DFSMSdss operations.....	 155
Determining the source of the failure: DFSMSdfp, DFSMSdss, or DFSMSHsm.....	155
Using keywords to identify the problem.....	156
Component identification keyword.....	156
Release-level keyword.....	156
Type-of-failure and function keywords.....	157
Module keyword.....	161
Maintenance-level keyword.....	163
Using the IBM Support Center.....	163
Using the software support facility.....	164
Using IBMLink/ServiceLink.....	164
Info/System.....	164

Chapter 11. ACS routine information.....	165
ACS variables available during Copy function.....	165
ACS variables available during RESTORE and CONVERTV processing.....	166
Using SIZE and MAXSIZE variables.....	167
Chapter 12. Dumping and restoring Linux for IBM Z partitions and volumes.....	169
Preparing to work with Linux volumes.....	169
Understanding the hardware environment.....	169
Choosing VOLSERS for Linux volumes.....	170
Formatting and partitioning Linux volumes.....	170
Obtaining authorization for Linux volumes.....	171
Backing up a Linux volume with partitions.....	171
Using DFSMSdss dump and restore commands.....	172
Example 1. DUMP FULL.....	172
Example 2. DUMP FULL with CONCURRENT COPY.....	173
Example 3. DUMP DATASET.....	173
Example 4. COPY FULL.....	174
Example 5. COPY FULL COPYVOLID ALLEXCP.....	174
Example 6. RESTORE FULL.....	175
Example 7. RESTORE DATASET.....	175
Example 8. COPYDUMP.....	177
Submitting JCL batch jobs to a z/OS system using FTP.....	177
Using DFSMSdss stand-alone services.....	177
Chapter 13. Format of the DFSMSdss dump data set.....	179
Data set and volume backup format.....	179
ADRBMB data area.....	180
ADRBMB constants.....	180
ADRBMB cross-reference.....	181
ADRTAPB data area.....	181
ADRTAPB constants.....	191
ADRTAPB cross-reference.....	192
z/OS UNIX file format.....	198
ADRTAPB version 2 data area.....	198
ADRTAPB constants.....	207
ADRTAPB cross-reference.....	207
Chapter 14. DFSMSdss patch area.....	213
Using RESET with CLONE.....	213
Sample JCL.....	213
Forcing the use of preallocated VSAM data sets (PN04574).....	214
Ignoring VSAM duplicate key errors (PN05529).....	214
Modifying the timeout period for enqueue lockout detection (PL84514).....	215
Controlling the wait/retry time for serialization of system resources (PN11523).....	215
Using CONVERTV on data sets with a revoked user ID in the RESOWNER field (OY59957).....	216
Restoring inconsistent PDSE data sets (OY60301).....	216
Changing default protection status during RESTORE (PN37489).....	217
Overwriting existing objects during logical data set DUMP to an object store cloud.....	217
Restoring or copying undefined, multivolume SMS-managed data sets (OY63818).....	218
Bypassing backup-while-open processing (OY63531).....	218
Bypassing storage and management class authorization checking during RESTORE (OY65348).....	218
Issuing notification for tape and migrated data sets (OY66092).....	219
Using RESET with concurrent copy (OY65555).....	219
Forcing RESTORE after message ADR482E (OY67532).....	220
Restoring VSAM KSDS or VRRDS after messages ADR789W, ADR364W, and ADR417W (OY67942).....	220
Restoring VSAM data sets with expiration date of 1999365 (OW00780).....	220

Restoring VSAM data sets with expiration dates beyond 2000 (OW00780).....	221
Changing default insertion of EOF track during COPY with ALLDATA specified (OW15003).....	221
Using RESET or UNCATALOG in a logical data set dump (PN60114).....	222
Changing secondary allocation quantity in format 1 DSCB for PDSE data sets (OW07755).....	222
Changing reference date default settings during data set COPY and RESTORE processing (OW12011).....	223
Changing default protection processing during COPY (OW10314).....	224
Bypassing management and storage class access checks during COPY (PN72592).....	224
Changing default handling of invalid tracks created during data set COPY and RESTORE processing (OW08174).....	225
Forcing RESTORE to the same volumes as the source VSAM data set (OW07077).....	225
Modifying number of volumes allocated for SMS data sets during logical RESTORE and COPY (OW15880).....	226
Dumping a keyed VSAM data set that has data CAs without corresponding index CIs (OW17877).....	227
Changing the default DEFRAG processing of checkpointed data sets (OW20285).....	227
Setting the percentage to overallocate target data set space (OW27837).....	228
Bypassing RLS processing (OW32817).....	228
Changing creation date default settings during data set COPY and RESTORE (OW19618).....	229
Copying and dumping a PDSE data set using the VALIDATE PDSE option (OW48074).....	229
Changing the default maximum number of active parallel subtasks.....	230
Changing the default initialization processing during DUMP with FCWITHDRAW (OA18929).....	230
Changing the default DEFRAG processing of LINKLIST-indicated data sets (OW43874).....	231
Changing the FASTREPLICATION default setting during Copy and Defrag (OA11637).....	232
Tuning hardware assisted compression (OA13300).....	232
Resetting the data-set-changed indicator during physical full or partial RESTORE operation (OA20907).....	233
Requesting that DFSMSdss double-check data set high used RBA values for LDS data sets.....	233
Enabling or disabling use of the catalog search interface for data set name filtering.....	234
Requesting that DFSMSdss restore the VM-formatted volume that was DUMPed by z/OS V1R10 before OA27531 was applied.....	234
Adding timestamps to messages.....	234
Enabling building appropriate channel programs.....	235
Requesting that DFSMSdss attempt to fix ESDSes with corrupted RDFs.....	235
ADRPTCHB data area.....	236
ADRPTCHB cross-reference.....	239
Overriding the EATTR attribute during logical COPY.....	240
Overriding the ADR310W warning message during full volume DUMP.....	241
Restore an extended-format data set that was dumped with the ZCOMPRESS keyword when zEDC services were not used.....	241
Omitting the ADR826W warning message when the STORGRP keyword is specified	242
Check for a CDA Provider file regardless of the CDAPROVIDERFILE keyword's presence.....	242
Omit specified warning message.....	242

Part 2. DFSMSdss Storage Administration Reference..... 243

Chapter 15. Specifying DFSMSdss commands.....	245
Command syntax.....	245
How many subkeywords are allowed?.....	246
Specifying subkeywords in a command data set.....	246
How to read syntax diagrams.....	246
JCL that you need.....	248
How to control DFSMSdss through PARM information in the EXEC statement.....	249
Examples of invoking DFSMSdss with JCL.....	252
Chapter 16. DFSMSdss filtering—choosing the data sets you want processed.....	255
How DFSMSdss filters data sets.....	255

Virtual storage access method (VSAM) data set considerations.....	255
Filtering by data set names.....	256
Using an asterisk in partially qualified data set names.....	256
Using a percent sign in partially qualified data set names.....	256
Examples of fully and partially qualified data set names.....	256
Relative generation filtering.....	257
Filtering by data set characteristics.....	258
Some examples of the BY keywords.....	263
Standard catalog search order.....	263
Broken data set considerations.....	263
Chapter 17. DFSMSdss filtering—choosing the z/OS UNIX files you want processed.....	265
Path name considerations.....	265
UNIX wildcard characters.....	266
Chapter 18. Syntax - DFSMSdss function commands.....	267
What DFSMSdss commands do.....	267
Building the stand-alone IPL-able core image.....	267
Using DUMP and RESTORE for backup and recovery.....	267
Moving data with COPY.....	267
Converting to and from Storage Management Subsystem (SMS) with CONVERTV.....	268
Managing space with COMPRESS, CONSOLIDATE, DEFRAG, and RELEASE.....	268
Using COPY for partitioned data set (PDS) and partitioned data set extended (PDSE) conversions.....	268
Copying DFSMSdss-produced dump data with COPYDUMP.....	268
Printing for diagnostic purposes with PRINT.....	268
Managing extent space efficient volume space with SPACEREL.....	269
BUILDSA command for DFSMSdss.....	269
BUILDSA syntax.....	270
Explanation of BUILDSA command keywords.....	270
BUILDSA command examples.....	272
CGCREATED command for DFSMSdss.....	274
CGCREATED syntax.....	274
Explanation of CGCREATED command keywords.....	275
CLOUDUTILS command for DFSMSdss.....	275
CLOUDUTILS syntax.....	275
Explanation of CLOUDUTILS command keywords.....	276
Examples of CLOUDUTILS operations.....	280
Delete a DFSMSdss dump data set.....	280
COMPRESS command for DFSMSdss.....	281
COMPRESS syntax.....	281
Explanation of COMPRESS command keywords.....	282
Example of compress operations.....	285
CONSOLIDATE command for DFSMSdss.....	286
CONSOLIDATE command syntax.....	286
Explanation of CONSOLIDATE command keywords.....	287
Example of a CONSOLIDATE operation.....	295
CONVERTV command for DFSMSdss.....	295
CONVERTV command syntax.....	295
Explanation of CONVERTV command keywords.....	296
Examples of CONVERTV operations.....	299
COPY command for DFSMSdss.....	300
Special considerations for COPY.....	301
COPY DATASET Command Syntax for Logical Data Set.....	301
COPY DATASET Command Syntax for Physical Data Set.....	308
COPY FULL and COPY TRACKS syntax.....	311
Explanation of COPY Command Keywords.....	315
Data Integrity Considerations for Full or Tracks Copy Operation.....	360

Examples of Full and Tracks Copy Operations.....	360
Examples of Data Set Copy Operations.....	361
ALLDATA and ALLEXCP Interactions.....	365
COPYDUMP command for DFSMSdss.....	369
COPYDUMP syntax.....	370
Explanation of COPYDUMP command keywords.....	370
Examples of COPYDUMP operations.....	370
DEFRAG command for DFSMSdss.....	371
DEFRAG syntax.....	373
Explanation of DEFRAG command keywords.....	374
Examples of DEFRAG operations.....	383
Results of a successful DEFRAG operation.....	384
DUMP command for DFSMSdss.....	386
Special considerations for dump.....	387
DUMP FULL command syntax.....	387
DUMP TRACKS command syntax.....	390
DUMP DATASET syntax for logical data set.....	392
DUMP DATASET syntax for physical data set.....	395
DUMP PATH syntax.....	397
Explanation of DUMP command keywords.....	397
Data integrity considerations for full or tracks dump operation.....	426
Format of the output data set.....	426
Examples of full and tracks dump operations.....	426
Examples of physical data set dump operations.....	427
Examples of logical data set dump operations.....	429
Examples of file dump operation.....	432
PRINT command for DFSMSdss.....	432
PRINT syntax.....	433
Explanation of PRINT command keywords.....	434
Examples of print operations.....	439
RELEASE command for DFSMSdss.....	441
RELEASE syntax for physical processing.....	442
RELEASE syntax for logical processing.....	442
Explanation of RELEASE command keywords.....	444
Example of a release operation.....	451
RESTORE command for DFSMSdss.....	451
Special considerations for RESTORE.....	452
Data integrity considerations for full or tracks restore operations.....	453
RESTORE FULL command syntax.....	454
RESTORE TRACKS command syntax.....	455
RESTORE DATASET command syntax for logical data set.....	456
RESTORE DATASET command syntax for physical data set.....	460
RESTORE PATH command syntax.....	462
Explanation of RESTORE command keywords.....	462
DFSMSdss RESTORE process.....	494
Examples of full and tracks restore operations.....	497
Examples of physical data set restore operations.....	498
Examples of logical data set restore operations.....	500
SPACEREL command for DFSMSdss.....	502
SPACEREL syntax.....	503
Explanation of SPACEREL command keywords.....	503
Chapter 19. Syntax—auxiliary commands.....	507
Writing to the operator for DFSMSdss.....	507
WTO command.....	507
WTOR command for DFSMSdss.....	507
Scheduling tasks.....	508
SERIAL command for DFSMSdss.....	508

PARALLEL command for DFSMSdss.....	508
Controlling task processing.....	508
SET command—setting condition codes and patch bytes.....	508
IF-THEN-ELSE command sequence for DFSMSdss—using condition codes.....	510
EOJ command—ending your DFSMSdss step.....	514
Chapter 20. DFSMSdss Utilities	515
DFSMSdss stand-alone services	515
Preparing to run the stand-alone services program.....	515
IPLing and running the Stand-Alone Services Program.....	519
Building the IPL-able core image.....	529
DFSMSdss Compression Tool.....	530
Control.....	530
Example of invoking the compression tool.....	531
Chapter 21. Data security and authorization checking.....	533
Effects of SPECIAL, OPERATIONS, and DASDVOL.....	533
SPECIAL.....	533
OPERATIONS.....	534
DASDVOL.....	534
General data security information.....	535
Protecting resources and data sets.....	535
Protecting the usage of DFSMSdss.....	535
Password protection.....	536
Protected user and group data sets.....	537
Generic and discrete profile considerations.....	537
Security-level, category, and label checking.....	539
Protect-all and always-call.....	539
Standard naming conventions.....	539
DFSMSdss temporary data set names.....	539
Discretely protected multivolume data set.....	541
Erase-on-scratch.....	541
SMS-managed data set protection.....	541
Logging.....	542
DFSMSdss storage administrator.....	542
ADMINISTRATOR keyword.....	542
FACILITY class profiles for the ADMINISTRATOR keyword.....	543
DFSMSdss access authority.....	544
Volume access and DASDVOL.....	544
Data set access authorization levels.....	547
Protected catalogs.....	547
Non-SMS versus SMS authorization.....	547
System operator authorization, special data set types.....	548
Access authorization for DFSMSdss commands.....	548
CGCREATED.....	548
CLOUDUTILS.....	549
COMPRESS.....	549
CONSOLIDATE.....	549
CONVERTV.....	549
COPY.....	549
COPYDUMP.....	554
DEFRAG.....	554
DUMP.....	554
PRINT.....	555
RELEASE.....	555
RESTORE.....	555
Chapter 22. Data integrity—serialization.....	561

Volume serialization.....	562
Avoiding lockout.....	562
The WAIT option.....	563
Data set serialization.....	563
Enqueuing—ENQ.....	563
Dumping HFS data sets.....	564
zFS data sets.....	565
Dynamic allocation (DYNALLOC).....	566
Enqueuing versus dynamic allocation of data sets.....	566
Read/Write serialization scheme.....	566
WAIT option.....	569
An example of RESERVE-ENQUEUE processing.....	570
Backup-while-open data sets (CICS and DFSMStvs).....	570
Backup-while-open status definition.....	572
Backup-while-open processing.....	572
Backup-while-open and concurrent copy.....	573
TOLERATE (ENQFAILURE) and SHARE considerations.....	574
CICS recovery data.....	574
Backup-while-open data sets (IMS).....	574
Container serialization.....	575
Object serialization.....	575
 Chapter 23. Application programming interface.....	 577
Calling block structure.....	577
User interactions.....	579
Service considerations.....	580
Cross-memory Application Interface overview.....	580
Using the cross memory application interface to control DFSMSdss.....	582
Cross-memory application return codes.....	583
System programming information.....	584
Application interface blocks.....	584
Exit identification block.....	585
Application programming interface restrictions.....	587
Cross-memory application interface restrictions.....	588
User interaction module exit option descriptions.....	589
Function startup (Eioption 00).....	589
Reading SYSIN record (Eioption 01).....	590
Printing SYSPRINT record (Eioption 02).....	590
Reading physical tape record (Eioption 03).....	591
Reading logical tape record (Eioption 04).....	591
Writing logical tape record (Eioption 05).....	591
Writing physical tape record (Eioption 06).....	592
Reading disk track (Eioption 07).....	592
Writing disk track (Eioption 08).....	592
Reading utility SYSPRINT (Eioption 09).....	592
Writing SYSPRINT record (Eioption 10).....	593
Writing WTO message (Eioption 11).....	593
Writing WTOR message (Eioption 12).....	593
Presenting ADRUFO record (Eioption 13).....	594
Function ending (Eioption 14).....	594
Presenting WTOR response (Eioption 15).....	594
OPEN/EOV tape volume security and verification exit (Eioption 16).....	594
OPEN/EOV nonspecific tape volume mount (Eioption 17).....	595
Insert logical VSAM record during restore (Eioption 18).....	595
Output tape I/O error (Eioption 19).....	595
Volume notification (Eioption 20).....	595
Data set verification (Eioption 21).....	596
Bypass verification exit (Eioption 22).....	596

Data set processed notification exit (Eioption 23).....	599
Concurrent copy initialization complete (Eioption 24).....	601
Backspace physical tape record (Eioption 25).....	602
Dump volume output notification (Eioption 26).....	603
Physical data set processed notification exit (Eioption 27).....	603
Target data set allocation notification exit (Eioption 28).....	604
Physical data set volume allocation notification exit (Eioption 30).....	604
Store application metadata object (Eioption 31).....	605
Retrieve application metadata object (Eioption 32).....	605
Output object notification exit (Eioption 33).....	606
Cloud bypass verification (Eioption 35).....	606
z/OS UNIX file path notification (Eioption 41).....	606
z/OS UNIX file path bypass notification (Eioption 42).....	607
z/OS UNIX file path processed notification (Eioption 43).....	607
z/OS UNIX file path clone initialization notification (Eioption 44).....	608
Avoiding lockout.....	608
Application interface summary.....	608
ADREID0 data area.....	609
Constants for ADREID0.....	619
Cross reference for ADREID0.....	619
Example: invoking DFSMSdss by using an application program.....	624
How to determine DFSMSdss version, release, and modification level.....	625
Chapter 24. Examples of the application program with the user interaction module (UIM).....	627
Chapter 25. Data set attributes.....	641
Chapter 26. z/OS UNIX file attributes.....	645
Appendix A. Coexistence considerations.....	647
Restoring backups using DFSMSdss.....	647
Appendix B. Accessibility.....	649
Notices.....	651
Terms and conditions for product documentation.....	652
IBM Online Privacy Statement.....	653
Policy for unsupported hardware.....	653
Minimum supported hardware.....	653
Programming interface information.....	654
Trademarks.....	654
Glossary.....	655
Index.....	673

Figures

1. TCT data movement.....	9
2. TCT data movement using a provider file to a non-TS7700 cloud.....	11
3. Data movement during Direct to Cloud solution.....	11
4. Output from a Dump of an Integrated Catalog Facility User Catalog.....	51
5. Output from Restore of Integrated Catalog Facility User Catalog.....	81
6. SMS Report.....	139
7. Sample JCL for dumping the contents of a volume.....	172
8. Sample JCL for dumping two or more output tapes at the same time.....	173
9. Sample JCL for DUMP FULL with CONCURRENT COPY.....	173
10. Sample JCL for DUMP DATASET.....	174
11. Sample JCL for dumping all of the Linux partitions.....	174
12. Sample JCL for making a full volume copy of a volume.....	174
13. Sample JCL for creating a backup copy of a Linux volume.....	175
14. Sample JCL for restoring a full volume from a DFSMSdss dump.....	175
15. Sample JCL for restoring individual partitions or data sets.....	176
16. Sample JCL for renaming data sets to be restored.....	176
17. Sample JCL for restoring only one partition of a volume.....	177
18. Sample JCL for copying Linux volume dumps.....	177
19. Printed Output Resulting from a Successful DEFRAG Run on nonEAV:.....	385
20. A Section of the Printed Output Resulting from a Successful DEFRAG Run on EAV:.....	385
21. Output Resulting from a PRINT Command.....	441
22. Restore Actions on Non-VSAM Data Sets.....	495
23. DFSMSdss Target Class Selection.....	496

24. Stand-Alone Services Restore Process Overview.....	519
25. DFSMSdss Data Security Decisions.....	545
26. Block Diagram for Backup-While-Open Serialization.....	571
27. DFSMSdss Application Interface Structure.....	579
28. DFSMSdss Exit Interface Structure.....	584
29. The application program process.....	627

Tables

1. Minimum storage requirements for DFSMSdss operations with I/O Buffers below 16 megabyte virtual.....	19
2. Minimum storage requirements for DFSMSdss operations with I/O Buffers above 16 megabyte virtual.....	20
3. Minimum storage requirements for a Restore to an unlike device.....	20
4. Module Names for DFSMSdss/ISMF Line Operators.....	36
5. Module Names for DFSMSdss/ISMF Data Set Application Commands.....	36
6. RACF FACILITY Class Profile Names for DFSMSdss Keywords.....	37
7. RSA private tokens and required cryptographic hardware for decryption.....	70
8. DFSMSdss processing of dump encryption requests.....	72
9. DFSMSdss handling of the data-set-changed indicator during data set restore operations.....	77
10. Last Backup Date Handling.....	92
11. Data Mover Selection Matrix for Data Set Copy.....	105
12. Default Situation for DFSMSdss to Allocate the SMS-Managed GDG Data Set.....	118
13. Non-VSAM Data Set Erase Table with z/OS Security Server (RACF element).....	153
14. Summary of Type-of-Failure Keywords.....	157
15. Variables Passed to ACS Routines during DFSMSdss Copy Function.....	165
16. Variables Passed to ACS Routines during DFSMSdss Restore and CONVERTV Processing.....	166
17. ADRBMB Mapping Macro.....	180
18. ADRBMB Mapping Macro.....	180
19. ADRTAPB Mapping Macro.....	181
20. ADRTAPB Mapping Macro.....	191
21. ADRPTCHB-Mapping Macro.....	236
22. RACF Resources to Restrict Use of This Patch Byte.....	241

23. BY Keywords.....	259
24. ALLDATA and ALLEXCP Interactions When Copying to LIKE Device.....	366
25. ALLDATA and ALLEXCP Interactions When Copying to UNLIKE Device.....	368
26. Physical Data Set Restore Actions on SMS-Managed Data Sets.....	494
27. Stand-Alone Services Options when Using an IBM Supported Tape Library.....	517
28. DFSMSdss FACILITY Class Profiles.....	535
29. DASDVOL Access Authority.....	546
30. Access Authority for the DFSMSdss COPY Command.....	550
31. DFSMSdss COPY Command and RACF Profiles.....	551
32. DFSMSdss Copy and Define Discrete Profile Summary.....	552
33. Access Authority for the DFSMSdss DUMP Command.....	555
34. Access Authority for the DFSMSdss RESTORE Command.....	556
35. DFSMSdss Copy and Define Profile Summary.....	558
36. Level of access obtained by DFSMSdss.....	561
37. Data Set Enqueue Options for Non-VSAM Data Sets, except for Poses that are targets of a COPY or RESTORE operation, Specified on DFSMSdss Commands.....	564
38. Read/Write Access Serialization Scheme.....	566
39. Resource Serialization.....	568
40. DFSMSdss Support for IMS Backup-While-Open Data Sets.....	575
41. Cross-memory application return codes.....	583
42. Data Set Attributes and How They Are Determined.....	641
43. Attributes of the restored files.....	645

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 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. Verify that all of the parameters specified for each command are used correctly; see [Part 2, “DFSMSdss Storage Administration Reference,”](#) on page 243.
2. Correct any errors you might find 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: Refer to the following for more information.

- For descriptions and syntax of the DFSMSdss commands, [Chapter 18, “Syntax - DFSMSdss function commands,”](#) on page 267.
- For information about DFSMSdss messages, [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 B, “Accessibility,”](#) on page 649.

Required product knowledge

To use this document effectively, you should be familiar with:

- DFSMSdfp
- DFSMSHsm
- job control language (JCL)
- RACF (a component of Security Server for z/OS)
- IBM Support.

Readers of this publication are presumed to have a background in programming—especially programming with TSO commands—and in z/OS concepts and terms. This book is written primarily for the system programmer and storage administrator, both of whom must understand the information in [z/OS DFSMS Introduction](#) before reading this publication.

z/OS information

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. For more information, see [How to send feedback to IBM](#).

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Note: IBM z/OS policy for the integration of service information into the z/OS product documentation library is documented on the z/OS Internet Library under [IBM z/OS Product Documentation Update Policy](http://www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy) (www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy).

Summary of changes for z/OS 3.2

This information contains no technical changes for this release.

New

The following content is new.

September 2025 release

- None.

Changed

The following content is changed.

September 2025 release

- Information about FIXCAT is added to [“Using IBMLink/ServiceLink”](#) on page 164.

Deleted

The following content is deleted.

September 2025 release

- Information about preventive service planning (PSP) buckets is deleted because PSP buckets for many IBM products, including z/OS 2.5 and 3.1, are no longer updated. For more information, see the following IBM Support document: [PSP bucket information for IBM Z products](http://www.ibm.com/support/pages/node/7127792) (www.ibm.com/support/pages/node/7127792).

Summary of changes for z/OS 3.1

The following content is new, changed, or no longer included in z/OS 3.1.

New

The following content is new.

June 2025 release

- Added omit specified warning message to the ADRPTCHB data area. For more information, see [Table 21 on page 236](#) and [“Omit specified warning message” on page 242](#). (APAR OA66953, which also applies to z/OS 2.5).

March 2025 release

- Added CDACOMPRESS. For more information, see [“CDACOMPRESS” on page 401](#).
- Added Using DSS cloud solutions. For more information see, [“Using DSS cloud solutions” on page 8](#).

September 2024 release

Added DFSMSdss Compression Tool. For more information see, [“Example of invoking the compression tool”](#) on page 531, [“DFSMSdss Compression Tool”](#) on page 530 and [“Control”](#) on page 530.

future refresh

- The following information is new for APAR OA63892 (applies to z/OS V2R5 and newer).
 - [“Check for a CDA Provider file regardless of the CDAPROVIDERFILE keyword's presence”](#) on page 242 is a new DFSMSdss patch area.
 - [“CLOUDUTILS command for DFSMSdss”](#) on page 275 has a new section for keyword [“CDAPROVIDERFILE”](#) on page 277.
 - [“DUMP command for DFSMSdss”](#) on page 386 has a new section for keyword [“CDAPROVIDERFILE”](#) on page 401.
 - [“RESTORE command for DFSMSdss”](#) on page 451 has a new section for keyword [“CDAPROVIDERFILE”](#) on page 466.

September 2023 release

- Updated [“Omitting the ADR826W warning message when the STORGRP keyword is specified ”](#) on page 242 (APAR OA64567, which also applies to z/OS 2.4 and 2.5).

Changed

The following content is changed.

April 2025 release

- DFSMSdss Data Security Decisions image is updated. For more information, see [“Volume access and DASDVOL”](#) on page 544.

March 2025 release

- Updated CLOUD. For more information , see [“CLOUD”](#) on page 276
- Updated CDAPROVIDERFILE. For more information , see [“CDAPROVIDERFILE”](#) on page 277
- Updated CONTAINER. For more information , see [“CONTAINER”](#) on page 277
- Updated DEBUG. For more information , see [“DEBUG”](#) on page 279
- Updated CLOUD. For more information , see [“CLOUD”](#) on page 400
- Updated CDAPROVIDERFILE. For more information , see [“CDAPROVIDERFILE”](#) on page 401
- Updated CONTAINER. For more information , see [“CONTAINER”](#) on page 402
- Updated DEBUG. For more information , see [“DEBUG”](#) on page 403
- Updated OBJECTPREFIX. For more information , see [“OBJECTPREFIX”](#) on page 402
- Updated CLOUD. For more information , see [“CLOUD”](#) on page 465
- Updated CDAPROVIDERFILE. For more information , see [“CDAPROVIDERFILE”](#) on page 466
- Updated CLOUDCREDENTIALS. For more information , see [“CLOUDCREDENTIALS”](#) on page 466
- Updated CONTAINER. For more information , see [“CONTAINER”](#) on page 467
- Updated DEBUG. For more information , see [“DEBUG”](#) on page 469

February 2025 release

- Updated Table. For more information , see [“ACS variables available during Copy function”](#) on page 165

future refresh

- The following information is updated for APAR [APAR OA63892 \(www.ibm.com/support/pages/apar/OA63892\)](#) (applies to z/OS V2R5 and newer).
 - [“Storage requirements”](#) on page 19.

- “ADRPTCHB data area” on page 236 includes new offsets 91, 92, and 94-98.
- “Physical volume dump” on page 55 is updated with new section [“Physical volume dump to an object storage cloud”](#) on page 56.
- “CLOUDUTILS command for DFSMSdss” on page 275 and the following subsections are updated to include information for the new CDAPROVIDERFILE keyword.
 - [“CLOUDUTILS syntax”](#) on page 275
 - [“CLOUD”](#) on page 276
 - [“CONTAINER”](#) on page 277
 - [“LIST”](#) on page 278
- “DUMP command for DFSMSdss” on page 386 and the following subsections are updated to include information for the new CDAPROVIDERFILE keyword.
 - [“DUMP FULL command syntax”](#) on page 387
 - [“DUMP DATASET syntax for logical data set”](#) on page 392
 - [“CLOUD”](#) on page 400
 - [“CONTAINER”](#) on page 402
- “RESTORE command for DFSMSdss” on page 451 and the following subsections are updated to include information for the new CDAPROVIDERFILE keyword.
 - [“RESTORE FULL command syntax”](#) on page 454
 - [“RESTORE DATASET command syntax for logical data set”](#) on page 456
 - [“CLOUD”](#) on page 400
 - [“CONTAINER”](#) on page 467

March 2024 release

- Specifying TOLERATE(ENQFAILURE) is updated. For more information, see [“Enqueuing—ENQ”](#) on page 563.
- Updated Using the cross memory application interface to control DFSMSdss. For more information, see [“Using the cross memory application interface to control DFSMSdss”](#) on page 582 (APAR OA65569: DFSMSDSS SERVER LAUNCHED BY JCL DOES NOT SHUTDOWN IN 1 MINUTE (www.ibm.com/support/pages/apar/OA65569), which also applies to z/OS 2.5).

February 2024 release

- Updated the first bullet in the list for Restrictions for the Copy command. For more information, see [“Restrictions for the COPY command”](#) on page 115.

Part 1. DFSMSdss Storage Administration Guide

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 z/OS environment.

You can use DFSMSdss to do the following tasks:

- Copy and move data sets between volumes of like and unlike device types. Like devices have the same track capacity and number of tracks per cylinder (for example, 3380 Model D, Model E, and Model K). In contrast, unlike DASD have different track capacities (for example, 3380 and 3390), a different number of tracks per cylinder, or both.
- Dump and restore z/OS UNIX files, 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, or consolidating data set extents.

Understanding the role of DFSMSdss

To understand the role of DFSMSdss in an SMS environment, you need a basic understanding of SMS. Also, how you use DFSMSdss depends on which other DFSMS components are in use at your site, such as the DFSMSHsm component.

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 actions 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 more information about SMS, see the ACS routine information in [z/OS DFSMSdss Storage Administration](#).

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.

Note:

1. DFSMSdss treats a striped data set in the same way as it does other multivolume SMS data sets.
2. 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 more information about striped data sets, 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.

Note: DFSMSdss does not perform record counting for backups to or recoveries from an object storage cloud.

Related reading: For more information about these 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 enqueueing 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 more information about these exit routines, see [z/OS DFSMS Installation Exits](#).

Authorization checking

Related reading: For information about authorization checking, see [Chapter 21, “Data security and authorization checking,”](#) on page 533.

Managing availability with DFSMSdss

DFSMSdss availability management consists of backing up data on DASD and restoring from the backup if the original is lost, damaged, or inadvertently changed. Backups can be directed to DASD, tape, or an object storage cloud.

There are several forms of backup:

Data set backup

Protects against the loss of individual data sets.

Volume backup

Protects against the loss of a volume.

File backup

Protects against the loss of individual files.

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.

Use file level backups to simplify the recovery of files within your z/OS UNIX environment. Without file level backup, recovery of an individual file requires the entire file system to be backed up. At recovery time, the entire file system is restored to a different mount point and the file to be recovered is then copied back into the production file system. File level dump and restore allows you to recover an individual file without having to restore the entire file system first.

Note:

1. If a large format data set is used as the output of a DUMP command, it cannot be used as input to a RESTORE command on a system that is prior to z/OS V1R7.
2. If an extended format data set is used as the output of a DUMP command, it cannot be used as input to a RESTORE command on a system that is prior to z/OS V1R12.
3. Only logical data set backups can be directed to an object storage cloud.
4. Encryption eligible sequential basic and/or large format data sets cannot be used as output of the **DUMP** command.

Backing up and restoring individual UNIX files

You can use the DFSMSDss DUMP command to back up individual z/OS UNIX files and the DFSMSDss RESTORE command to recover them. The DFSMSDss PATH keyword identifies that z/OS UNIX files are to be processed.

z/OS UNIX System Services supports these types of files:

- Regular files
- Directories (special files with list of regular files and other directories)
- Character special file
 - A terminal
 - The default controlling terminal for a process (/dev/tty)
 - A null file
 - A zero file
 - Random number files
 - File descriptor file
 - System console file
 - A UNIX domain socket name file
 - A Communications Server remote tty file
 - The Communications Server character special file
- FIFO special file
- Symbolic links
- Sockets

DFSMSDss uses the term *file* to refer to any of the z/OS UNIX file types described and only refers to a specific file type when the context requires it.

Virtual File System (VFS) server

DFSMSDss processes files within z/OS UNIX System Services (z/OS UNIX) by behaving as a virtual file system (VFS) server. A VFS server uses the z/OS UNIX VFS callable services application programming interface to issue file requests. Those requests are routed to the logical file system (LFS) to the appropriate physical file system (PFS).

DFSMSDss can only process files that are available to z/OS UNIX.

Supported file systems

Only UNIX files residing within a mounted zFS are supported. Any UNIX files that are encountered within a non-zFS file system are rejected for processing because they are not supported by DFSMSDss.

Regular files

Paths that resolve to regular files are processed by DFSMSDss.

Sparse files are regular files that have an apparent size that is larger than the occupied disk size. Blocks within the file have no user data (binary zeros) and are not written to disk. The goal of a backup and recovery application is to identify and maintain a sparse file as being sparse. DFSMSDss supports sparse files for backup and restore and maintains the occupied space of the file when CLONE keyword processing is used to backup the data.

Hard links are regular files that have other names for the same data, these files have a link count greater than one. DFSMSDss backs up each instance of the linked file. Upon restore, separate copies of the same data are not created. The first file encountered is restored and any subsequent files are linked to that original file, as long as both are restored in the same invocation.

Directory files

DFSMSDss processes all directories that are specified in a path.

DFSMSDss only processes a directory file's attributes and ACLs. Other files within the directory or sub-directories and their files are not processed (recursion is not supported).

Character special files

DFSMSDss does not support processing character special files. If encountered, DFSMSDss issues an error message indicating that it is not a supported file type.

FIFO special files

Paths that resolve to FIFO special files are processed.

DFSMSDss only processes the file's attributes and ACLs.

Symbolic links

Paths that resolve to symbolic links are processed by DFSMSDss.

When processing a symbolic link, DFSMSDss only dumps and restores the reference to the file. The actual file or directory (known as the target of the symbolic link) that is referenced is not processed and is not checked to see if it exists.

Sockets

DFSMSDss does not support processing socket files. If encountered, DFSMSDss issues an error message indicating that it is not a supported file type.

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 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 databases 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 database 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 database or any collection of data at a point in time and with minimum down time for the database. The database is unavailable only long enough for DFSMSDss to initialize a concurrent copy 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 concurrent copy 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 concurrent copy, this exposure exists for the entire duration of the dump. With concurrent copy, the exposure exists only during initialization.

Note:

1. If you are using concurrent copy 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.
3. Concurrent copy and virtual concurrent copy are not supported for backups directed to an object storage cloud.

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

If a concurrent copy operation fails after signaling that the concurrent copy 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 concurrent copy operation was started. This is because the data may have been updated while the copy operation was progressing.

Virtual concurrent copy

DFSMSDss uses virtual concurrent copy (VCC) to provide a concurrent copy-like function when the source device supports data set FlashCopy® or SnapShot.

During virtual concurrent copy, data is "flushed" or "snapped" from the source location to an intermediate location, and the data is gradually copied to the target location through standard I/O methods. The operation is logically complete after the source data is "flushed" or "snapped" to the intermediate location and physically complete after the data is moved to the target media.

The working space data set is used as an intermediate location for virtual concurrent copy. For more information, see ["Virtual concurrent copy working space" on page 61](#).

Using the stand-alone restore program of DFSMSDss

DFSMSDss stand-alone restore is a single-purpose program. It is designed to help you restore vital system packs during disaster recovery without relying on a z/OS 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 more information about the DFSMSDss stand-alone restore program, see ["DFSMSDss stand-alone services" on page 515](#).

Using DSS cloud solutions

DFSMSDss supports storing backups in an object storage cloud using one of the two cloud storage solutions:

- Transparent Cloud Tiering (TCT)
- Direct to Cloud

These cloud storage solutions support full volume and logical data set DUMP and RESTORE operations, as well as CLOUDUTILS LIST and DELETE operations to manage data in the cloud. Neither cloud storage solution supports physical data set restore from a full volume backup.

Backups to cloud must be restored using the same system they were created with – a backup created with TCT cannot be restored using Direct to Cloud, and vice versa.

Transparent Cloud Tiering (TCT)

Transparent Cloud Tiering (TCT) is an IBM DS8000 disk storage solution. For TCT, the z/OS host system tells the DS8000 what data to move, and the DS8000 then sends that data directly to the cloud without

ever reading the data into the host system. This process allows CPU MIPS to be offloaded onto the DS8000 disk storage system.

Note: All data flows through the DS8000 endpoint.

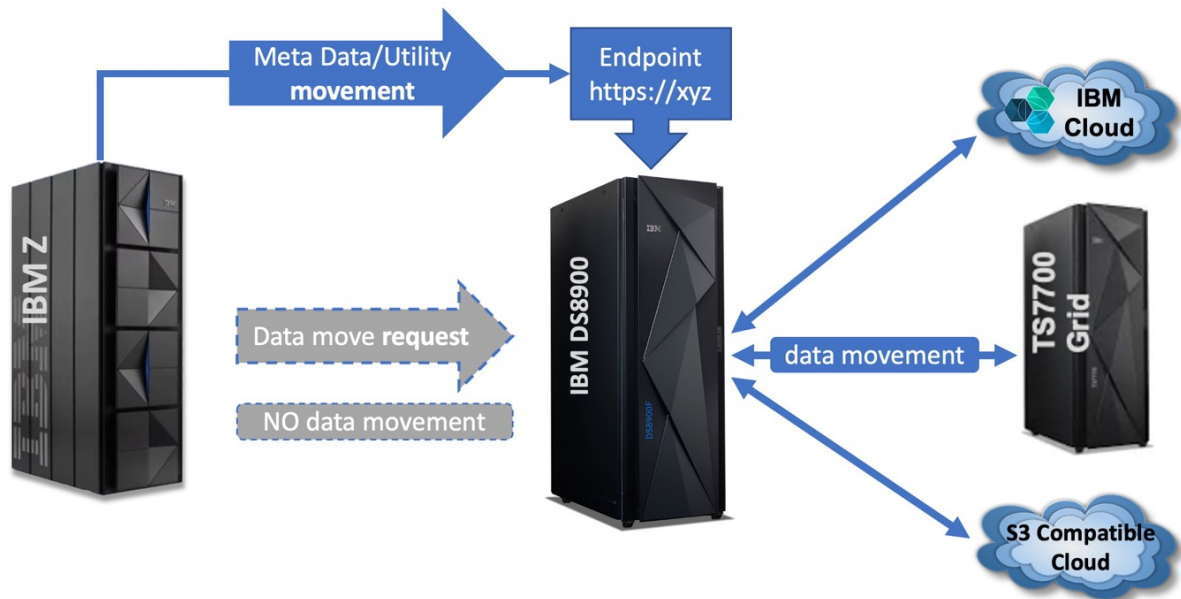


Figure 1. TCT data movement

Note: This is the only cloud solution for a user who wishes to communicate with an IBM TS7700 Advanced Cloud Object Storage system.

Transparent Cloud Tiering (TCT) requirements

DFSMSdss can write to cloud object storage using Transparent Cloud Tiering (TCT) when the following requirements exist:

- TCT is an IBM disk storage solution.
- The source device is in a storage controller that supports transparent cloud tiering.
- The source device is in a storage controller that is connected to an object storage cloud.
- The storage controller and z/OS host are both connected to the same object storage cloud.

Note: In order to use TCT, a cloud must be defined either by an SMS network connection or a CDA Provider File. For more information, see [“Defining an object storage cloud” on page 12](#).

Transparent Cloud Tiering (TCT) limitations

DFSMSdss can create logical data set backups to an object storage cloud by using an IBM DS8000 Storage solution called transparent cloud tiering (TCT). Using this functionality results in server-less movement between the z/OS Host and the DS8000. TCT translates into significant savings in CPU utilization within z/OS. Since this process eliminates data movement from the host, DFSMSdss is unable to perform any data 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. The REBLOCK keyword and the indicator in the VTOC is ignored when a backup is being restored from an object storage cloud.

PDS compression

By default, DFSMSDss compresses a PDS data set during backup processing. You can specify the NOPACKING keyword to prevent DFSMSDss from compressing the PDS. There is no special action that is required in order to back up a PDS to an object storage cloud. When creating a backup to an object storage cloud, DFSMSDss operates as if NOPACKING(**) is specified.

Changing stripe counts

The source stripe count must be the same as the target stripe count for a striped extended format data set. When restoring a backup from an object storage cloud, DFSMSDss ensures that the stripe count does not change. If a data set cannot be created with the same number of stripes the allocation will fail, and the data set is not restored.

An individual stripe extending to more than one volume

DFSMSDss cannot backup striped VSAM data sets when one or more of the stripes spans volumes. DFSMSDss also cannot backup a single striped version 1 extended format sequential data sets that are multivolume.

Block-by-block processing of direct-access data sets

Block-by-block processing occurs when you specify the RELBLOCKADDRESS OR the AUTORELOCKADDRESS keyword. When restoring a backup from an object storage cloud, DFSMSDss operates as if RELBLOCKADDRESS and AUTORELOCKADDRESS are not specified.

Compression and Encryption

DFSMSDss is unable to perform compression or encryption of its buffers since the data movement is being offloaded to the DS8000 storage.

Using a CDA Provider File with Transparent Cloud Tiering (TCT)

DFSMSdfp Cloud Data Access (CDA) should be leveraged to tailor and assist your TCT environment. To this end, a CDA provider file can be used to define a cloud for backups, instead of using an SMS Cloud Network Connection. A provider file can be used to define both a cloud and a TS7700 endpoint being used as a cloud.

Using a CDA provider file is the recommended option for defining a cloud to use for TCT, due to having several key advantages over an SMS Cloud Network Connection.

- A provider file allows more configuration of the properties of the cloud.
- A provider file also allows for multiple endpoints to be specified for a TS7700 configuration, eliminating the single point of failure issue.

For more information on how to set up a provider file, see [Provider file](#) in *z/OS MVS Programming: Callable Services for High-Level Languages*.

In order to set up a provider file to use TCT, the following is required:

1. The provider file must contain the 'tctType' key. If the cloud is a TS7700 endpoint, the associated value should be 'TAPE-OBJECT', otherwise, the value must be 'TCT'.
2. The CDAPROVIDERFILE keyword must be used.

If a provider file is used on a non-TS7700 cloud, a slightly different data movement path is used compared to other TCT operations. DFSMSdfp Cloud Data Access (CDA) is used to move metadata and utility function commands directly to the object storage cloud, while the rest of the data is moved through TCT. This has the benefit of reducing the load on the DS8000. This data movement path is shown in figure below.

Note: The metadata / utility operations go directly to the cloud.

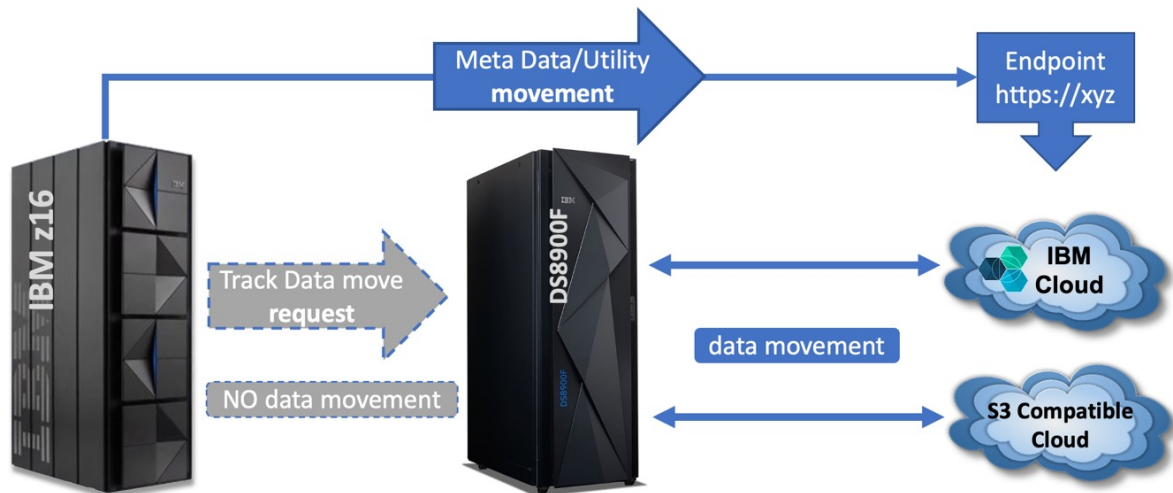


Figure 2. TCT data movement using a provider file to a non-TS7700 cloud

For TS7700 endpoints, data movement when using a provider file is identical to that when using an SMS Cloud Network Connection. In addition, a cloud credential JCL keyword (such as **CDACREDSTORE** or **CLOUDCREDENTIALS**) is required.

Direct to cloud

Direct to Cloud storage solution is disk storage vendor agnostic. This is in contrast to TCT, which requires an IBM DS8000 storage subsystem. This solution does not benefit from the TCT CPU-savings, as it follows a more traditional data movement as our TAPE/DASD backup solution.

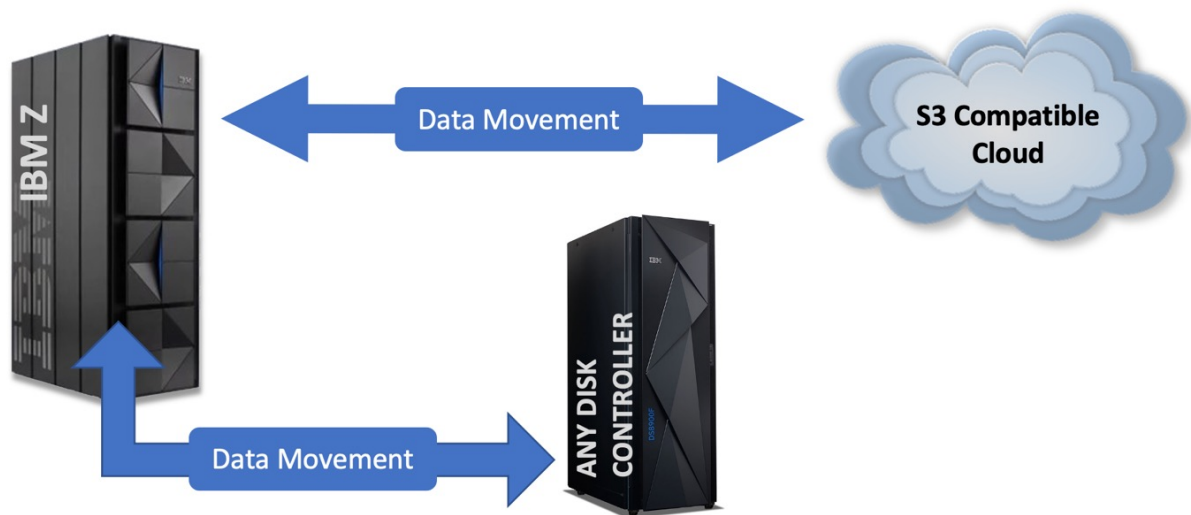


Figure 3. Data movement during Direct to Cloud solution

To invoke this support, the **CDAPROVIDERFILE** keyword must be specified on the dump or restore request, and a valid provider file with all the necessary key-value pairs must be found on the system. For more information on the provider file. For more information, see [“Provider file” on page 12](#)

Direct to cloud limitations

The following limitations exist for Direct to Cloud during logical data set DUMP and RESTORE:

Changing stripe counts

The source stripe count must be the same as the target stripe count for a striped extended format data set. When restoring a backup from an object storage cloud, DFSMSDss ensures that the stripe count does not change. If a target data set cannot be created with the same number of stripes, then the allocation will fail and the data set will not be restored.

An individual stripe extending to more than one volume

DFSMSDss cannot backup striped VSAM data sets when one or more of the stripes spans volumes. DFSMSDss also cannot backup a single striped version 1 extended format sequential data sets that are multivolume.

Reblocking

Reblocking occurs when you specify the REBLOCK keyword or when the VTOC indicates that the data set can be reblocked. The REBLOCK keyword and the indicator in the VTOC is ignored when a backup is being restored from an object storage cloud.

Defining an object storage cloud

There are two ways to define an object storage cloud for DFSMSDss to use, by:

- Using a CDA provider file
- Using an SMS Cloud Network Connection

Either method can be used with the TCT solution, but a provider file is needed to use Direct to Cloud.

For TCT, the CDA provider file is recommended. For more information, see [“Using a CDA Provider File with Transparent Cloud Tiering \(TCT\)”](#) on page 10.

Provider file

The provider file is a .json file used by Cloud Data Access (CDA) to define a cloud connection. DFSMSDss obtains the cloud definition from a provider file when the CDAPROVIDERFILE keyword is specified.

The following section provides information on setting up a provider file that DFSMSDss can use. For more information about the setup and configuration of z/OS Cloud Data Access (CDA) to include provider file, key file updates, and all CDA topics, see [Cloud Data Access \(CDA\) Services in z/OS MVS Programming: Callable Services for High-Level Languages](#)

Sample provider files

Two sample CDA provider files with the required DFSMSDss JSON key-value pairs can be found in `/usr/lpp/dfsms/dss/samples`:

- S3CLOUD.json
- TAPECLWD.json

The sample files provided in `/usr/lpp/dfsms/cda/` should be used as prevailing examples for future enhancements.

• S3CLOUD.json

This provider file assumes that the DS8K has a cloud named S3CLOUD. For any cloud type that CDA supports, use this example. This provider file is set up for a TCT connection to a non-TS7700 cloud. To use this file for Direct to Cloud, remove keys “tctType” and “tctIdentity”.

• TAPECLWD.json

This provider file assumes that the DS8K has a TAPE-OBJECT cloud named TAPECLWD. Use this provider file example for HMCs available for serving as a proxy to communicate with the TS7700.

Keys

The provider file is a UNIX file in .JSON format. The characteristics of the cloud connection are defined in the provider file using a series of JSON key-value pairs.

Required Keys

- enableDFSMSdss
 - This key identifies whether or not DFSMSdss can utilize this provider file.
 - A value other than YES or the absence of this key will result in the provider file being ignored.
- supportedOperations
 - This key must be present as an empty list (i.e. []) or a list of operations.
 - When the tctType is not TAPE-OBJECT, the following operations must be present:
 - GETOBJECT
 - GETLARGEOBJECT
 - WRITEOBJECT
 - WRITELARGEOBJECT
 - LISTOBJECT
 - DELETEOBJECT
 - CREATEBUCKET
 - DELETEBUCKET
 - LISTBUCKETS
 - The full required text for this list should be copied from the sample provider file S3CLOUD.json located at `/usr/lpp/dfsms/dss/samples`.

TCT specific keys

- tctType
 - When the value is set to TAPE-OBJECT the cloud described is a TS7700 Advanced Object Store. All data and utility operations flow through the DS8000 storage cloud proxy using either FICON or Ethernet connections.
 - When the value is set to TCT, all user-data flows through the DS8000 storage subsystem cloud proxy using FICON. All metadata and utility operations flow directly with the storage cloud via Ethernet instead of using the DS8000 proxy.
- tctIdentity
 - This key must have the value of the DS8000 HMC service account ID. The same value that would be specified in the Identity field of the SMS network connection.
- cloudName
 - If cloudName is not specified, the cloud name is assumed to be the name of the provider file. If you prefer to use a provider file name that is different from your cloud name you must specify the key cloudName with the value of the cloud name known to the DS8000.
 - If the key 'tctType' is not specified, this keyword will be ignored.

Cloud definition keys

The following keys are used to define the characteristics of the cloud. For more information about specifying these keys, see [Provider file](#) in *z/OS MVS Programming: Callable Services for High-Level Languages*.

- host

- port
- sslVersion
- sslKey

SMS cloud network connections

For information on how to define a cloud using SMS, see [Defining clouds](#) in *z/OS DFSMSdss Storage Administration*.

Using an object storage cloud

Data in the cloud is structured in the following way:

Container

A container (also known as a bucket) is similar to a folder in which objects are stored.

DSS created backup

A DSS created backup is comprised of multiple objects, each sharing an object prefix unique to the container. For more information on the components of a DSS created backup, see [“Data set and volume backup format”](#) on page 179.

Keywords

There are three required keywords when performing a cloud DUMP or RESTORE operation:

CLOUD(*cloud_name*)

- This keyword tells DFSMSdss that the backup is to be directed to an object storage cloud and identifies the cloud definition to use. An object storage cloud can be defined either with an SMS construct or with a Cloud Data Access (CDA) provider file.
- When the **CDAPROVIDERFILE** keyword is specified, DFSMSdss first checks for a CDA provider file called *cloud_name.json*. If one does not exist or the CDA provider file is empty, then the information in the SMS Network connection is used to communicate with the object storage cloud.
- If **CDAPROVIDERFILE** is not specified, only the SMS network connection will be checked.

CONTAINER(*container_name*)

- Specifies the container in which the objects are to be referenced. During a DUMP operation, the container specified as input does not have to exist at the time of the backup. If necessary, DFSMSdss creates it for you.
- The container name that is specified will include a DFSMSdss specific prefix when invoked using batch JCL. This is to distinguish DFSMSdss backups.
 - When using the TCT cloud storage solution and no prefix is specified, the prefix prepended will be 'SYSZADR.'
 - When using the Direct to Cloud storage solution, the prefix prepended will be 'SYSZADR-'
 - If either 'SYSZADR.' or 'SYSZADR-' is specified, no prefix will be prepended regardless of the cloud storage solution used.

OBJECTPREFIX(*object_prefix*)

Specifies a unique prefix that DFSMSdss is to use for all the objects that make up a particular backup. The prefix provides uniqueness among multiple backups in the same container. One way to ensure uniqueness among multiple backups of the same data is to use timestamps as part of the object prefix. At the beginning of the backup process, DFSMSdss determines whether a backup that uses the same *object_prefix* exists in the specified *cloud_name* and *container_name*. If a backup exists with the same *object_prefix*, then DFSMSdss fails the backup. To overwrite a backup that uses the same *object_prefix*, you can set a patch at offset X'5D'. For more information on DFSMSdss patch bytes, see Chapter 14, [“DFSMSdss patch area,”](#) on page 213.

Additionally, when using SMS network connection information (or for a provider file with tctType of TAPE-OBJECT) one of the following keywords is required:

CLOUDCREDENTIAL(*cloud_credentials*)

Specifies the password for the account that is defined within the cloud construct. This password is case-sensitive and is suppressed from SYSPRINT and SVCDUMPS. The cloud credentials must be kept secure. If there are batch JCL jobs that specify this keyword, then those jobs should be in a data set or library that has limited access and controlled by a security product. If the credentials cannot be kept secure, then do not use this keyword. It is recommended to use the CDACREDSTORE keyword instead of the CLOUDCREDENTIAL keyword.

CDACREDSTORE

Specifies that the z/OS Cloud Data Access Authorization Utility is encrypted and stored credentials for the specified cloud_name.

Managing backups with CLOUDUTILS

DFSMSDss provides a CLOUDUTILS command to help manage your backups created in an object store. The CLOUDUTILS command can list and delete the DFSMSDss containers or backups in a DFSMSDss container. The DFSMSDss container is one that is either the prefixed "SYSZADR." or "SYSZADR-".

When developing the backup strategy, at least two of these factors must be considered:

1. The frequency of your backups.
 - If you create your backups daily or weekly, then you may find adding a date to your object prefix enough to manage multiple copies of the same data.
 - If you create your backups more frequently, such as more than once per day, then consider adding a timestamp to the object prefix.
2. How many copies to keep for recovery?

Using a combination of date and time in the object prefix allows for a CLOUDUTILS delete to discard a backup copy that is to be expired upon successful backup. You can use REXX to generate the JCL and complete the date portion of the object prefix for the memory dumps that are to be deleted. Below is an example of a skeleton JCL.

```
//STEP0001 EXEC PGM=ADRDSSU
//SYSPRINT DD DSP=(NEW,CATLG),DSN=MY.&JOBNAME.INVENTORY (+1)...
//SYSIN DD *
PARALLEL
DUMP FULL INDYNAM(vol001) CDACREDS CLOUD(cloudname) CONTAINER(DSSDUMPS)
OBJECTPREFIX('&JOBNAME./vol1001/D&YR4.&JDAY.')
DUMP FULL INDYNAM(vol002) CDACREDS CLOUD(cloudname) CONTAINER(DSSDUMPS)
OBJECTPREFIX('&JOBNAME./vol1002/D&YR4.&JDAY.')
SERIAL
IF MAXCC=0 THEN
DO
PARALLEL
CLOUDUTILS DELETE CDACREDS CLOUD(cloudname) CONTAINER(DSSDUMPS)
OBJECTPREFIX('&JOBNAME./vol1001/D&EXPIRE.')
CLOUDUTILS DELETE CDACREDS CLOUD(cloudname) CONTAINER(DSSDUMPS)
OBJECTPREFIX('&JOBNAME./vol1002/D&EXPIRE.')
SERIAL
END
```

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.

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 might 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.

Related reading: For more information about SMS, see [z/OS DFSMSdss Storage Administration](#).

Moving data with concurrent copy

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

Related reading: For more information about concurrent copy and how to use it, see [“CONCURRENT” on page 320](#).

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 more information about moving data using FlashCopy, see [Chapter 7, “Managing data movement with DFSMSdss,” on page 97](#).

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 more information about moving data using SnapShot, see [Chapter 7, “Managing data movement with DFSMSdss,” on page 97](#).

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 more information about converting data sets, see [Chapter 8, “Converting data to and from SMS management,”](#) on page 135.

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 more information about converting volumes, see [Chapter 8, “Converting data to and from SMS management,”](#) on page 135.

Managing space with DFSMSdss

DFSMSdss provides the following functions to help you manage 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.

CONSOLIDATE

Performs data set extent consolidation or reduction on a volume.

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. (You can also use the COPY command to combine data set extents.)

RELEASE

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

SPACEREL

Releases physical space associated with free space extents on an extent space efficient (ESE) volume back to the extent pool. The command can be issued against one or more volumes or storage groups.

Related reading: For more information about managing space, see [Chapter 9, “Managing space with DFSMSdss,”](#) on page 143.

JCL examples

The JCL examples in this document might use REGION=4096K, however, you might have to tailor this to fit your environment.

Chapter 2. Requirements for running DFSMSdss

This topic describes the requirements for running DFSMSdss.

Understanding the operating environment

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

You can run the DFSMSdss stand-alone restore program on an IBM Z server in S/390® mode, or on an IBM System 390 server in S/390 mode or S/370 mode. The available modes are dependent on your CPU type and model.

Additionally, you can run the DFSMSdss stand-alone restore program on a z/VM® virtual machine in XA mode.

Storage requirements

Usually, you can let DFSMSdss determine the amount of storage it uses for an operation. However, sometimes you might want closer control of the amount of storage DFSMSdss uses. Use the storage estimates that follow as a starting point for determining minimum region sizes in which DFSMSdss can run. [Table 1 on page 19](#) and [Table 2 on page 20](#) show the minimum storage requirements, in bytes, to run each DFSMSdss operation. [Table 3 on page 20](#) shows the minimum storage requirements, in bytes, to restore partitioned and VSAM data sets to unlike devices. The legend for [Table 1 on page 19](#), [Table 2 on page 20](#), and [Table 3 on page 20](#) is displayed beneath [Table 3 on page 20](#). The values include the storage that is 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.

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

DFSMSdss dynamically allocates data sets by using below the line DSABs. Consider this when determining the REGION size of your job that uses DFSMSdss.

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 that is required is the sum of the storage that is 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.

When performing a DUMP FULL to cloud by using the Cloud Data Access (CDA) support, an additional 8 megabytes of overhead is required due to CDA.

Table 1. Minimum storage requirements for DFSMSdss operations with I/O Buffers below 16 megabyte virtual	
DFSMSdss Command	Storage Requirements
COMPRESS	$\text{dsssize} + (2 * \text{trksize of largest device}) + (\text{number of additional volumes, up to five} * \text{copsizsize})$
CONVERTV	$\text{dsssize} + (\text{buffersize} * 5)$
COPY (FULL)	$\text{dsssize} + (\text{trksize} * 5)$
COPYDUMP	$\text{dsssize} + (\text{buffersize} * 5)$

Table 1. Minimum storage requirements for DFSMSdss operations with I/O Buffers below 16 megabyte virtual (continued)

DFSMSdss Command	Storage Requirements
DEFRAG	dsssize + (trksize * 5) + (16 KB if VSAM present) + (1 KB * 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 16 MB Virtual	Storage Above 16 MB 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 + (16 KB if VSAM present) + (1 KB * 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) + 8 KB)
VSAM Data Sets	dsssize + (((trksize + 64) * 5) + (2 * maximum record size) + (3 * buffspace))

Legend: This legend applies to [Table 1 on page 19](#), [Table 2 on page 20](#), and [Table 3 on page 20](#).

KB

1024 bytes.

dsssize

DFSMSdss load module size, 2550 KB.

copysize

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

buffersize

256 KB is the default for I/O to tape unless DFSMSdss is forced to use a smaller size. Otherwise, it is the maximum of the largest trksize that is used or 32 KB.

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. You can also use DFSMSdss with object storage clouds.

Note:

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.
4. DFSMSdss supports public and private clouds that use an Openstack Swift based API to put and get objects. The authentication methods supported are either version 1 or version 2 of the Openstack Identity API.

Volume formats

You can use DFSMSdss with the following DASD volume formats:

- Volumes with indexed VTOCs, see [“Indexed VTOC” on page 22](#) for details.
- 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.

For an SMS-managed volume to be dynamically allocated it must be in a status other than DISALL or NOTCON.

Usage of Read-Only DASD volumes as the source or target of DFSMSdss commands is not supported.

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.

Indexed VTOC

DFSMSDss recognizes three different formats for a VTOC index:

- SYS1.VTOCIX.Vvolser
- SYS1.VTOCIX.volser
- SYS1.VTOCIX.Volser.

If the first character of a VOLSER is numeric, DFSMSDss renames it (or prefixes it) so that the VOLSER begins with the letter "V." For example:

- If a VOLSER is 12345, DFSMSDss renames it to V12345
- If a VOLSER is 123456, DFSMSDss renames it to V23456.

Data set organizations

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

- DB2®
- Direct access
- EXCP (execute channel program)
- Partitioned, including:
 - PDS (partitioned data set)
 - PDSE (partitioned data set extended)
- 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 (z/OS File System) data set
- Unmovable data set types (PSU, POU, DAU, ABSTR, ISU, and direct with OPTCD=A).

Note:

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 or CONSOLIDATE data set—

Allocated by DFSMSDss to contain information about relocated DASD extents. The data set name is in the following format:

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

where *.....* 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 or CONSOLIDATE operation ends successfully.

If the operation is interrupted (for example, if DFSMSDss is canceled), the data set is left on the volume. To delete the data set, repeat the DEFRAG or CONSOLIDATE operation. Before doing so, however, observe the following considerations:

- Determine whether you need to convert the index VTOC (IXFORMAT) volume to a non-indexed VTOC (OSFORMAT) before rerunning the DEFRAG or CONSOLIDATE operation. Otherwise, the volume free space values might be incorrect.
- 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 (<i>n</i>)	Temporary name
1	<i>dsnhlq.Thhmsst.chmiju00</i>
2	First 2 qualifiers. <i>Thhmsst.chmiju00</i>
>2	First 3 qualifiers. <i>Thhmsst.chmiju00</i>

In the next-to-last qualifier, *Thhmsst*, *hhmsst* is the first part of the time stamp information: hours (*hh*), minutes (*mm*), seconds (*ss*), and tenths of a second (*t*).

In the last qualifier, *chmiju00*, *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 *hmiju* is rest of the time stamp information: hundredths of a second (*h*), milliseconds (*m*), ten-thousandths of seconds (*i*), hundred-thousandths of seconds (*j*), and microseconds (*u*).).

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.Thmmsstt
```

- 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.Thmmsstt
```

- 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. This topic describes these two aspects of DFSMSdss.

UNIX file filtering

The following section describes how to use UNIX File filtering to select files for processing.

Defining logical and physical processing

DFSMSdss can perform two kinds of processing when executing COPY, DUMP, and RELEASE, 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 rename data sets through the application program interface (API) during logical dump processing.
- 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.

Note:

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 keyword is only supported for non-VSAM data sets during physical data set restore. VSAM Alternate Indexes (AIX) cannot be renamed during physical data set copy or 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 COPY or DUMP command and input volumes with the PHYSINDDDNAME or PHYSINDYNAM keyword. This produces a physical data set copy or 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 the DFSMSdss stand-alone restore program (for physical dump tapes only).
- 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 z/OS 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 might not properly select broken data sets for processing because it 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:

- COMPRESS
- CONSOLIDATE

- Data set copy
- Logical data set copy
- Logical data set dump
- Logical data set restore
- Physical data set copy
- Physical data set dump
- Physical data set restore
- 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)

CREDIT

Creation date (absolute or relative)

DATACLAS

Data class for SMS

DSCHA

Whether the data-set-changed indicator is on

DSORG

Data set organization (SAM, PAM, PDS, PDSE, BDAM, EXCP, 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 indicator 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).

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.

Related reading: For more information, see [*z/OS DFSMS Using the Interactive Storage Management Facility*](#).

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.

Related reading: For JCL information and examples, see the topic on specifying DFSMSdss commands in Chapter 15, “Specifying DFSMSdss commands,” on page 245.

Invoking DFSMSdss with the application interface

This topic documents General-Use Programming Interface and Associated Guidance Information.

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 non-supervisor state.

Related reading: For more information about the application interface, see [Chapter 23, “Application programming interface,”](#) on page 577.

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.
- Specify some information on how a new target data is allocated.

Related reading: For more information about UIM exit functions, see [Chapter 24, “Examples of the application program with the user interaction module \(UIM\),”](#) on page 627.

Chapter 5. Protecting DFSMSdss functions

You can protect DFSMSdss/ISMF functions and some DFSMSdss keywords. This topic discusses the functions of DFSMSdss for which you can control access through the RACF element of z/OS Security Server.

Protecting DFSMSdss and ISMF functions with RACF

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, DGTL LIB. The load module names are listed in Table 4 on page 36 and Table 5 on page 36. The module names are found in the DGTSMMD1 member of the panel library.

Table 4 on page 36 lists the names for the corresponding line operators. The module names for line operators are found in the DGTTL PD3 member of the load library. Table 5 on page 36 lists the names for commands. These names are in the DGTTC TD2 member of the load library.

<i>Table 4. Module Names for DFSMSdss/ISMF Line Operators</i>		
Line Operator	Data Set Application Module Name	Volume Application Module Name
CGCREATE	—	DGTFCG01
COMPRESS	DGTFCM01	DGTFC S01
CONSOLID	—	DGTFCI01
CONVERTV	—	DGTFCN01
COPY	DGTFCY01	DGTFCV01
DEFRAG	—	DGTDFD01
DUMP	DGT FDP01	DGT FDM01
RELEASE	DGT FRL01	DGT FRV01
RESTORE	DGT FRT01	DGT FRO01

<i>Table 5. Module Names for DFSMSdss/ISMF Data Set Application Commands</i>	
Command	Module Name
COMPRESS	DGTFCP01
COPY	DGTFCO01
DUMP	DGT FDU01
RELEASE	DGT FRE01
RESTORE	DGT FRR00

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.DGTL LIB. 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 DGTL LIB.

End 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.

For more information about how to perform these steps and the options you have using program control, refer to [z/OS Security Server RACF Security Administrator's Guide](#).

Protecting DFSMSdss functions with RACF FACILITY class profiles

Besides protecting DFSMSdss/ISMF functions, you can also protect certain DFSMSdss keywords and functions. You do so by defining RACF FACILITY class profiles and restricting access to those profiles. [Table 6 on page 37](#) lists these keywords and functions, and their associated RACF FACILITY 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 profile has been defined.

When the RACF FACILITY class is active and one of the profiles listed in [Table 6 on page 37](#) is defined, you must have READ access authority to use the indicated command or keyword. Otherwise, anyone can use the indicated command or keyword. If RACF FACILITY class checking is not set up for these keywords, any DFSMSdss user can use them.

<i>Table 6. RACF FACILITY Class Profile Names for DFSMSdss Keywords</i>	
Keyword or Function	Profile Name
BYPASSACS with COPY	STGADMIN.ADR.COPY.BYPASSACS
BYPASSACS with RESTORE	STGADMIN.ADR.RESTORE.BYPASSACS
CGCREATED	STGADMIN.ADR.CGCREATE
CLOUD with DUMP	STGADMIN.ADR.DUMP.CLOUD
CLOUD with RESTORE	STGADMIN.ADR.RESTORE.CLOUD
CLOUDUTILS	STGADMIN.ADR.CLOUDUTILS
CONCURRENT with COPY	STGADMIN.ADR.COPY.CNCURRNT
CONCURRENT with DUMP	STGADMIN.ADR.DUMP.CNCURRNT
CONSOLIDATE	STGADMIN.ADR.CONOLID
CONVERTV	STGADMIN.ADR.CONVERTV
DEFRAG	STGADMIN.ADR.DEFRAG
DETECATALOGENTRY with RESTORE	STGADMIN.ADR.RESTORE.DELCATE
DELETE with CLOUDUTILS	STGADMIN.ADR.CLOUDUTILS.DELETE
FCCGFREEZE with COPY	STGADMIN.ADR.COPY.FCFREEZE
FCFASTREVERSERESTORE with COPY	STGADMIN.ADR.COPY.FCFRR
FCSETGTOK with COPY	STGADMIN.ADR.COPY.FCSETGT
FCTOPPRCPPRIMARY with COPY	STGADMIN.ADR.COPY.FCTOPPRCP
FCTOPPRCPPRIMARY with DEFRAG	STGADMIN.ADR.DEFRAG.FCTOPPRCP

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

Keyword or Function	Profile Name
FCTOXRCPPRIMARY with COPY	STGADMIN.ADR.COPY.FCTOXRCP
FlashCopy with CONSOLIDATE	STGADMIN.ADR.CONSolid.FLASHCPY
FlashCopy with COPY	STGADMIN.ADR.COPY.FLASHCPY
FlashCopy with DEFRAG	STGADMIN.ADR.DEFRAG.FLASHCPY
FORCE with CLOUTUTILS	STGADMIN.ADR.CLOUD.FORCE
IMPORT with RESTORE	STGADMIN.ADR.RESTORE.IMPORT
INCAT(<i>catname</i>) with COPY	STGADMIN.ADR.COPY.INCAT
INCAT(<i>catname</i>) with DUMP	STGADMIN.ADR.DUMP.INCAT
INCAT(<i>catname</i>) with RELEASE	STGADMIN.ADR.RELEASE.INCAT
PRINT	STGADMIN.ADR.PRINT
PRINT with TRACKS	STGADMIN.ADR.PRINT.TRACKS
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
RESET with DUMP	STGADMIN.ADR.DUMP.RESET
RESET(YES) with RESTORE	STGADMIN.ADR.RESTORE.RESET.YES
SPACEREL	STGADMIN.ADR.SPACEREL
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
TOLERATE(WRITERS) with DUMP	STGADMIN.ADR.DUMP.TOLERATE.WRITERS
ZCOMPRESS with DUMP	STGADMIN.ADR.DUMP.ZCOMPRESS

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

For more information about the installation options exit routine, refer to [z/OS DFSMS Installation Exits](#).

For more information about RACF class profiles, refer to [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.

For more information about how to protect a library, refer to [“Protecting DFSMSdss/ISMF modules” on page 36](#).

Chapter 6. Managing availability with DFSMSDss

One of the major functions of DFSMSDss is the backing up and recovery of data. Using the DUMP and RESTORE commands, you can backup and recover data sets, z/OS UNIX files, 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 data sets, and z/OS UNIX files—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, or an object storage cloud, 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, with the exception of logical backups to an object storage cloud, lets you restore data sets to unlike devices.

General backups of z/OS UNIX files should also be performed to protect against accidental deletion or changes to user's files.

Data set and UNIX file backup

For data set and UNIX file backups, 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.

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. To perform it, however, 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 DFSMSHsm is not 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 indicator, 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.

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 example, DFSMSHsm or DFSMSdss incremental backups) for disaster recovery. Disaster recovery backups require 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 or stored in an object storage cloud that both the primary and remote disaster recovery site can be connected to.

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
```

Note: Logical data set backups directed to an object storage cloud cannot be restored to unlike devices.

The following is an example of creating a disaster recovery backup to an object storage cloud:

```
DUMP -  
  DATASET (INCLUDE (CRIT*.**) -  
            BY (MGMTCLAS, EQ, MCNAME)) -  
  CLOUD (cloud_name) -  
  CONTAINER (dss_dr_2016) -  
  OBJECTPREFIX (16100_dr_) -  
  CLOUDCREDENTIALS (cloud_password)
```

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 might need to use the DELETECATALOGENTRY or IMPORT keywords or both. For information about using these keywords, refer to [“Logical restore of data sets with phantom catalog entries”](#) on page 89.

For more information about ABARS, refer to *z/OS DFSMSHsm Storage Administration*.

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.

Note: Vital record backups directed to an object storage cloud cannot be restored to unlike devices.

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
```

The following is an example of creating a vital record backup to an object storage cloud:

```
DUMP -
  DATASET(INCLUDE(VR*.**) -
    BY(MGMTCLAS,EQ,MCNAME)) -
  CLOUD(cloud_name) -
  CONTAINER(dss_vr_2016) -
  OBJECTPREFIX(16100_vr_) -
  CLOUDCREDENTIALS(cloud_password)
```

Archiving data sets

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.

Note: Archives directed to an object storage cloud cannot 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
```

In another 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, 2016 to a cloud named *cloud_name* in a container named *dss_archives_2016*, with each object having a prefix of *100_* :

```
DUMP -
  DATASET(BY((REFDT LT 16100) (MGMTCLAS EQ MCNAME1))) -
  CLOUD(cloud_name) -
  CONTAINER(dss_archives_2016) -
  OBJECTPREFIX(100_) -
  CLOUDCREDENTIALS(cloud_password) -
  DELETE -
  PURGE
```

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), or as objects in an object storage cloud. The storage medium for the

sequential data set can be tape or DASD. When the output resides on DASD, it may be a basic, large, or extended format data set. When the output resides in cloud storage, data is storage as a set of objects.

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 [Chapter 16, “DFSMSDss filtering—choosing the data sets you want processed,”](#) on page 255.

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 indicator (DSCHA) after successfully dumping the data set. For more information about the RESET keyword, refer to [“Backup with concurrent copy”](#) on page 45.

Note:

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.

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.

For more information about the data-set-changed indicator and REFDT, refer to [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.

Renaming data sets during dump processing

You can specify new names for dumped data sets through the NEWNAMEUNCONDITIONAL keyword on the DUMP command. With this keyword you assign new names to data sets during dump processing, rather than renaming them later during restore processing. You might find NEWNAMEUNCONDITIONAL to be useful if your installation keeps its dumped data sets cataloged to avoid name contention between the data sets that are backed up (dumped) and the production data sets.

The NEWNAMEUNCONDITIONAL keyword can only be used when invoking DFSMSdss through the Application Programming Interface (API).

To assign a new name for a data set, you can specify a source data set with a corresponding new name. For VSAM data sets, DFSMSdss derives VSAM component names based on whether or not the name assigned for the VSAM cluster is currently cataloged in the standard order of search. If the data set is not cataloged, or the installation requests DFSMSdss to bypass checking to determine if the new cluster name is cataloged in the standard order of search by the way of the ADRUFO field, then DFSMSdss uses the same criteria that it uses for the RENAME or RENAMEUNCONDITIONAL keywords to determine the new names for the VSAM components (refer to “Renaming data sets” on page 101). For example, DFSMSdss might append .DATA (for a data component) and .INDEX (for an index component) to the new cluster name when deriving the new VSAM component names. If the new named data set is cataloged in the standard order of search, DFSMSdss associates the component names and the catalog name in the dump records for the new named data set.

Data sets that do not meet the NEWNAMEUNCONDITIONAL criteria retain their original names.

For the other dump records in the ADRTAPB Data Area, DFSMSdss associates the source SMS constructs and volume serial numbers from the source data set.

To use the NEWNAMEUNCONDITIONAL keyword, you do not require access authority to the source (READ) or the target (ALTER) data sets and their catalogs. Instead, renaming data sets is authorized through the existing ADMINISTRATOR keyword and is protected by the RACF FACILITY class, STGADMIN.ADR.STGADMIN.DUMP.NEWNAME. This profile does not permit you to delete a data set. Also, DFSMSdss propagates any passwords retrieved from the source data set to the new named data set.

Note:

1. If you specify the NEWNAMEUNCONDITIONAL keyword with the SPHERE keyword to process a base VSAM cluster, DFSMSdss processes the entire VSAM sphere and assigns a new name to associated alternate indexes and paths. If you do not specify rename filter criteria for all of the data sets in the sphere, DFSMSdss issues an error message and does not process the source data set.
2. If the new name filter has errors, DFSMSdss does not process the data set. The new name is truncated to fit 44 characters. If it ends with a period, that period is also truncated.
3. You cannot change the number of qualifiers unless you use fully-qualified names, for example, NEWNAMEUNCONDITIONAL((A.B.C,A.B.C.D)). If the new name is not fully qualified, it must contain the same number of qualifiers as the old name. For example, given the old name filter DATE.**, the new name filter DATE.**.LIST, DATE.MARCH.TODAY.OLDLIST is renamed, but DATE.MARCH.OLDLIST is not.
4. GDG relative generation filtering cannot be used for old or new names.

For more information about dump records, refer to “ADRTAPB data area” on page 181.

For more information about ADRUFO installation options exit routine, refer to [z/OS DFSMS Installation Exits](#).

For more information about specifying the NEWNAMEUNCONDITIONAL keyword, refer to [“NEWNAMEUNCONDITIONAL” on page 414](#).

Backup with concurrent copy

Programming Interface Information

DFSMSdss provides the concurrent copy function to allow you to backup data while minimizing the amount of time in which the data is unavailable. The database 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. You can invoke DFSMSdss directly or through the DFSMSdss application program interface (API) to do a concurrent copy of the entire database. After initialization, DFSMSdss releases any serialization it holds on the data sets and prints a message to SYSPRINT and the console that the concurrent copy operation is logically complete. If you invoke DFSMSdss through the API, DFSMSdss informs the caller through the UIM exit option, Eioption 24, and the application can resume normal operation.

End Programming Interface Information

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

If the source device supports data set FlashCopy or SnapShot, DFSMSDss optionally uses the FlashCopy or SnapShot to provide virtual concurrent copy.

For more information on concurrent copy and virtual concurrent copy, refer to [“Performance considerations”](#) on page 59.

Specifying concurrent copy for DUMP requests

On the DFSMSDss DUMP command, you can specify that DFSMSDss is to use the concurrent copy function to process data. To do so, you specify the CONCURRENT keyword, and, optionally, one of several available sub-keywords to indicate the type of concurrent copy to be used and whether DFSMSDss can use other methods of data movement. If you do not specify the CONCURRENT keyword, your DUMP request does not use concurrent copy.

The CONCURRENT keyword applies to all of the data being dumped. You cannot apply this function to a subset of the data being processed.

If you specify the CONCURRENT keyword, DFSMSDss might use a function equivalent to the cache-based concurrent copy, called *virtual concurrent copy*. During virtual concurrent copy, data is “flushed” or “snapped” from the source location to an intermediate location, and then copied to the target location using standard I/O. The operation is logically complete after the source data is “flushed” or “snapped” to the intermediate location and physically complete after the data is moved to the target media.

If the source volume supports data set FlashCopy, DFSMSDss uses FlashCopy to provide virtual concurrent copy. If the source volume is a RAMAC Virtual Array (RVA), DFSMSDss uses SnapShot. For more information about virtual concurrent copy, refer to [“Using concurrent copy”](#) on page 7.



Attention: Use concurrent copy only during periods of light update activity for the data sets or volumes involved. Performing cache-based concurrent copy operations against many large data sets when there is also heavy update activity (such as reorganizing data sets or initializing the volume the data sets reside on) might result in a shortage of storage, because data is transferred to z/OS data space storage faster than DFSMSDss can process it. When you use multiple simultaneous concurrent copy tasks to process large, heavily updated data sets, you might also experience long run times and contention for SYS.DATA.SPACE.LATCH.SET. You must ensure that during the concurrent copy operation another system does not reserve volumes that are to be processed. You must also ensure that jobs and address spaces that are to use the concurrent copy are assigned a WLM service class with a high execution velocity. Do not assign a discretionary goal to concurrent copy work. You should spread multiple concurrent copy jobs across as many LPARs as possible and avoid the use of PARALLEL mode in DFSMSDss.

Note:

1. To help ensure data integrity, do not update the data during concurrent copy initialization.
2. If a concurrent copy operation fails after signaling that the concurrent copy initialization is complete (and update activity on the data has resumed), you cannot recover the data to the point-in-time at which the concurrent copy operation was started. The data might have been updated while the copy operation was progressing.
3. VM mini-volumes are supported if you are using RVA devices to the extent that they are supported by IBM Extended Facilities Product (IXFP) device reporting.
4. The use of concurrent copy and virtual concurrent copy with the DFSMSDss DUMP command is controlled by the RACF FACILITY class profile, STGADMIN.ADR.DUMP.CNCURNT.
5. When creating a backup to an object storage cloud, the use of concurrent copy and virtual concurrent copy is not supported.

For information about specifying CONCURRENT and other DUMP command keywords, refer to [“DUMP command for DFSMSDss”](#) on page 386.

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.

For more information about the installation options exit, refer to [z/OS DFSMS Installation Exits](#).

For more information about the CONCURRENT keyword, refer to [“CONCURRENT” on page 320](#).

Backing up data sets to an object storage cloud

DFSMSdss can create logical data set backups to an object storage cloud by using an IBM DS8000 Storage solution called transparent cloud tiering (TCT). Using this functionality results in server-less movement between the z/OS Host and the DS8000. TCT translates into significant savings in CPU utilization within z/OS. Since this process eliminates data movement from the host, DFSMSdss is unable to perform any data 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. The REBLOCK keyword and the indicator in the VTOC is ignored when a backup is being restored from an object storage cloud.

PDS compression

DFSMSdss compresses a PDS data set during backup processing, by default. You can specify the NOPACKING keyword to prevent DFSMSdss from compressing the PDS. There is no special action that is needed in order to backup a PDS to an object storage cloud. When creating a backup to an object storage cloud, DFSMSdss operates as if NOPACKING(**) is specified.

Changing stripe counts

The source stripe count must be the same as the target stripe count for a striped extended format data set. When restoring a backup from an object storage cloud, DFSMSdss ensures that the stripe count does not change. If a data set cannot be created with the same number of stripes the allocation fails and the data set is not restored.

An individual stripe extending to more than one volume

DFSMSdss cannot backup striped VSAM data sets when one or more of the stripes spans volumes. DFSMSdss also cannot backup a single striped version 1 extended format sequential data sets that are multivolume.

Block-by-block processing of direct-access data sets

Block-by-block processing occurs when you specify the RELBLOCKADDRESS OR the AUTORELOCKADDRESS keyword. When restoring a backup from an object storage cloud, DFSMSdss operates as if RELBLOCKADDRESS and AUTORELOCKADDRESS are not specified.

Compression and Encryption

DFSMSdss is unable to perform compression or encryption of its buffers since the data movement is being offloaded to the DS8000 storage. This includes the ZCOMPRESS keyword.

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 or an RLS user catalog. 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 provides 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 for a VSAM base cluster. There are advantages to using the COPY command instead of the DUMP command:

- 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 instantaneously to a data set FlashCopy-capable ESS.
- 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.

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.

For more information about using the CICSVRBACKUP keyword on the DUMP command, refer to [“CICSVRBACKUP” on page 399](#).

For more information about using different methods to generate a unique output data set name, refer to *CICS VSAM Recovery Implementation Guide*.

For more information about the CICSVR generated new name and the required RENAMEUNCONDITIONAL specification, refer to *CICS VSAM Recovery Implementation Guide*.

A backup scenario

As discussed in [“Backup and recovery” on page 39](#), 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 indicator. 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 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.

For more information about the serialization of HFS data sets, refer to [Chapter 22, “Data integrity—serialization,”](#) on page 561.

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 copy or dump

Back up mounted zFS data sets with logical data set copy or dump. This will provide the quiesce serialization mechanism (BPX1PCT) to ensure data integrity.

For non-shared file systems, you must run the DFSMSdss copy or dump job on the same system where the data set is mounted for update; the quiesce is only effective on the system it is issued from. For shared file systems, you can run the DFSMSdss job from any system in the sysplex, because the quiesce is handled globally across all systems. TOL(ENQF) is not honored during logical dump of zFS and is not honored for a source zFS data set during logical copy.

Physical dump

Physical dump does not provide the quiesce serialization mechanism, and it is not recommended for backing up mounted zFS data sets. If the data set is mounted before the physical dump job begins, z/OS UNIX will have the zFS data set allocated and opened. Allocation and open processing will have obtained SYSDSN and SYSVSAM enqueues. DFSMSdss will therefore not be able to obtain the exclusive enqueues to perform the physical dump.

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. For more details, see [“zFS data sets”](#) on page 565.

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.

Note: DFSMSDss does not support backing up integrated catalog facility user catalogs to an object storage cloud.

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(INCL(TEST.CAT.PUBSEXMP)) -
            OUTDDNAME (TAPE)
/*
```

Figure 4 on page 51 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
ADR001I (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
TEST.CAT.PUBSEXMP      CLUSTER NAME
SYS1.MVSRES.MASTCAT    CATALOG NAME
TEST.CAT.PUBSEXMP      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 4. Output from a Dump of an Integrated Catalog Facility User Catalog

For more information about the LOCK attribute, refer to [z/OS DFSMS Managing Catalogs](#).

For more information about the installation authorization exit routine, refer to [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 when the data set is encrypted 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.

Restrictions:

- Non-encrypted extended format VSAM data sets cannot be dumped to DASD or tape with the NOVALIDATE keyword.
- DFSMSdss does not support VALIDATE processing when backing up indexed VSAM data sets to an object storage cloud, or when backing up encrypted indexed VSAM data sets. If VALIDATE is specified, it will be ignored during processing of encrypted VSAM data sets.

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.

For more information about the RACF FACILITY class profile, refer to [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 42](#). 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 IGDSMSxx PARMLIB member.

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

For more information about the RLS timeout value used during DFSMSdss operations, refer to [z/OS DFSMSdss Storage Administration](#).

For more information about the SETSMS command, refer to [z/OS MVS System Commands](#).

Backing up data sets with extended attributes

When you use DFSMSdss to back up data sets with the extended attributes variable DS1EATTR set in the VTOC, you must make sure that these data sets can be restored into an environment that supports them. If the vendor attributes in the F9 DSCB are essential to the validity of the data set, make sure that the environment in which they might be restored has EAVs to support these F9 fields.

Backing up UNIX files

With the DUMP command, you can dump individual z/OS UNIX files residing within a zFS to a sequential data set, which can be a generation data set within a generation data group (GDG). The storage medium for the sequential data set can be tape or DASD. When the output resides on DASD, it can be a basic, large, or extended format data set. There is currently no cloud object storage support.

When you create backups of files 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 can code:

```
DUMP PATH ( -  
    INCLUDE ('dir1/dir2/foo.txt')) WORKINGDIRECTORY('/u/user/ernestof') -  
    OUTDDNAME(TAPE1, TAPE2, TAPE3)
```

This technique can be helpful if you want to create several backups for replication or 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, DFSMSdss ends without completing any backups.

Hard links

Hard links are regular files that are linked to the same set of data (resolve to the same inode). DFSMSdss processes the user data for each regular file that is encountered, and for hard-links, this results in the same data being processed for as many file names that resolve to the same set of data. Each instance of the name-data pair results in separate file open that disallows write access for the duration of the data being processed (close operation).

Sparse files

Sparse files are regular files that have an apparent size that is larger than the occupied disk size. Blocks within the file have no user data (binary zeros) and are not written to disk. The desire for a backup and recovery application is to identify and maintain a sparse file sparse. DFSMSdss supports sparse files for backup and restore and maintains the occupied space of the file when CLONE keyword processing is used to back up the data. If CLONE is not specified for backup, DFSMSdss is unable to maintain a sparse file sparse.

Last backup date

DFSMSdss leverages the z/OS UNIX file Logical File System (LFS) attributes structure (ATTR – as mapped by BPXYATTR macro) during its processing. The file's ATTRREFTIME64 attribute is used to hold the last backup date when the RESET keyword is specified during backup.

Note: If you are using DFSMSdss to backup files that DFSMSHsm is also backing up, you should not use the RESET keyword because it can cause confusion as to which backup is most current, or have other undesired side effects while using DFSMSHsm for your files.

Backup with CLONE processing

Programming Interface Information

DFSMSDss provides the CLONE function to allow you to backup regular file data while minimizing the amount of time in which the data is unavailable. The caller 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. You can invoke DFSMSDss via batch or application program interface (API) to perform such a backup. After clone initialization, DFSMSDss releases the file it holds and, if the NOTIFYCLONE keyword is specified, prints a message to SYSPRINT and to the console stating the clone initialization is complete.

If you invoke DFSMSDss through the API, DFSMSDss informs the caller through the UIM exit option 44 of the clone initialization request resulting return and reason code as returned from zFS.

End Programming Interface Information

Dump processing of a regular file can be performed from a read-only snapshot image which reflects the state of the file at the time the snapshot was created with use of the CLONE keyword. Snapshots encompass the regular file only and cannot be requested at the directory level to initiate a snapshot of all files within the directory at the same time.

When CLONE is specified, all regular files that are selected are cloned prior to processing. This enables the caller or invoking applications to resume access to those files after their clone is initialized instead of after the data movement of the file has completed. If a clone initialization fails, the error is reported and DFSMSDss does not attempt other backup methods to process the file.

Note:

1. zFS allows a single snapshot image for each set of user data (inode). Since DFSMSDss clones all regular files that are specified, hard links are not supported for clone processing. If a hard link is encountered during Clone initialization (CLONE(REQUIRED)) the operation fails.
2. Clone processing results in blocks being the unit of I/O.

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.

For information about using DFSMSDss to back up Linux® for IBM Z partitions and volumes, refer to Chapter 12, “Dumping and restoring Linux for IBM Z partitions and volumes,” on page 169.

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 to specify it. Deallocated tracks are not dumped. The following example shows how you can specify the DUMP command to physically back up a volume:

Physical volume dump to an object storage cloud

When creating a backup in object storage cloud, the CLOUD specification is used instead of the OUTDDNAME keyword. The OUTDDNAME and the CLOUD keywords are mutually exclusive.

It's not possible to perform a physical data set restore from a full volume image when it is in object storage cloud.

Backing up system volumes

If you plan to use the DFSMSDss stand-alone restore program to restore a volume without the use of a host system environment, you must dump the volume physically. In addition, when doing a full physical volume dump to back up a system residence volume, you must use JCL to invoke DFSMSDss.

You cannot use the DFSMSDss stand-alone restore program with an encrypted tape. DFSMSDss does not interface with the Encryption Key Manager or the Tape Controller and therefore the correct keys cannot be provided to the controller to decrypt data. If you attempt to use a stand-alone restore with an encrypted tape, DFSMSDss issues message ADRY3501I to indicate that the dump data set resides on an encrypted tape and thus, cannot be read with the stand-alone restore program. DFSMSDss also issues message ADRY509D to prompt the operator to continue or end the function.

Backing up VM-format volumes

You can use DFSMSDss to back up VM-format volumes that are accessible to your z/OS 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. You cannot use the DFSMSDss stand-alone restore program to restore dumps of VM-format volumes.

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. The OPTIMIZE keyword does not apply when performing UNIX file processing.

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. When backing up VSAM encrypted data sets using logical data set dump, backup utilization may increase since DFSMSDss is dumping up to the high allocated relative byte address (RBA).

Note: During logical dump and restore, encrypted indexed VSAM data sets are processed at a track level. This prohibits DFSMSDss from performing VALIDATE processing on encrypted VSAM data sets during logical dump operations.

Note: When creating a backup to an object storage cloud the OPTIMIZE, COMPRESS, HWCOMPRESS and ZCOMPRESS keywords are not supported. Since the data movement of the user data is offloaded to the

z/OS host to the storage controller the optimization and compression algorithms within DFSMSdss cannot be applied to the backup.

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.

For more information about full volume copy, FlashCopy, and SnapShot, refer to [“Moving volumes” on page 98](#).

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

These keywords are described in the sections that follow.

DUMPCONDITIONING — Allows you to make a copy of the source volume in a full volume copy operation—including volume index information—while keeping the target volume online. Use this keyword when you want to create a copy of the source volume for backup purposes, rather than to allow applications to use the target volume.

With DUMPCONDITIONING in effect, the volume serial number of the target volume does not change, and the target volume remains online after the copy. The VVDS and VTOC index names on the target volume do not change to match the target volume serial number; they continue to match the source volume serial number.

In Step “2” on page 57, for example, you can include the DUMPCONDITIONING keyword on the full volume copy command to allow the target volume to remain online for dumping.

The target of a full volume copy operation using DUMPCONDITIONING is referred to as a *conditioned* volume. A full volume dump of a conditioned volume appears as if it were dumped from the original source volume of the copy operation. However, if the conditioned volume is copied back using DUMPCONDITIONING, conditioning is not performed on the original source volume. Instead, DFSMSdss recognizes that it is copying from the target of a previous conditioned-backup and recovers the original source volume.

For example, suppose that you specify the DUMPCONDITIONING keyword when you perform a full volume copy of volume VOL001 to volume VOL002. If you then perform a full volume dump of VOL002 to tape, the output appears as if you had dumped VOL001 directly. Now suppose that you copy VOL002 back

to VOL001. Here, the VOL002 volume serial number is not copied to VOL001's volume label, because DFSMSdss treats VOL002 as a copy of VOL001.

This example 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 would not look as if it was dumped from the original source volume VOL001. Rather, it would 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 — Using these two keywords in your procedure is recommended when you use FlashCopy.

- You can specify the FCNOCOPY keyword on the COPY command to prevent the ESS subsystem from performing a full physical copy of the volume. Doing so can save subsystem resources and can help to avoid affecting the performance of other I/O operations done by the ESS subsystem.
- You can specify the FCWITHDRAW keyword on the DUMP command to cause DFSMSdss to withdraw the FlashCopy relationship after the volume has been successfully dumped. Doing so frees the subsystem resources that are used to maintain the FlashCopy relationship.

During DUMP FULL and DUMP TRACKS operations, DFSMSdss invokes ICKDSF to initialize the source volume of the DUMP operation at the end of dump processing, when all of the following conditions are true:

- FCWITHDRAW is specified
- The VTOC tracks on the source volume of the DUMP operation are the target of a FlashCopy relationship or the volume is dump conditioned
- If TRACKS is specified, it designates one extent range that represents the entire volume
- The volume is not a VM-format volume (CP volume)
- The volume supports data set FlashCopy or space efficient FlashCopy.

Note: If you perform a dump operation with FCWITHDRAW specified, and the dump source volume is shared between multiple systems, ensure that the DASD is offline to all systems except the one performing the dump.

Timing is an important factor in the successful use of these keywords. In your procedure, allow only a short amount of time to elapse between the completion of step 2 (copy function) and the start of step 4 (backup function). Here, 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

- For cases in which the FlashCopy target volume could not be initialized during FCWITHDRAW processing, message ADR288W is issued and FCWITHDRAW processing is not performed. This prevents leaving the volume in an indeterminate state. You can consider adding return code checking after your DUMP FULL/TRACKS job steps to perform an ICKDSF INIT when the return code of the DUMP FULL/TRACKS job step ends in a return code of 4.
- The FCWITHDRAW keyword is not supported for volumes attached at device address X'0000'.

Related information

For more information about the DUMPCONDITIONING keyword, refer to [“DUMPCONDITIONING” on page 327](#).

For more information about the FCNOCOPY keyword, refer to [“FCNOCOPY” on page 334](#).

For more information about the FCWITHDRAW keyword, refer to [“FCWITHDRAW” on page 407](#).

For information about ICKDSF, refer to [Device Support Facilities \(ICKDSF\) User's Guide and Reference](#).

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 the optimum block size for the output device (262 144 is the maximum). You can change this default to 32 760 by using the installation options exit routine. Refer to the discussion of system-determined block size in [z/OS DFSMS Using Data Sets](#) for a description of the optimum block sizes of the supported tape devices.

For more information about the installation options exit routine, refer to [z/OS DFSMS Installation Exits](#).

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.

Note:

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.
 - When dumping to DASD, specify an output dump data set that is extended format in the compressed format. Avoid specifying the `COMPRESS` or `HWCOMPRESS` keywords when the output dump data set is extended format in the compressed format. Performance could be degraded since the data may be compressed twice.

Performance considerations

This information 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. To obtain the best performance with DFSMSdss and 3592's, 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 z/OS data spaces to contain track image copies of the data being processed by the DFSMSdss. The 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

DFSMSDss might use virtual concurrent copy for storage devices that support SnapShot or data set FlashCopy when the CONCURRENT keyword is specified with the DFSMSDss commands. During virtual concurrent copy, data is "flushed" or "snapped" from the source location to the intermediate location, and then copied to the target location using standard I/O. The operation is logically complete after the source data is "flushed" or "snapped" to the intermediate location and physically complete after the data is moved to the target media.

Virtual concurrent copy using SnapShot

Before you can use the SnapShot function for virtual concurrent copy, you must ensure that working space is available by allocating working space data sets (WSDS) on one or more volumes in the same RAMAC Virtual Array (RVA) subsystem as the source data sets. System Data Mover (SDM) uses the working space data sets as the intermediate location for virtual concurrent copy.

The naming convention for these working space data sets is:

SYS1.ANTMAIN.Ssysname.SNAPnnnn.

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.

The SnapShot working space data sets must be physical sequential, non-extended-format, single volume, and cataloged. SDM performs a numerically sequential catalog search for each data set, starting with *hlq*.ANTMAIN.Ssysname.SNAP0001 until a data set is found or until it encounters a catalog locate error, indicating that the data set was not found. SDM does not use working space data sets with the naming convention beyond the data set that was not found. The SnapShot working space data sets can be SMS-managed or non-SMS-managed. You cannot allocate VSAM or multivolume data sets as SnapShot working space data sets.

If you want to allocate secondary space, you must extend the data set by filling it with data before you start the DFSMSdss job. System Data Mover (SDM) does not extend a working space data set. SDM holds an enqueue for the data set when the SnapShot operation uses working space data set and releases the enqueue after SnapShot operation finishes using the data set. You can reallocate or extend a working space data set only when SDM does not have the data set enqueued. SDM uses the new reallocated or extended data set on subsequent runs of the SnapShot operation.

You can add more working space data sets after ANTMAIN completes the initialization process. SDM uses these data sets the first time it encounters an out-of-working-space condition 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.Ssysname.SNAP0001.

The LRECL and block size can 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.

Virtual concurrent copy using FlashCopy

Before you can use the FlashCopy function for virtual concurrent copy, you must ensure that working space is available by allocating working space data sets (WSDS) on one or more volumes in the same, data set FlashCopy enabled, storage subsystem as the source data sets. System Data Mover (SDM) uses the working space data sets as the intermediate location for virtual concurrent copy.

The naming convention for using these working space data sets is:

hlq.ANTMAIN.FCWKnnnn

Variable *hlq* is the high level qualifier that you specify in the SDM PARMLIB member and *nnnn* is a four-digit decimal number in the value range 0000-9999. If you use both VSAM and physical sequential data sets, you must specify unique *nnnn* component of the name across both kinds of data sets.

The working space data sets must be cataloged. SDM performs a catalog search for usable working space data sets that match the naming convention. You must allocate data sets as single-volume, non-indexed VSAM data sets such as LDS and ESDS, or non-extended-format sequential data sets. You can use extended-format non-indexed VSAM data sets. The data sets can be SMS-managed or non-SMS-managed.

If you want to allocate secondary space, you must extend the data set by filling it with data before you start the DFSMSdss processing. System Data Mover (SDM) does not extend a working space data set. SDM holds an enqueue for the data set when the FlashCopy operation uses working space data set and releases the enqueue after FlashCopy operation finishes using the data set. You can reallocate or extend a working space data set only when SDM does not have the data set enqueued. SDM uses the new reallocated or extended data set on subsequent runs of the FlashCopy operation.

You can add more working space data sets after ANTMAIN completes the initialization process. SDM uses these data sets the first time it encounters an out-of-working-space condition. When this condition occurs during a FlashCopy operation, SDM refreshes the list of working space data sets by performing a catalog search for data set names that match the naming convention.

The LRECL and block size can be any valid combination. The VSAM control interval (CI) size can be any value. SDM uses the tracks within the data set as the target of FlashCopy operations, and you should not try to access them using normal data access methods.

Common working space data sets considerations

To ensure that unauthorized users cannot access sensitive data, IBM recommends that your installation use RACF, or an equivalent security product, to protect the working space data sets.

You must allocate data sets on a volume in each storage subsystem that you are using for virtual concurrent copy. If you define more than one device type on the storage subsystem, you must allocate a working space data set on each device type that contains a data set that you intend to process using SnapShot or FlashCopy.

You must allocate at least one working space data set, if a system or device type for concurrent copy operation runs simultaneously from more than one system and accesses data on the same storage subsystem. For example, you must allocate three working space data sets to process data on an RVA subsystem from three z/OS systems, on devices of each device type containing data processed with concurrent copy.

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

For more information about virtual concurrent copy and working space data sets, refer to [*z/OS DFSMS Advanced Copy Services*](#).

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 database 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.

For more information about the installation options exit, refer to [*z/OS DFSMS Installation Exits*](#).

Clone processing

When processing UNIX regular files during dump, you can get a copy of your data at a specific time by specifying the CLONE keyword. Clone processing initializes a read-only snapshot image which reflects the state of the file at the time the snapshot was created.

Clone initialization is performed after the time it takes DFSMSdss to complete filtering. Clones are requested for all regular files that are selected for processing. After clone initialization is complete, the files are released and are available for user access. DFSMSdss backs up the data from the read-only clone.

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.

Backing up and restoring volumes with incremental FlashCopy

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. Incremental FlashCopy operates at the full volume level.

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.

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. If the new target is the source of a Multiple Incremental FlashCopy relationship (Incremental FlashCopy Version 2), before reversing the direction of the FlashCopy, you must withdraw any other relationships. 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.

Using the FCINCREMENTAL keyword

You can use the FCINCREMENTAL keyword for a COPY FULL or COPY TRACKS CPVOLUME commands 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 copies the changed tracks in the new direction specified on the INDDNAME/INDYNAM and OUTDDNAME/OUTDYNAM keywords. The new direction can be the same or the reverse of the original (existing) direction.

Using the FCINCREMENTALLAST keyword

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 is to be non-persistent and change recording is to be stopped after the final increment has been established. The FlashCopy relationship ends when background copy for the final increment has completed.

Using the FCINCRVERIFY keyword

You can use the FCINCRVERIFY(NOREVERSE | REVERSE) keyword to verify that the existing Incremental FlashCopy direction is what you expected. DFSMSdss fails the copy attempt if the existing direction is not as expected or the original source volume has existing Incremental FlashCopy relationships.

Using the FCWAIT keyword

When you reverse the direction of an Incremental FlashCopy, the storage facility requires that the previous background copy be 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.

Note:

1. Incremental FlashCopy Version 2 supports up to the maximum number of full volume flashcopies allowed for a source, which is 12. The initial Incremental FlashCopy support, known as Incremental FlashCopy Version 1, allows only 1 full volume FlashCopy to be incremental. A Version 1 Incremental FlashCopy relationship can coexist with Version 2 Incremental FlashCopy relationships.
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 cannot 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. Before performing a subsequent incremental copy using DFSMSDss, the offline volume's volume serial number must be changed through a utility such as ICKDSF and the volume must be varied online.
6. The PURGE keyword might 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 does not inhibit target writes. 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 (here, we use a query interval value of 30 seconds and 10 retries) 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
FCWAIT(30,10)
```

Optionally, FCINCRVERIFY(REVERSE) can be specified to verify that the Incremental FlashCopy direction is what was expected:

```
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.

For more information, refer to [“FCINCREMENTAL” on page 331](#), [“FCINCREMENTALLAST” on page 332](#), [“FCINCRVERIFY” on page 333](#), and [“FCWAIT” on page 338](#).

For more information about Incremental FlashCopy and Persistent FlashCopy, refer to [z/OS DFSMS Advanced Copy Services](#).

Securing your tape backups

Data encryption is an important tool for protecting against the possible misuse of confidential information that could occur should tapes be lost or stolen. Unless the possessor of the tape has the required key, any encrypted data on the tape will remain confidential and will be unreadable. Thus, securing tape backups should be part of your installation's overall security plan.

You can secure your tape backups either through *tape device encryption* (with an encryption-capable tape drive) or through *host-based encryption* — that is, by requesting that DFSMSdss encrypt the data before writing it to the dump data set on DASD or tape (during a DUMP or COPYDUMP operation).

With tape device encryption, you secure data by using encryption capable tape drives that encrypt data as the output is written to tape during DUMP and COPYDUMP functions. To request that a tape device encrypt data, you specify JCL DD keywords or DATACLAS definitions for the output DFSMSdss dump data set. You can restore tape device encrypted dump data sets through the RESTORE and COPYDUMP functions.

DFSMSdss allows a mixture of encrypting and non-encrypting tape devices for the output of DUMP and COPYDUMP commands.

In general, you can use tape device encryption or software encryption to encrypt a particular tape volume, but not both methods. If you request software encryption and one of the output devices is encrypting the data written to it, then DFSMSDss overrides software encryption. DFSMSDss does not write the data to any of the output devices that do not encrypt data and issues an ADR519E message indicating that some outputs are not processed. DFSMSDss issues the ADR519E message for each of the output data sets that are not backed up. DFSMSDss continues processing.

If you do not request software encryption, DFSMSDss writes to multiple outputs where some are encrypting data in the hardware and some are not. DFSMSDss continues to notify an application through its Application Programming Interface (Exit 06 and Exit 26) that the particular output is encrypting the data written to it.

Observe the following considerations:

- Use tape device encryption if your installation includes one or more encryption-capable tape drives. Here, you specify by data class which data is to be encrypted when stored on the tape drives.
- Use host-based encryption if you do not have an encryption-capable tape drive. You can encrypt tape backups through the host-based encryption method described in the sections that follow.
- Encryption eligible sequential basic and/or large format data sets cannot be used as output of the DUMP command.

For a description of how DFSMSDss handles conflicting encryption requests, refer to [“DFSMSDss processing of dump encryption requests” on page 71](#).

Using host-based encryption to secure backups

When backing up your data, you can secure it through host-based encryption. You request host-based encryption on the DFSMSDss DUMP command. As with tape device encryption, host-based encryption through DFSMSDss provides a means of encrypting your installation's tape volumes.

Usually, DFSMSDss does not allow double encryption of data with the DUMP command. For a description of how DFSMSDss handles conflicting encryption requests, refer to [“DFSMSDss processing of dump encryption requests” on page 71](#).

In the unlikely event that your installation requires double encryption for a dump data set, you can use the procedure described in [“If double encryption is required” on page 72](#) to dump and restore a doubly-encrypted data set.

Types of host-based encryption

DFSMSDss can use the following types of host-based encryption to secure your data:

- Triple-length Data Encryption Standard (TDES) clear keys
- Secure TDES keys
- 128-bit Advanced Encryption Standard (AES) clear keys.

What is DES and AES?

To manage cryptographic keys for encrypted data, DFSMSDss uses IBM Cryptographic Services Facility (ICSF), which 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).

With 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 uses TDES triple-length keys and 128-bit AES keys for host-based 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.

Considerations for host-based encryption

The choice of which type of host-based encryption to use depends on several factors, including performance and level of security. On the DFSMSdss DUMP command, you can request the different types of host-based encryption:

- A clear TDES key; specify the CLRTDES subparameter of the ENCRYPT keyword
- A secure TDES key; specify the ENCTDES sub parameter of the ENCRYPT keyword
- A clear 128-bit AES key; specify 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 require.

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.

The CLRAES128 option generates a 128-bit AES key. The key value can appear in application storage. If DFSMSdss is running on a z9 or z10 processor, the data is encrypted using CPACF. If DFSMSdss is running on any other type of processor, the data is encrypted by ICSF.

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 passwords 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.

Using compression with host-based encryption

When archiving large amounts of encrypted data, you can compress 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 prior to 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.

Examples of host-based encryption

In the following example, DFSMSdss is used 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(CLRDES) RSA(SYSTEM.PRIVATE.S01024) -  
HWCOMPRESS OPTIMIZE(4)
```

The following is an example of the RESTORE command you would use to restore the data on a system that has the same RSA private key with the same label:

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

The following 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)
```

The following 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 and decryption

If you plan to use the RSA option with DFSMSdss to encrypt the data-encrypting key, you must consider the cryptographic hardware that exists at the site that will decrypt the data. Not all types of RSA private keys are supported by all types of cryptographic hardware.

Table 7 on page 70 summarizes the RSA private tokens and required cryptographic hardware for decryption.

<i>Table 7. RSA private tokens and required cryptographic hardware for decryption</i>	
RSA private key token (internal)	Required cryptographic hardware
RSA private key token 1024 – Modulus-Exponent Internal form	One of the following: <ul style="list-style-type: none"> • Cryptographic Coprocessor feature • PCI X Cryptographic Coprocessor • Crypto Express2 Coprocessor.
RSA private key token 1024 – Chinese Remainder Theorem Internal form	One of the following: <ul style="list-style-type: none"> • PCI Cryptographic Coprocessor • PCI X Cryptographic Coprocessor • Crypto Express2 Coprocessor.
RSA private key token 2048 – Chinese Remainder Theorem Internal form	One of the following: <ul style="list-style-type: none"> • PCI Cryptographic Coprocessor with LIC January 2005 or later, and z/OS ICSF HCR770B or later • PCI X Cryptographic Coprocessor • Crypto Express2 Coprocessor.

For more information, refer to [z/OS Cryptographic Services ICSF Application Programmer's Guide](#).

Performance and processor types

The performance of host-based encryption can vary, depending on the type of processor and encryption being used. For example, DFSMSdss can use the Cipher Message with Chaining (KMC) IBM Z instruction for some types of encryption if it is running on the appropriate processor. If the KMC instruction is not available or the type of encryption requires it, DFSMSdss uses the appropriate ICSF service to perform the encryption.

The table below describes the method of encryption that is used under various encryption types and processors.

Processor 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 and decryption

To perform encryption and decryption, DFSMSdss requires the following software to be installed and active on your system:

- Encryption Facility DFSMSdss Encryption Feature (HCF773D)
- IBM Cryptographic Services Facility (ICSF) (HCR770B or higher).

ICSF callable services for DFSMSdss

DFSMSdss invokes ICSF callable services for the DUMP command. If your installation uses RACF or a similar security product, ensure that the security administrator authorizes DFSMSdss to use the following services and any cryptographic keys that are specified as input:

- 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.

Similarly, ensure that the security administrator authorizes DFSMSdss to use the following ICSF callable services and cryptographic keys 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.

For information about ICSF callable services, refer to [*z/OS Cryptographic Services ICSF Application Programmer's Guide*](#).

DFSMSdss processing of dump encryption requests

You can use tape device encryption, or host-based encryption to encrypt a tape volume, but you cannot use both methods. When dumping to DASD, you can use host-based encryption or you can dump to an encrypted-format data set, but you cannot use both methods. Because DFSMSdss avoids performing double encryption of data, you must determine which type of encryption, if any, is to be used for your output. In general, DFSMSdss prevents you from combining two types of encryption to perform double encryption of dump output.

Table 8 on page 72 shows how DFSMSdss processes potential double encryption requests, specified through the DFSMSdss DUMP command.

Table 8. DFSMSdss processing of dump encryption requests

Dump encryption request	DFSMSdss action
Your DUMP command specifies host-based encryption (through the RSA or KEYPASSWORD keywords), and all of the available tape drives are encryption-capable, or output dump data sets are encrypted-format.	<ul style="list-style-type: none"> DFSMSdss issues informational message ADR518I to indicate that host-based encryption was bypassed due to an encrypting tape device or an encrypted-format dump data set.
Your DUMP command specifies host-based encryption, and that one or more of the available outputs are not encrypting tape drives or encrypted-format data sets.	<ul style="list-style-type: none"> DFSMSdss issues error message ADR519E to indicate that one or more of the available outputs will not be encrypted. To avoid performing double encryption of data, DFSMSdss uses only encryption-capable tape drives or encrypted format dump data sets. DFSMSdss issues error message ADR324E to list the unused output devices. DFSMSdss continues processing the DUMP request while there are usable output devices. On completion, DFSMSdss ends the task with return code 8.
Your DUMP command does not specify host-based encryption and all of the available outputs are either encryption-capable tape drives or encrypted-format data sets.	<ul style="list-style-type: none"> Encryption is performed on encryption-capable tape drives or on encrypted-format data sets.
Your DUMP command does not specify host-based encryption, and one or more of the available outputs specified are not encryption capable.	<ul style="list-style-type: none"> DUMP requests for encryption-capable tape drives are encrypted by the tape drives DUMP requests for non-encrypting tape drives are processed by the access method. DUMP requests for non-encrypting tape drives are processed without encryption of any type.

For dump output that require host-based encryption, ensure that your dump-requesting jobs only use non-encrypted-format dump data set, and tape drives that are not encryption capable. To do so, check the data classes of the output *ddnames* to ensure that the jobs do not specify a data class that requests encryption on the output.

If double encryption is required

In the unlikely event that your installation requires double encryption for a dump data set, you can use the following procedure:

1. Request host-based encryption for the data set (through the RSA or KEYPASSWORD keywords) and write the data set to a non-encrypting output device.
2. Use the DFSMSdss COPYDUMP command to copy the dump data set to an encrypting tape device.

To restore the double-encrypted dump data set, use the DFSMSdss RESTORE command. The encryption capable tape drive decrypts the dump data set, and the DFSMSdss performs host-based decryption for the data set.

If double encryption is required

In the unlikely event that your installation requires double encryption for a dump data set, you can use the following procedure:

1. Request host-based encryption for the data set (through the RSA or KEYPASSWORD keywords) and write the data set to a non-encrypting output device
2. Use the DFSMSdss COPYDUMP command to copy the dump data set to an encrypting tape device.

To restore the double-encrypted dump data set, use the DFSMSdss RESTORE command. The encryption capable tape drive decrypts the dump data set and then DFSMSdss performs host-based decryption for the data set.

Restoring data sets

With the RESTORE command, you can restore data to DASD volumes from DFSMSdss-produced dump volumes. DFSMSdss backups can be identified by using either the INDDNAME keyword or the CLOUD 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.

Attention: You should restore a dumped data set that has extended attributes in an F9 DSCB to a volume that supports F8/F9 DSCBs. Otherwise, these extended attributes are lost. DFSMSdss propagates the vendor attributes if they exist in the F9 DSCB of the primary volume when DFSMSdss performs catalog processing or if they exist in the first volume that DFSMSdss processes when you specify input volumes. To prevent losing extended attributes, all volumes that contain data sets with vendor attributes in the F9 DSCB must be extended address volumes.

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).

For more information about filtering, refer to [Chapter 16, “DFSMSdss filtering—choosing the data sets you want processed,”](#) on page 255.

For information about using automatic class selection (ACS) routines during DFSMSdss restore operations, refer to [Chapter 11, “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. Otherwise, DFSMSdss will attempt to restore the data set to 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 information about restoring such data sets, refer to [“Restoring indexed sequential, unmovable, direct, and absolute track data sets”](#) on page 81 and [“Restoring integrated catalog facility catalogs”](#) on page 80.

To use a preallocated data set, you must specify the REPLACE or REPLACEUNCONDITIONAL keyword. 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
 - Key length (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.
- The target data set has the IMBED or REPLICATE attributes. It will be reallocated without those attributes.

You may use data set RESTORE to upgrade your basic format sequential data sets to large format data sets. To do this, simply preallocate a large format data set. In restoring a basic format sequential data set, if 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 is scratched and reallocated as a large sequential data set. In restoring a large format data set, if a preallocated basic format sequential data set is found, the basic format sequential data set is used and is upgraded to a large format data set. If the preallocated basic format sequential data set is not large enough to hold the data being restored, it is scratched and reallocated as a large format data set.

If a user wishes to downgrade a large format data set to a basic format sequential data set, restore the large format data set with no preallocated target, allocate a basic format sequential data set, and use a utility such as IEBGENER or IDCAMS REPRO to copy the data from the large format data set to the basic format sequential data set.

If a user wishes to upgrade or downgrade an extended format sequential data set you must preallocate the target data set to the extended format version number.

During logical restore processing, 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.

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.
- If VSAM, the cluster types match
- The DSORG matches.

- If the data set being processed is multivolume or single volume and the non-SMS preallocated data set matches the multivolume or single volume.

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.OLDSDS)) -
      RENAME(*.OLDSDS,*.NEWSDS)
```

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 RENAMEUNCONDITIONAL. 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.

Restoring data sets with the IMBED or REPLICATE attributes

During logical restore processing, DFSMSdss converts data sets with IMBED or REPLICATE attributes to indexed key-sequenced data sets (KSDS) without the IMBED and REPLICATE attributes. If the target data set already exists and has these attributes, it is deleted and reallocated without the IMBED and REPLICATE attributes.

Restoring data sets with the KEYRANGES attribute

During logical restore processing, DFSMSdss restores data sets with the KEYRANGES attribute as they were dumped. DFSMSdss does not convert these data sets, but issues message ADR508I to bring attention to their existence.

DFSMSdss handling of the expiration date during logical restore

For preallocated targets, the expiration date of the preallocated target is preserved. For non-preallocated targets, the expiration date depends on whether the data set is VSAM or non-VSAM, whether the source data set is SMS-managed, and whether the target data set is SMS-managed. SMS also ensures that the expiration date conforms with the target's management class retention period.

Allocating to SMS

For VSAM data sets, DFSMSdss uses the expiration date from the source catalog entry 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 is zero.

For non-VSAM data sets, DFSMSdss uses the expiration date from the source VTOC 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 modifies the date to conform with the management class.

Allocating to non-SMS

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

For non-VSAM data sets, DFSMSdss uses the expiration date from the source VTOC 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.

DFSMSdss handling of the data-set-changed indicator during restore

When restoring a data set, understand that the value of the data-set-changed indicator depends on several factors, such as whether you invoke DFSMSdss directly or through its Application Programming Interface (API), and whether you rename the data set during the restore operation.

Table 9 on page 77 shows how DFSMSdss handles the data-set-changed indicator in each of these situations.

Table 9. DFSMSdss handling of the data-set-changed indicator during data set restore operations.		
How is DFSMSdss invoked?	Is the data set renamed?	Result
Invoked directly to restore the data set.	No	DFSMSdss sets the data-set-changed indicator in the VTOC to the value the data set had when it was backed up.
Invoked directly to restore the data set.	Yes	DFSMSdss turns off the data-set-changed indicator for the data set because it is newly allocated.
Invoked by an application through the DFSMSdss API.	No	DFSMSdss leaves the data-set-changed indicator unmodified, so that other backup applications can continue to track the data set.
Invoked by an application through the DFSMSdss API.	Yes	DFSMSdss leaves the data-set-changed indicator unmodified, so that other backup applications can continue to track the data 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.

Note:

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, ensure that the data set is cataloged in the same catalog from which it was dumped. When recataloging multi-volume VSAM data sets, ensure that the order of the volumes is maintained. 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.

Coexistence considerations

For information about restoring dumps created with previous releases, refer to “Restoring backups using DFSMSdss” on page 647 in [Appendix A, “Coexistence considerations,”](#) on page 647.

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 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.

For information about when DFSMSDss scratches and reallocates target data sets, refer to [“Restoring to preallocated targets” on page 82](#).

Copying or restoring multivolume data sets

When you copy or restore multivolume data sets, be aware of the following:

- DFSMSDss does not preserve candidate volumes. (This includes the guaranteed-space candidate space 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.

Restoring integrated catalog facility catalogs

Integrated catalog facility (ICF) user catalogs can be logically restored only to volumes with the same device type as the volumes from which they were dumped. With physical data set restore, ICF user catalogs can be restored only to the same volumes (volumes with 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.

Note: DFSMSdss cannot be used to restore a master catalog that is active on any system.

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 might become inaccessible.

An integrated catalog facility user catalog can be restored dynamically. Catalog recovery jobs must be modified to include any of the following options:

- Add the IDCAMS ALTER LOCK|SUSPEND command to lock or suspend the existing catalog before the DFSMSdss restore operation. After the recovery is complete, unlock or resume the catalog by using IDCAMS ALTER UNLOCK|RESUME. The LOCK or SUSPEND attribute on the dump tape is used if the catalog does not exist.
- Issue the F CATALOG,RECOVER,LOCK|SUSPEND(ucat) command before the DFSMSdss restore operation. After the recover is completed, unlock or resume the catalog by using IDCAMS ALTER UNLOCK|RESUME. The LOCK or SUSPEND attribute on the dump tape is used if the catalog does not exist.
- Specify the BCSRECOVER(LOCK|SUSPEND) keyword in the DFSMSdss RESTORE command. DFSMSdss performs the necessary unlock and resume during internal processing. This option applies only to existing (pre-allocated) catalogs. If the catalog does not exist, DFSMSdss defines the catalog as locked to ensure that the catalog is not accessible before DFSMSdss completes restore processing.
- For more information about the BCSRECOVER parameter, see [“BCSRECOVER” on page 463](#).
- For more information about using the MODIFY CATALOG command, see [z/OS DFSMS Managing Catalogs](#).

The following example shows the JCL used to restore an integrated catalog facility user catalog. If the master catalog is protected with RACF, 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 5 on page 81](#) shows the printed output that is 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)-FRLB0(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 5. Output from Restore of Integrated Catalog Facility User Catalog

For more information about the LOCK attribute and the access authority, see [z/OS DFSMS Managing Catalogs](#).

For more information about the installation authorization exit routine, see [z/OS DFSMS Installation Exits](#).

When you attempt to replace an existing integrated catalog facility (ICF) user catalog, it must not violate any of the following restrictions:

- The number of extents for the target must not exceed the number of extents of the data set being restored.
- If the ICF user catalog was defined with the SHARE keyword, the target high index used CI must not exceed the high index used CI of the data set being restored.
- If the ICF user catalog was defined with the SHARE keyword, the target high index allocated CI must not exceed the high index allocated CI of the data set that is being restored.
- The target ICF user catalog's high used relative byte address must not exceed the high used relative byte address of the data set being restored.
- The target ICF user catalog's high allocated relative byte address must not exceed the high allocated relative byte address of the data set being restored. A violation of these restrictions results in a failure to restore the ICF user catalog.

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 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.

For more information about using BDAM processing options with DFSMSdss keywords, refer to [“RESTORE command for DFSMSdss”](#) on page 451.

For more information about the installation options exit, refer to [z/OS DFSMS Installation Exits](#).

For information about using DFSMS macros for non-VSAM data sets, refer to [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 an extended-format VSAM data set with stripe count of one

When performing a logical restore operation of an extended-format VSAM data set with a stripe count of one, the resulting target will remain a VSAM data set with a stripe count of one, even if the target storage class is multi-stripped.

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 the Device Support Facilities program (ICKDSF) BUILDIX command.

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. The PDSE's original version level is preserved during restoration if restored on z/OS V2R1.0 or later. The version level that was dumped stays the same when the PDSE is restored, if restored on the same release of z/OS that it was dumped from. PDSE member generations are also preserved if restored on z/OS V2R1.0 or later.

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.

End Programming Interface Information

For information about writing ACS routines, refer to [Chapter 18, “Syntax - DFSMSDss function commands,”](#) on page 267.

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: You cannot use the VOLCOUNT keyword to convert the following types of data sets to multivolume:

- TTR-BDAM or unmovable data sets. If DFSMSDss encounters an existing multivolume TTR-BDAM or unmovable data set, a DADSM error occurs.
- Partitioned data sets (PDS or PDSE). If an existing multivolume PDS or PDSE data set is encountered, it is converted to single-volume.

- You cannot restore a non-SMS-managed non-VSAM single volume data set that was dumped with DFSMSdss to multivolume.

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.

Note that the RESTORE command does not invoke the data class ACS routine, so no data class is assigned. (The dataset attributes are already defined, therefore the DATACLAS routine is not called because it's not relevant in this process.) To cause the data class ACS routine to be invoked, create a new dataset either with the IDCAMS DEFINE command or a JCL DD statement with DISP=(NEW,CATLG).

For more information about the variables available to ACS routines during restore processing, refer to [“ACS variables available during RESTORE and CONVERTV processing” on page 166](#).

For more information about using the RESTORE command to convert data to and from SMS management, refer to [Chapter 8, “Converting data to and from SMS management,” on page 135](#).

For more information about ACS routines, refer to [Chapter 11, “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 refer to 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 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 Distributed FileManager) can be restored to non-SMS-managed target volumes but the DFM attributes will be lost and a warning message will be issued.

For information about DFM, see [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.

Using the DELETECATALOGENTRY keyword

You can use the DELETECATALOGENTRY keyword to cause DFSMSdss to perform 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 to preformat it. Open processing requires the data set to be cataloged in the standard catalog search order. Thus, to restore a preformatted empty VSAM data set, the target data set must be cataloged in the standard catalog search order.

Restoring UNIX files

With the Restore command, you can restore UNIX files to a mounted zFS from DFSMSDss-produced dump data sets. DFSMSDss backups containing UNIX files can be identified by using the INDDNAME keyword only. There is currently no cloud object storage support.

You specify what files are to be restored by specifying the path in the INCLUDE statement. The path must at least be a partial path of what was specified at DUMP time, starting from the beginning of the path. See example below:

If the following was specified during Dump.

```
DUMP PATH ( -  
    INCLUDE ( 'dir1/foo.txt' '/dir1/dir2/foo.txt' ) -  
    WORKINGDIRECTORY('/u/usr/ernestof') -  
    OUTDDNAME(TAPE1)
```

Example 1: The following could be specified during Restore:

```
RESTORE PATH ( -  
    INCLUDE ( 'dir1/dir2' ) WORKINGDIRECTORY('/u/usr/zaid') -  
    INDDNAME(TAPE1)
```

Example 2: The following would result in an error because the path does not exist in the backup.

```
RESTORE PATH ( -  
    INCLUDE ( 'dir2/foo.txt' ) WORKINGDIRECTORY('/u/usr/zaid') -  
    INDDNAME(TAPE1)
```

During Restore, the WORKINGDIRECTORY keyword specifies the target location where the path is to be restored. In Example 1, 'dir1/dir2/foo.txt' is restored to the 'u/usr/zaid' directory.

Determining what files are in a backup

DFSMSDss lets you find out what files reside in a backup without actually restoring any data. You can do this by specifying TYPRUN=NORUN on the JCL EXEC PARM field.

DFSMSDss does not generally support wild card filtering during Dump or Restore of z/OS UNIX file processing. The following was enabled so that users can determine what files reside in a backup.

Example:

```
//RESTORE EXEC PGM=ADRDSSU,PARM='TYPRUN=NORUN'
//DD9XWRK DD DSN=TDS.DUMP2,UNIT=3390,
//          VOL=SER=D9XWRK,
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
RESTORE PATH(INCLUDE('*')) -
          WORKINGDIRECTORY('/ernestof') INDD(DD9XWRK)
/*
)
```

Note:

1. TYPRUN=NORUN is not required. It is possible to restore everything that resides in a backup to the target WORKINGDIRECTORY.
2. When performing this invocation, there should only be a single entry in the INCLUDE statement. If more paths are specified, this results in a failure.

Restoring to preexisting files

DFSMSDss supports overwriting existing regular files only. If an existing regular file is encountered, the Restore operation will fail unless the REPLACEUNCONDITIONAL keyword is specified. Both the source file type and target file type must be regular files. Regular file support of REPLACEUNCONDITIONAL is necessary to preserve hard links.

For directory files, In the following invocation:

```
RESTORE PATH( -
          INCLUDE( -
            '/dir1/dir2/dir3')) -
          WORKINGDIRECTORY('/ernestof') INDD(DD9XWRK)
```

This request specifies dir3 as the file to be restored. dir1/dir2 are directories along the path.

During Restore, any directory along a path to a file that is requested to be restored can be re-created with the attributes of the source directory at the time it was backed up. If the directories exist, no attributes are restored to the preexisting directories. This behavior only applies to the directories along the path to the file to be restored. In the preceding example, if directory dir3 exists, the operation fails. REPLACEUNCONDITIONAL keyword only applies to regular files.

Hard links

If any hard links are encountered during the Dump operation, DFSMSDss processes the data as many times as a file name resolves to the same data. This results in data duplication residing in the backup.

DFSMSDss does not require that every hard-link is specified during a Restore operation. Instead, it allows you to specify at least one. Since it is not known which file name is to be requested during restore, each file name has its associated user data.

If more than one hard-linked regular file is requested to be restored, only the first file results in data being processed. Any subsequent file that DFSMSDss encounters that resolves to a previously restored hard-link (as indicated in the backup), results in a hard-link being created to the first file that was restored. This results in no data duplication on the restore operation.

DFSMSDss does not know the state of files outside of those that reside in the backup. If a hard-linked regular file is to be restored because of an inadvertent change to a z/OS UNIX file, it is necessary to restore the same regular file (path) while specifying REPLACEUNCONDITIONAL. This ensures that any other files in z/OS UNIX which are hard-linked to the same file are kept in sync. If the file is deleted (unlinked) from the directory prior to the file being restored, only that directory link to the inode is broken and there can be other file links that still point to the corrupted data.

Last backup date

DFSMSDss leverages the z/OS UNIX Logical File System (LFS) attributes structure (ATTR – as mapped by BPXYATTR macro) during its processing. The file's ATTRREFTIME64 attribute is used to hold the last backup date.

When restoring a regular file, it is important to understand that the value of the last backup date depends on many factors. The following table describes how DFSMSDss handles the last backup date in priority order:

Table 10. Last Backup Date Handling	
IF...	Result
ADMINISTRATOR keyword is specified	DFSMSDss sets the ATTRREFTIME64 (last backup date) to the source value in the backup.
WORKINGDIRECTORY is the same location as specified during dump. (i.e. restoring to the same location as the source)	DFSMSDss sets the ATTRREFTIME64 (last backup date) to the source value in the backup.
WORKINGDIRECTORY is different location as specified during dump.	DFSMSDss does not set the ATTRREFTIME64 (last backup date). 1. If the file is pre-existing, the file's last backup date persists as it was prior to the restore operation. 2. If the file does not pre-exist the value results in zeros.

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.

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
```

Note:

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.

For information about using DFSMSDss to restore Linux for IBM Z partitions and volumes, see [Chapter 12, “Dumping and restoring Linux for IBM Z partitions and volumes,”](#) on page 169.

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 associated with retained locks or data in the coupling facility, a full-volume or tracks restore can result in data integrity problems.

When restoring data in a full volume or tracks operation, DFSMSdss resets the data-set-changed indicator in the VTOC for each restored data set. This action indicates that the data set has not changed since the previous backup.

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 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: If you perform a full-volume restore to DASD that is shared between multiple systems, you must ensure that the DASD is offline to all systems except the system that is performing the restore.

When performing a full-volume restore operation to DASD, DFSMSdss automatically corrects the free-space information on the volume and might invoke ICKDSF to rebuild the VTOC index. DFSMSdss takes this action when it copies data to a larger-capacity DASD from a tape dumped from a smaller-capacity DASD or when 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. You can ignore any IEC614I messages that DFSMSdss generates during this process.

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,uuu,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 about CAS allocation, see [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 DFSMSdss stand-alone restore program 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. The stand-alone restore program allows you to recover system volumes so that you can start the host environment.

You can also use the DFSMSdss stand-alone restore program in a VM environment. The stand-alone restore program operates in ESA/390 mode, ESA/370 mode, or System/370 XA mode.

You cannot use the stand-alone restore program with an encrypted tape. If you attempt to do so, DFSMSdss issues message ADRY0513I to indicate that the dump data set resides on an encrypted tape and thus, cannot be read with the stand-alone restore program. DFSMSdss also issues message ADRY509D to prompt the operator to continue or end the function.

You cannot use the stand-alone restore program with a dump that has been encrypted with the ENCRYPT keyword or compressed with the HWCOMPRESS or ZCOMPRESS keywords. If you attempt to do so, DFSMSdss issues message ADRY501I to indicate that the dump is not supported with the stand-alone restore program. The facilities required to process encrypted or compressed data do not exist in the stand-alone environment.

For more information about how to perform a restore using DFSMSdss stand-alone services, see [“DFSMSdss stand-alone services”](#) on page 515.

Recovering VM-format volumes

You can use DFSMSdss to recover VM-format volumes that are accessible to your z/OS 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.

Coexistence considerations

For information about restoring dumps created with previous releases, refer to [“Restoring backups using DFSMSdss” on page 647 in Appendix A, “Coexistence considerations,” on page 647.](#)

Chapter 7. 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.

For more information about using the DUMP command to back up data, see [Chapter 6, “Managing availability with DFSMSdss,”](#) on page 39.

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. Physical processing involves volumes, tracks and the parts of data sets that reside on a particular volume. 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 Chapter 11, “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.

Attention: You should restore a dumped data set that has extended attributes in an F9 DSCB to a volume that supports F8/F9 DSCBs. Otherwise, these extended attributes are lost. DFSMSdss propagates the vendor attributes if they exist in the F9 DSCB of the primary volume when DFSMSdss performs catalog processing or if they exist in the first volume that DFSMSdss processes when you specify input volumes. To prevent losing extended attributes, all volumes that contain data sets with vendor attributes in the F9 DSCB must be extended address volumes.

Note: Concurrent copy operation fails and a message is issued if the DELETE keyword is specified.

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 data set copy

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

```
COPY -  
  DATASET( INCLUDE( USER.**)) -  
  RENAMEUNCONDITIONAL( USER2 )
```

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: When processing input volumes, DFSMSdss 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.

Physical data set copy

If you specify DATASET and the PHYSINDDNAME or PHYSINDYNAM keywords, DFSMSdss performs a physical data set copy. This method of moving data sets on a per volume basis only allows data movement between like devices. Single volume data sets will be copied from the source volume to the target volume. Single volume non-VSAM SMS managed data sets will be cataloged when they are either renamed or the DELETE keyword was specified. Multivolume data sets must be copied a volume at a time, and then recataloged by the user.

A conditioned volume created through the DFSMSdss COPY FULL DUMPCONDITIONING command can be specified as a source volume. A non-conditioned volume can also be specified as the source volume. However, the target volume cannot be a conditioned volume.

The following is an example of the syntax to specify a COPY command that results in a physical data set copy operation. You can specify only one volume on the PHYSINDD or PHYSINDYNAM keyword.

```
COPY DATASET(INCLUDE(**)) -
      PHYSINDDNAME(DASD1) OUTDYNAM(VOLS02) -
      REPLACE
```

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 order of search.

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 119.)

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

For non-SMS output volume selection, DFSMSdss selects volumes based on size of the allocation necessary. The first volume that has enough space will be used for the allocation. When moving data sets with the extended attributes variable DS1EATTR set in the VTOC, the system selects non-SMS output volumes as follows:

1. EATTR=NO: All volumes will be considered for allocation.
2. EATTR = OPT: For both VSAM and non-VSAM data sets, DFSMSdss prefers EAVs when the data set organization and type is supported in the EAS and the data set's size is greater than the BPV. EATTR = OPT is the default. If EATTR is not specified, it is the same as EATTR = OPT.

Preferring EAVs means that all volumes that are EAVs will be evaluated before any non-EAVs are even considered. If the EAVs in the list of output volumes cannot satisfy the allocation request, all of the output volumes will then be evaluated to accommodate the allocation request.

Note: The BPV is derived, in the following order, from the Storage Group attribute, the IGDSMSxx parmlib member, and the system default of 10 cylinders. The BPV can be changed dynamically in the storage group definition or changed with the SETSMS BPV operator command to override the IGDSMSxx parmlib member. However, for non-SMS volume selection, there will be no Storage Group attribute to evaluate.

Renaming data sets

You can rename data sets by using the RENAMEUNCONDITIONAL (RENAMEU) keyword with the COPY and RESTORE commands. For VSAM data sets, you can only rename clusters. DFSMSdss derives the new names for the components of VSAM clusters as follows:

- If the operation is a logical data set copy or restore and the data set being processed is a linear data set or the operation is a physical data set copy or restore, 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 conditions 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 or the operation is a physical data set copy or restore
 - 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:

Instance	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:

Instance	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:

Instance	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:

Instance	Cluster Name	Data Component Name
Old	SYS1.IODF00.DATASET	SYS1.DFSMS
New	SYS2.IODF00.DATASET	SYS2.IODF00.DATASET.D

- If renaming the data set results in a component name that exceeds 44 characters in length, DSS replaces any DATA or INDEX specific qualifier with "D" or "I" respectively.

Example: When RENAMEU (NEWNAM) is specified, the following configuration occurs:

Instance	Cluster Name	Data Component Name
Old	IBM.DFSMS.DSS.LARGE.VSAM.DSNAME.TEST	IBM.DFSMS.DSS.LARGE.VSAM.DSNAME.TEST.DATA
New	NEWNAM.DFSMS.DSS.LARGE.VSAM.DSNAME.TEST	NEWNAM.DFSMS.DSS.LARGE.VSAM.DSNAME.TEST.DA TA

- 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.

Note: These examples represent common renaming scenarios. A combination of renaming rules might apply, depending on the source and target names of any given cluster or component.

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 about defining RACF profiles, see [Chapter 21, “Data security and authorization checking,” on page 533](#).

Moving data sets with utilities

In some cases, DFSMSdss invokes a utility to move a data set. [Table 11 on page 105](#) 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 order of search.

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 11. 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

Note:

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. For help in determining the volume count, [“VOLCOUNT” on page 357](#).
 - 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 either the source HURBA=HARBA or it has extended indexes.

Moving data sets with concurrent copy

Programming Interface Information

The DFSMSDss concurrent copy 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 concurrent copy 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 the UIM exit option, Eioption 24 (for more information, see [“CONCURRENT” on page 320](#)). The application can resume normal operation at this time.

End Programming Interface Information

If for any reason data cannot be processed with concurrent copy (for example, the hardware being used does not support concurrent copy), DFSMSDss optionally uses another method of data movement and does not release the serialization until the copy is completed.

If the source device supports data set FlashCopy or SnapShot, DFSMSDss can use FlashCopy or SnapShot function to provide a concurrent copy-like function called virtual concurrent copy.

Specifying concurrent copy for COPY requests

On the DFSMSDss COPY command, you can specify that DFSMSDss is to use the concurrent copy function to process data. To do so, you specify the CONCURRENT keyword, and, optionally, one of several available sub-keywords to indicate the type of concurrent copy to be used and whether DFSMSDss can use other methods of data movement. If you do not specify the CONCURRENT keyword, your COPY request does not use concurrent copy.

The CONCURRENT keyword applies to all of the data being copied. You cannot apply this function to a subset of the data being processed.

If you specify the CONCURRENT keyword, DFSMSDss might use a function equivalent to cache-based concurrent copy, called virtual concurrent copy. During virtual concurrent copy, data is "flushed" or "snapped" from the source location to an intermediate location, and then copied to the target location through standard I/O. The operation is logically complete after the source data is "flushed" or "snapped" to the intermediate location and physically complete after the data is moved to the target media.

If the source volume supports data set FlashCopy, DFSMSDss uses FlashCopy to provide virtual concurrent copy. If the source volume is a RAMAC Virtual Array (RVA), DFSMSDss uses SnapShot. For more information about virtual concurrent copy, see [“Using concurrent copy” on page 7](#).



Attention: Use concurrent copy only during periods of light update activity for the data sets or volumes involved. Performing cache-based concurrent copy operations against many large data sets when there is also heavy update activity (such as reorganizing data sets or initializing the volume the data sets reside on) might result in a shortage of storage, because data is transferred to z/OS data space storage faster than DFSMSDss can process it. When you use multiple simultaneous concurrent copy tasks to process large, heavily updated data sets, you might also experience long run times and contention for SYS.DATA.SPACE.LATCH.SET. You must ensure that during the concurrent copy operation another system does not reserve volumes that are to be processed. You must also ensure that jobs and address spaces that are to use the concurrent copy are assigned a WLM service class with a high execution velocity. Do not assign a discretionary goal to concurrent copy work. You should spread multiple concurrent copy jobs across as many LPARs as possible and avoid the use of PARALLEL mode in DFSMSDss.

Note:

1. To help ensure data integrity, do not update the data during concurrent copy initialization.
2. If a concurrent copy operation fails after signaling that concurrent copy initialization is complete (and update activity on the data has resumed), you cannot recover the data to the point-in-time at

which the concurrent copy operation was started. The data might have been updated while the copy operation was progressing.

3. Performing cache-based concurrent copy operations against many large data sets when there is also heavy update activity (such as reorganizing data sets or initializing the volume the data sets reside on) might result in a shortage of storage. The shortage occurs because data is transferred to z/OS data space storage faster than DFSMSdss can process it.
4. If DFSMSdss invokes a utility such as IDCAMS REPRO or IEBCOPY for a data set copy operation, DFSMSdss does not perform concurrent copy.
5. VM mini-volumes are supported if you are using RVA devices to the extent that they are supported by IBM Extended Facilities Product (IXFP) device reporting.
6. The use of concurrent copy and virtual concurrent copy with the DFSMSdss COPY command is controlled by the RACF FACILITY class profile, STGADMIN.ADR.COPY.CNCURRNT.

For information about specifying CONCURRENT and the other COPY command keywords, refer to [“COPY command for DFSMSdss” on page 300](#).

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 (data set FlashCopy).
- 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. An extended-format VSAM data set in which the stripe count and volume count do not match is considered a multi-layered data set and cannot use FlashCopy.
 - **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 RELBLOCKADDRESS 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.

DFSMSdss FlashCopy Batch Protection allows DFSMSdss to direct a fast replication request to the storage group ACS routine. The ACS routine then assigns a storage group to allocate the data set during a logical data set COPY to SMS operation. This enables users, without modifying existing batch jobs, to have a set

of volumes dedicated for FlashCopy usage, limiting situations in which a volume is selected that does not allow FlashCopy to be used (such as when a volume is a Global Mirror volume or a z/OS Global Mirror (XRC) primary).

For FlashCopy Batch Protection, add the following statement to the storage group ACS routines:

```
IF &ACSENV2 = 'FLASHCPY' THEN
DO
    SET &STORGRP = 'fcstrgrp'
    EXIT
END
```

where *fcstrgrp* is a new or existing storage group containing volume serials that DFSMSdss is to select for allocation.

For more information, refer to [Chapter 11, “ACS routine information,” on page 165](#).

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 for reasons other than normal reasons, DFSMSdss issues message ADR945W and error message ADR938E, which indicates that the processing of the current extent failed. DFSMSdss terminates DEFRAG processing. If the volume involved in the DEFRAG or CONSOLIDATE operations reside on a storage subsystem that is cascaded FlashCopy capable, FlashCopy failures for normal reasons will no longer occur except for the case when the maximum number of relations would be exceeded.

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.

For more information about the FASTREPLICATION keyword, refer to [“FASTREPLICATION” on page 329](#).

Preserve Mirror FlashCopy

You can choose to allow the target volume of a FlashCopy operation to be a Peer-to-Peer Remote Copy (PPRC) primary device. When the tracks associated with the FlashCopy relationship are copied to the PPRC secondary device, the PPRC (Metro Mirror) pair goes into a duplex pending state, to ensure the integrity of the mirror between the local site and the remote site. When the FlashCopy operation completes, the PPRC_SYNC volume pair returns to full duplex state.

IBM Remote Pair FlashCopy (also known as Preserve Mirror) mirrors the FlashCopy command that is issued at the local site, to the remote site. This allows FlashCopy operations to occur to PPRC primary volumes without affecting the PPRC duplex state.

When you specify the FCTOPPRCPPrimary keyword on the COPY command, you are requesting that DFSMSdss allows a PPRC primary volume to become the target volume of the FlashCopy operation. You can specify the following sub-keywords to indicate whether the PPRCP mirror is allowed to go to duplex pending state if the target volume of the FlashCopy operation is a metro mirror primary device:

PRESMIRREQ

specifies that if the target volume is a Metro Mirror Primary device, the pair must not go into a duplex pending state as the result of a FlashCopy operations.

PRESMIRPREF

specifies that if the target volume is a Metro Mirror primary device, it would be preferable that the pair does not go into a duplex pending state as the result of a FlashCopy operation. However, if a Preserve Mirror operation cannot be accomplished, the FlashCopy operation is still to be performed.

The PRESMIRPREF option is not valid for DS888x and newer, and compatible, storage subsystems. If the PRESMIRPREF option is specified and the volumes involved in the operation reside on DS888x or newer storage subsystems, the command results in a warning or error message. This is also true of the corresponding option provided in byte 33 (X'21') in the ADRUFO data area.

PRESMIRNONE

specifies that Preserve Mirror operation is not to be done, even if all of the configuration requirements for a Preserve Mirror operation are met. If the target specified is a Metro Mirror primary device, the pair is to go into a duplex pending state while the secondary device is updated with the tracks to be copied. PRESMIRNONE is the default if you specify FCTTOPPRCPPrimary without a subkeyword.

Note: If the target volume of FlashCopy operation is not a metro mirror primary volume, then the FCTOPPRCPPrimary keyword has no effect on the FlashCopy operation.

For more information about Peer-to-Peer Remote Copy (PPRC) and metro mirror operation, refer to [z/OS DFSMS Advanced Copy Services](#).

For more information about the FCTOPPRCPPrimary keyword on the COPY command, refer to [“COPY command for DFSMSdss”](#) on page 300.

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 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 or Preserve Mirror was not used. When you specify FASTREPLICATION(REQUIRED), DFSMSdss issues an informational message in addition to the ADR938E message, whether you have specified the DEBUG(FRMSG(MIN | SUM | DTL)) keyword or not.

Note:

1. The DEBUG(FRMSG) keyword might not have an effect if the target of the FlashCopy operation is not a PPRC primary device
2. If you specify FASTREPLICATION(REQUIRED) without specifying the DEBUG keyword, DFSMSdss still issues an informational message whenever a fast replication method cannot be used.
3. The DEBUG(FRMSG) keyword overrides the DEBUG=FRMSG parameter specified on the JCL EXEC statement.

For more information about the DEBUG keyword, refer to [z/OS DFSMSdss Storage Administration](#).

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:

- Performing a logical data set dump of the target data sets (of the data set copy) and specify the FCWITHDRAW keyword on the DUMP command.
- Entering 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.

For an overview of FlashCopy and more information about the FCWITHDR command, refer to [z/OS DFSMS Advanced Copy Services](#).

For more information, refer to [“FCNOCOPY” on page 334](#), [“FCNOCOPYTOCOPY” on page 334](#), and [“FCWITHDRAW” on page 407](#) keywords.

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 RELBLOCKADDRESS 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.

For more information about the FASTREPLICATION keyword, see [“FASTREPLICATION” on page 329](#).

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.

For more information about the DEBUG keyword, see [“DEBUG” on page 323](#).

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. For more information about RACF FACILITY class profiles, see [z/OS Security Server RACF Security Administrator's Guide](#).

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.

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 RENAMEUNCONDITIONAL 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. For a description of the LOCK attribute and the correct access authority, see [z/OS DFSMS Managing Catalogs](#).

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 RENAMEUNCONDITIONAL 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

Copying or restoring multivolume data sets

When you copy or restore multivolume data sets, be aware of the following:

- DFSMSDss does not preserve candidate volumes. (This includes the guaranteed-space candidate space 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, RENAMEUNCONDITIONAL, 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, RENAMEUNCONDITIONAL, 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:

- If DFSMSdss cannot use fast replication, it must invoke IDCAMS to copy an extended-format VSAM data set. Refer to <https://www.ibm.com/docs/en/zos/3.1.0?topic=dfsmsdss-moving-data-sets-flashcopy> for reasons why DFSMSdss cannot use fast replication.
- 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.
- When performing a logical copy operation of an extended-format VSAM data set with a stripe count of one, the resulting target will remain a VSAM data set with a stripe count of one, even if the target storage class is multi-striped.
- 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.

The version level of a PDSE remains the same after a move if restored on z/OS V2R1.0 or later – the resulting data set keeps the same version level of the original data set if copied on z/OS V2R1.0. PDSE member generations are also preserved if copied on z/OS V2R1.0 or later.

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.

The following DFSMSdss keywords implement the processing options (for details on their use, see [“Explanation of RESTORE command keywords” on page 462](#)):

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 more information, see *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 (see [“AUTORELBLOCKADDRESS” on page 462](#)).

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 RENAMEUNCONDITIONAL 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 RENAMEUNCONDITIONAL), 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 12 on page 118 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 12. 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 86](#) and [“Changing management class with restore processing” on page 87](#)).

End 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 directed to 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.

To understand how the data might be placed on the volumes selected when you copy a multivolume guaranteed-space data set, refer to [“Copying or restoring multivolume data sets” on page 114](#).

End 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 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 Programming Interface Information

Moving non-SMS-managed data sets

If the data set being moved is to be non-SMS-managed, use the NULLSTORCLAS and BYPASSACS keywords on the COPY command. By using these keywords, you can copy an SMS-managed data set into a non-SMS-managed data set. Using NULLSTORCLAS and BYPASSACS also prevents a non-SMS-managed data set from becoming SMS-managed. When copying a VSAM data set to a non-SMS-managed volume, ensure that a VVDS exists on the volume prior to running the job, to prevent potential allocation errors because of insufficient space. DFSMSdss assumes a VVDS exists on the volume when doing size calculations on non-SMS-managed volumes.

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.

If a user wishes to upgrade or downgrade an extended format nonVSAM sequential data set you must preallocate the target data set to the extended format version number.

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 that are 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 is scratched and reallocated if it is renamed.

You can use data set COPY to upgrade your basic format sequential data sets to large format data sets. When you copy a data set and a usable preallocated target is found, it is used as the target of the copy operation. When you copy a basic format sequential data set and a preallocated large format data set is found, it is used. If the preallocated large format data set does not have enough space for the source data, it is scratched and reallocated as a large format data set. When copying a large format data set and a basic format sequential data set is found, it is used and upgraded to a large format data set. If the preallocated basic format sequential data set does not have a large enough allocation to hold the source data, it is scratched as a large format data set.

If a user wishes to downgrade a large format data set to a basic format sequential data set, allocate a basic format sequential data set and use a utility such as IEBGENER or IDCAMS REPRO to copy the data from the large format data set to the basic format sequential data set.

For PDSEs, the source data set's original version level is preserved after the copy or move, regardless of the version level of the preallocated target data set. For example, if the preallocated target data set is a PDSE with version level 2, and the source data set is a PDSE with version level 1, after a copy or move the target would be a PDSE with level 1.

The extended-format, compression, and encryption attributes of the source and target non-VSAM data sets must match.

- If the source is a non-extended-format non-VSAM data set, then the target must be a non-extended-format non-VSAM data set.
- If the source is an extended-format non-VSAM data set, then the target must be an extended-format non-VSAM data set.
- If the source is a compressed and/or encrypted non-VSAM data set, then the target must be a compressed and/or encrypted non-VSAM 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 IGDSMSxx PARMLIB member.

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

For more information about using IGDSMSxx to control the RLS timeout value used during DFSMSdss operations, refer to [z/OS DFSMSdfp Storage Administration](#).

For more information about using the SETSMS command, refer to [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 order of search. Therefore, to copy a preformatted empty VSAM data set, the target data set must be cataloged in the standard order of search.

VTOC considerations for moving volumes

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. If you do not expand the VTOC when moving to a larger volume, DFSMSdss logical data set processing might fail. The following two sections describe how the size of the target VTOC is affected by DFSMSdss processing.

You can also use the REFORMAT EXTVTOC or REFORMAT NEWVTOC functions of ICKDSF to extend or reallocate the VTOC on a volume if it is not large enough.

When performing a full-volume restore operation to DASD, DFSMSdss automatically corrects the free-space information on the volume and can invoke ICKDSF to rebuild the VTOC index. DFSMSdss takes this action when it copies data to a larger-capacity DASD from a smaller-capacity DASD or when 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. You can ignore any IEC614I messages that DFSMSdss generates during this process.

Following a COPY or RESTORE operation, the VTOC location or the volume serial on the target volume may change. Before this volume can be accessed on any remote system, the UCBs on the remote systems must be refreshed. The refresh occurs automatically if the volume is online and the device manager REFUCB function is enabled. You enable the REFUCB function through PARMLIB member DEVSUPxx or the MODIFY DEVMAN command. For more information, refer to the description of the REFUCB keyword in *z/OS MVS Initialization and Tuning Reference* or *z/OS MVS System Commands*.

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 111). 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 Chapter 11, “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

Note: If you perform a full-volume copy operation to a DASD that is shared between multiple systems, ensure that the DASD is 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.

For information about automatic class selection (ACS) routines during DFSMSdss copy operations, see [Chapter 11, “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.

For more information about using the ADMINISTRATOR, ALLDATA, ALLEXCP, and PURGE keywords, see [“Explanation of COPY Command Keywords” on page 315.](#)

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.

For more information about the FASTREPLICATION keyword, see [“FASTREPLICATION” on page 329](#).

Determining why FlashCopy cannot be used

There might 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 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.

For more information about the DEBUG keyword, see [“DEBUG” on page 323](#).

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:

- Performing a full volume dump of the target volume, specifying the `FCWITHDRAW` keyword on the `DUMP` command.
- Entering the `TSO FCWITHDR` command.

DFSMSdss also issues an `FCWITHDRAW` with the Delete Data Space Withdraw (DDSW) option to the target volume during `COPY` and `RESTORE` commands using `FULL` and `TRACKS` operations.

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.

For more information about using the SETSMS command, refer to *z/OS MVS System Commands*.

For more information about using the SETSMS command, refer to *z/OS MVS System Commands*.

For more information, refer to “FCNOCOPY” on page 334, “FCNOCOPYTOCOPY” on page 334, and “FCWITHDRAW” on page 407.

Choosing space efficient FlashCopy

A *space efficient volume* does not have all of its physical space allocated when it is created. Instead, its physical space is allocated in a way that is dependent on the type of space efficient volume in use.

Track space efficient (TSE)

Physical space is allocated on a track basis. When data is written to a TSE volume, a track of physical space is taken from the segments that are assigned to a repository volume, and is used to hold the data for the TSE volume. A repository volume can provide the physical space for multiple space efficient volumes. These types of volumes are intended for limited use.

Extent space efficient (ESE)

Physical space is allocated on an extent basis. When data is written to an ESE volume, a physical space extent is taken directly from an extent pool as writes occur. Extent sizes vary.

DFSMSdss can use extent space efficient volumes as the source or target of a FlashCopy relationship, or track space efficient volumes as the target of a FlashCopy relationship. These types of FlashCopy relationship are called space efficient FlashCopies. Track space efficient volumes have limitations that are described below.

Allowing track space efficient volume use with the FCSETGTOK keyword

During full volume copy operations, DFSMSdss can use a track space efficient volume as the target of a FlashCopy relationship, when you specify the FCSETGTOK keyword on your COPY command.

You can use track space efficient FlashCopy for a full-volume copy operation only; that is, a COPY FULL operation or a COPY TRACKS command that specifies a full volume. Observe the following considerations:

- If you specify FCSETGTOK with COPY FULL, and the target is a track space efficient volume, DFSMSdss attempts to establish a full-volume FlashCopy relationship without excluding free space, which results in one FlashCopy relationship for the entire volume. If FlashCopy cannot be used, DFSMSdss issues an error message and the copy operation fails. DFSMSdss does not try any other methods of data movement.
- To use FCSETGTOK with COPY TRACKS, your command must specify one track range (an extent) that includes the entire volume (tracks 0 through *n*). Otherwise, the FCSETGTOK keyword has no effect on the copy operation.

DFSMSdss ignores the FCSETGTOK keyword for COPY operations in which:

- FlashCopy is not used to perform the copy operation
- The target volume is not a track space efficient volume
- Less than a full volume is to be copied, for example, a COPY DATASET operation.

Along with the FCSETGTOK keyword, you must also specify the FAILRELATION sub-keyword to indicate the action that the storage facility is to take if the space on the repository volume is exhausted while the space efficient FlashCopy relationship still exists.

Using FCSETGTOK might require RACF authorization. If your installation has defined the RACF FACILITY class profile, STGADMIN.ADR.COPY.FCSETGT, your user ID requires READ access to the profile. For more information, see “Protecting DFSMSdss functions with RACF FACILITY class profiles” on page 37.

Note: Space efficient FlashCopy to a track space efficient volume is intended for full volume copies that are short term in nature, such as those that are to be backed up to tape. Space efficient FlashCopy to

a track space efficient volume might also be appropriate for longer term copies, if the source and target volumes are not frequently updated. The physical background copy option is not permitted for space efficient FlashCopy to a track space efficient volume. That is, you must also specify FCNOCOPY with the FCSETGTOK keyword.

For an overview of FlashCopy, refer to *z/OS DFSMS Advanced Copy Services*.

For more information about the FCSETGTOK keyword and the FAILRELATION sub-keyword, refer to [“FCSETGTOK” on page 335](#).

Initializing the volume with the FCWITHDRAW keyword

During DUMP FULL and DUMP TRACKS operations, DFSMSdss invokes ICKDSF to initialize the source volume of the DUMP operation at the end of the dump processing, when all of the following conditions are true:

- FCWITHDRAW is specified
- The VTOC tracks on the source volume of the DUMP operation are the target of a FlashCopy relationship. Initializing the volume with the FCWITHDRAW keyword or the volume is dump conditioned
- TRACKS, if specified, designates one extent range that represents the entire volume
- The volume is not a VM-format volume (CP volume)
- The volume supports data set FlashCopy or space efficient FlashCopy.

For DFSMSdss users performing DUMP FULL or TRACKS specifying the FCWITHDRAW keyword, if DFSMSdss cannot perform an ICKDSF INIT, ADR288W is issued and the job step return code will be 4. Users of DFSMSdss need to be aware that the job step that previously ended with a return code of 0 will now end in a return code of 4.

If these conditions are not met, DFSMSdss performs a FlashCopy withdraw operation only.

For information about how to disable the volume initialization function, refer to [“Changing the default initialization processing during DUMP with FCWITHDRAW \(OA18929\)” on page 230](#).

For information about the FCWITHDRAW keyword, refer to [“FCWITHDRAW” on page 407](#).

Backing up volumes with FlashCopy consistency group

The sections that follow describe the use of a 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 CGCREATED 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.

Note: Because I/O activity is held on source volumes that are frozen, it is recommended *not* to include system volumes that are required to run the current COPY command (or subsequent COPY commands) needed to form a FlashCopy Consistency Group. Examples of such system volumes include spool, page, and volumes containing checkpoint data sets, catalogs, and RACF databases.

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 CGCREATED operation

When all volume copy operations have completed, you can use the DFSMSdss CGCREATED 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 CGCREATED 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, or select one volume from each LSS.

Example

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 CGCREATED 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 CGCREATED 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
CGCREATED FCCGVFY(SRC101) ACCVOL(SRC101,SRC203)
/*
```

Note:

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 CGCREATED 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).
8. During a CGCREATED operation, or a FCFREEZE operation that ends in error, DFSMSdss might issue a FlashCopy withdraw request for the source and target volume. If either volume is attached at device address X'0000', the system fails the FlashCopy withdraw operation with warning message ADR815W. Processing continues.

For more information, about using the FCCGFREEZE keyword on the COPY command, refer to ["FCCGFREEZE" on page 329](#).

For more information about the ACCESSVOLUME and FCCGVERIFY keywords on the CGCREATED command, refer to ["ACCESSVOLUME" on page 275](#) and ["FCCGVERIFY" on page 275](#).

For more information about FlashCopy Consistency Groups, refer to [z/OS DFSMS Advanced Copy Services](#).

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.

For more information about how to use the ADMINISTRATOR, ALLDATA, ALLEXCP, and PURGE keywords, see ["Explanation of COPY Command Keywords" on page 315](#).

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.

For more information about the FASTREPLICATION keyword, see ["FASTREPLICATION" on page 329](#).

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.

For more information about using the `DEBUG` keyword, see [“DEBUG” on page 323](#).

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 might invoke `ICKDSF` to rebuild 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, 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 do not expand the VTOC when moving to a larger volume, DFSMSdss logical data set processing might fail.

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 z/OS 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 8. 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 topic is organized as follows:

- **“Evaluating conversion to SMS management” on page 135** discusses the advantages and disadvantages of the two types of conversion.
- **“Conversion by data movement” on page 136** describes how to use the COPY and DUMP/RESTORE commands to convert data sets *to* SMS management.
- **“Conversion without data movement” on page 137** describes how to use the CONVERTV command to convert volumes *to* SMS management.
- **“Special data set requirements for conversion to SMS” on page 139** describes some of the data sets that have special requirements for conversion *to* SMS management.
- **“Converting from SMS management without data movement” on page 141** describes how to use the CONVERTV command to convert volumes *from* SMS management.
- **“Special data set requirements for conversion from SMS” on page 141** 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

Note:

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.

Note:

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 119.

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).

Note that neither the COPY nor the RESTORE command invokes the data class ACS routine, so no data class is assigned. To cause the data class ACS routine to be invoked, create a new dataset either with the IDCAMS DEFINE command or a JCL DD statement with DISP=(NEW,CATLG).

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. For a list of variables available to ACS routines during CONVERTV processing, see [“ACS variables available during RESTORE and CONVERTV processing”](#) on page 166.

Simulated conversion creates a report that identifies data sets ineligible for conversion. For a sample of this report, see [“SMS report”](#) on page 139. 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. For a list of variables available to ACS routines during CONVERTV processing, see [“ACS variables available during RESTORE and CONVERTV processing”](#) on page 166.

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 6 on page 139 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 6. SMS Report

For details on the messages, refer to *z/OS MVS System Messages, Vol 1 (ABA-AOM)*.

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.

Note:

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 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.

When a data exists on multiple volumes and is non-SMS managed, the catalog entry will show all volsers, even those with no data on them. When a data set only has data on the first volume, and all volumes in

the catalog are specified in the list to be converted, the resultant catalog entry will show only the volume where data resides with all remaining volumes showing as candidate volumes '*'. The original volumes in the catalog that were non-SMS will be converted, assuming all data set on those volumes are eligible to be SMS-managed.

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 RENAMEUNCONDITIONAL is not supported for data sets with DFM attributes. DFM attributes are not maintained for non-SMS data sets.

Chapter 9. Managing space with DFSMSdss

You can use DFSMSdss to help manage your DASD space. This topic discusses how to reclaim DASD space, and how to reduce fragmentation on volumes.

Reclaiming DASD space

You can use DFSMSdss to reclaim DASD space 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 [Chapter 16, “DFSMSdss filtering—choosing the data sets you want processed,” on page 255](#). 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 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.
- The data set has not been opened (DS1LSTAR = 0).

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

You can use the CONSOLIDATE command to consolidate the multi-extent data sets that reside on a single volume and are not excluded from data movement. For eligible data sets that consist of contiguous extents in sequential order, DFSMSdss consolidates data sets without extent relocation. Otherwise, DFSMSdss relocates eligible data set extents if contiguous free space exists on the volume to hold the extents.

The CONSOLIDATE command attempts to consolidate data set extents within a managed space of a volume and perform extent reduction for data sets that occupy multiple extents. When you process a volume with the CONSOLIDATE command, DFSMSdss searches each moveable data set. A data set that either is included or 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 set extents are relocated if enough free space exists on the volume to combine two or more. When DFSMSdss has completed consolidation for all eligible data sets, or has executed the amount of time specified in the MAXTIME option, processing is quiesced.

For information about specifying the CONSOLIDATE command, refer to [z/OS DFSMSdss Storage Administration](#).

Note:

1. The process of combining data set extents can cause the free space to be more fragmented than it was before the operation began.
2. Data set extents are not moved between track-managed space and cylinder-managed space of an extended address volume during CONSOLIDATE processing.
3. Despite the fact that DFSMSdss might perform freespace defragmentation following the consolidation of data set extents, the fragmentation index might be higher following a CONSOLIDATE operation than before the operation began.

As an alternative to CONSOLIDATE, you can use the DUMP command with the DELETE and PURGE keywords to scratch and uncatalog the data sets from DASD after they are dumped. If you restore these data sets to the same DASD, allocation attempts to obtain the space for the entire data set. In general, if the DASD volume has sufficient contiguous unused space, DFSMSdss allocates space in one contiguous extent. If you do not specify ALLDATA and ALLEXCP for sequential and partitioned data sets, only used spaces are allocated.

Attention: Do not use this technique for unmovable data sets such as ABSTR allocated or indexed sequential data sets. DFSMSdss does not delete unmovable data sets and the volume might become more fragmented after combining data sets extents than it was before the operation began. If that happens, you might not be able to restore the data sets.

Use the following steps to dump and delete (scratch and uncatalog) all movable non-VSAM data sets, defragment volumes, and restore all movable non-VSAM data sets.

1. Enter these 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. Enter this control statement to defragment the volume:

```
DEFRAG DDN(DASD1)
```

3. Enter these control statements to restore and catalog all dumped data sets:

```
RESTORE INDD(TAPE1) OUTDD(DASD1) -  
  DATASET(INCLUDE(**)) -  
  CATALOG
```

Enter this control statement to defragment the volume, and perform extent reduction if possible:

```
DEFRAG DDN(DASD1) CONSOLIDATE DDN(DASD1)
```

Consolidating free space and extents 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).

With 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.

Note: Data set extents will not be moved between the track-managed space and cylinder-managed space of an extended address volume during DEFRAG processing.

When to run DEFRAG and CONSOLIDATE functions

You can run DEFRAG and CONSOLIDATE functions on a volume at any time. However, these operations lock the VTOC (through the RESERVE macro) and the VVDS, if it exists on the volume. They also serialize 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 and CONSOLIDATE runs.

DFSMSdss erases the source location for every extent that is moved during the DEFRAG or CONSOLIDATE 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 or CONSOLIDATE operation if the device is in an ESS that supports data set FlashCopy. 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 or CONSOLIDATE 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.

For more information about the FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword, refer to [z/OS DFSMSdss Storage Administration](#).

Preserve Mirror FlashCopy

During DEFRAG and CONSOLIDATE processing, you can choose to allow the target volume of a FlashCopy operation to be a Peer-to-Peer Remote Copy (PPRC) primary device. When the tracks associated with the FlashCopy relationship are copied to the PPRC secondary device, the PPRC (Metro Mirror) pair goes into a duplex pending state, to ensure the integrity of the mirror between the local site and the remote site. When the FlashCopy operation completes, the PPRC_SYNC volume pair returns to full duplex state.

IBM Remote Pair FlashCopy FlashCopy (also known as Preserve Mirror) mirrors the FlashCopy command that is issued at the local site, to the remote site. This allows FlashCopy operations to occur to PPRC primary volumes without affecting the PPRC duplex state.

When you specify the FCTOPPRCPPrimary keyword on the DEFRAG and CONSOLIDATE command, you are requesting that DFSMSdss allows a PPRC primary volume to become the target volume of the FlashCopy operation. You can specify the following sub-keywords to indicate whether the PPRCP mirror is allowed to go to duplex pending state if the volume of the FlashCopy operation is a metro mirror device. Use the following sub-keywords to indicate whether the device pair is allowed to go to duplex pending state if the target volume of the FlashCopy operation is a metro mirror primary device:

PRESMIRREQ

specifies that if the volume is a Metro Mirror Primary device, the pair must not go into a duplex pending state as the result of a FlashCopy operations.

PRESMIRPREF

specifies that if the volume is a Metro Mirror primary device, it would be preferable that the pair does not go into a duplex pending state as the result of a FlashCopy operation. However, if a Preserve Mirror operation cannot be accomplished, the FlashCopy operation is still to be performed.

PRESMIRNONE

specifies that Preserve Mirror operation is not to be done, even if all of the configuration requirements for a Preserve Mirror operation are met. If the volume specified is a Metro Mirror primary device, the pair is to go into a duplex pending state while the secondary device is updated with the tracks to be copied. PRESMIRNONE is the default if you specify FCTTOPPRCPrimary without a subkeyword.

Note: If the volume of FlashCopy operation is not a metro mirror primary volume, then the FCTOPPRCPrimary keyword has no effect on the FlashCopy operation.

For more information about Peer-to-Peer Remote Copy (PPRC) and metro mirror operation, refer to [z/OS DFSMS Advanced Copy Services](#).

For more information about the FCTOPPRCPrimary keyword on the CONSOLIDATE and DEFRAG commands, refer to “CONSOLIDATE command for DFSMSdss” on page 286 and “DEFRAG command for DFSMSdss” on page 371.

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.

For more information about the DEBUG(FRMSG(MIN | SUM | DTL)) keyword, refer to [z/OS DFSMSdfp Storage Administration](#).

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.

For more information about the FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword, refer to [z/OS DFSMSdfp Storage Administration](#).

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 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 `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.

For more information about the `DEBUG(FRMSG(MIN | SUM | DTL))` keyword, refer to [z/OS DFSMSdss Storage Administration](#).

Data sets excluded from DEFRAG or CONSOLIDATE processing

DFSMSdss automatically excludes and does not relocate the following types of data sets in a `DEFRAG` or `CONSOLIDATE` 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 that is associated with them (only the first extent of this type of data set is excluded from the `DEFRAG` operation).
- For `CONSOLIDATE` operation, user specified data sets (do not satisfy `INCLUDE` criteria).

Because the `DEFRAG` and `CONSOLIDATE` functions do not relocate these data sets, the effectiveness of a `DEFRAG` and `CONSOLIDATE` 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 is 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:

DYNALLOC

Uses dynamic allocation to serialize the use of data sets, rather than enqueue. This method does not always provide cross-system serialization.

FRAGI(n)

Performs a DEFRAG operation only if the fragmentation index is more than *n*.

MAXMOVE(n,p)

Stops the DEFRAG run when *n* contiguous free tracks are assembled. If *n* contiguous free tracks already exist, DEFRAG processing attempts to further reduce the fragmentation of the volume, but no more than *n* tracks can be relocated. If more than *n* tracks must be relocated, DFSMSDss does not perform the DEFRAG request.

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 attempt to assemble the tracks.

MAXTIME(nummins)

Stops the DEFRAG operation after the number of minutes specified has passed. This allows you to control the amount of time that the operation can run. MAXTIME is checked after each data set is processed. When the MAXTIME value is reached, the DEFRAG function ends.

nummins

Specifies number of minutes a DEFRAG operation can run.

MMOVPCT(n,p)

Stops the DEFRAG operation when *n*% contiguous tracks on the volume are assembled as free. If *n*% contiguous tracks already exist as free tracks, 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 percentage of tracks on the volume that DFSMSDss is to try to assemble as free tracks in a contiguous area.

p

The number of passes DFSMSDss is to make in attempting to assemble the tracks.

PASSDelay

Specifies the time delay between the passes (p) specified in MAXMOVE(n,p) or MMOVPCT(n,p) to allow access to the volume between passes.

VERSION1

Specifies which version of DEFRAG is executed. During DEFRAG operations, you might want to execute the pre-z/OS V1R10 version of DFSMSDss DEFRAG. VERSION1 only supports volumes that are 65,520 cylinders or less. Any new function added to DEFRAG with z/OS V1R10 or later is not available.

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

- The MMOVPCT keyword is recommended over MAXMOVE when running DEFRAG on an extended address volume.
- To have the DEFRAG function 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 for *n*, or code the MMOVPCT(*n*) parameter using a high percentage amount. When the value is

higher than what the DEFRAG function can assemble, the process stops at the end of the first pass. For example:

```
DEFRAG DYNAM(388002) MAXMOVE(9999)
```

or

```
DEFRAG DYNAM(388002) MMOVPCT(75)
```

- Experimenting with the DEFRAG FRAGI and MAXMOVE or MMOVPCT parameters will allow you to compare results when you use 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. By default, MAXMOVE or MMOVPCT use FRAGI(3), which is the recommended value. To defragment DASD volume 388001, you can use the command, as follows:

```
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 that are provided by message ADR213I. Therefore, the information in message ADR213I might not reflect the state of the volume at the completion of DFSMSdss processing because another job might 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 DFSMSdfp Storage Administration](#).

The DEFRAG command does a RESERVE on the VTOC to serialize access to the VTOC.

The DEFRAG command does a RESERVE on the VTOC to serialize access to the VTOC, and does an ENQUEUE on the SYSZVDS - volser resource to ensure VVDS integrity. The DEFRAG command also serializes access to each data set before relocating the extent of a data set. The enqueue scheme that is used by the DEFRAG function ensures integrity on a single system but does not ensure integrity for data sets on DASD shared between systems. 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 systems except the one on which DEFRAG runs. After the DEFRAG function finishes, you can vary the volume back online for the other systems.
- 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 systems.

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. However, GRS might 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. For the restrictions that apply to enqueues and dequeues, see [z/OS MVS Planning: Global Resource Serialization](#). 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 at least twenty-four hours previously, using a selection age of two days. 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(LIST(CREDIT LE *, -1)). Data set A.B.C is selected because it has now been one day since creation

Because there is a date change between the two actions, A.B.C is selected for the DEFRAG operation.

Using a two-day delay enforces a convention that more than twenty-four 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 are open more than twenty-four hours is low. The following example uses a two-day delay to cause DEFRAG processing only for those data sets on volume SHARE3 that were created and referenced more than twenty-four hours previously and that are not temporary data sets.

```
DEFRAG BY(LIST((CREDIT LE *, -2), (REFDT LE *, -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(CREDIT, 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 twenty-four hours 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(CREDIT, 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) is installed and either:
 - The data set was defined to RACF with the RACF ERASE option.
 - The data set is VSAM.
 - The VSAM has the ERASE attribute.

See Table 13 on page 153 for more information.

<i>Table 13. Non-VSAM Data Set Erase Table with z/OS Security Server (RACF element).</i>		
	RACF Protected	Erased on Scratch
User Install Exit=ERASE (default)	No	No
	Yes (=ERASE)	Yes
	Yes (=NOERASE)	No
User Install Exit=NOERASE	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.

For more information about the installation options exit routine, refer to [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 (unmovable attribute in the DSORG field in the VTOC entry)
- 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
- Encrypted format data sets

Partitioned load modules can be reblocked in copy operations, even if they have note lists.

The reblockable indicator in the data set's VTOC entry also determines whether a data set is to be reblocked. 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'.

For more information about the installation reblock exit routine, refer to [*z/OS DFSMS Installation Exits*](#).

Chapter 10. Diagnosing problems in DFSMSdss operations

You might one day experience a problem with DFSMSdss operations 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 topic 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.

In general, use the following procedure as a guide to help you find a resolution to a 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.

The following sections provide more information about diagnosing and reporting problems:

- [“Determining the source of the failure: DFSMSdfp, DFSMSdss, or DFSMSHsm” on page 155](#)
- [“Using keywords to identify the problem” on page 156](#)
- [“Using the IBM Support Center” on page 163.](#)

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
CGCREATED
COMPRESS
CONSOLIDATE
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 DFSMSdss common services component, which consists of the following three routines:

- Common filter services
- DASD calculation services
- Device information services.

To begin the diagnostic procedure for a DFSMSdss failure, refer to [“Using keywords to identify the problem”](#) on page 156.

For information about the DFSMSdss dump data set, refer to [Chapter 13, “Format of the DFSMSdss dump data set,”](#) on page 179.

For information about the DFSMSdss patch area, refer to [Chapter 22, “Data integrity—serialization,”](#) on page 561.

For information about diagnosing DFSMSdss function failures, refer to [z/OS DFSMSdss Diagnosis](#).

For information about diagnosing DFSMSHsm function failures, refer to [z/OS DFSMSHsm Diagnosis](#).

Using keywords to identify the problem

This topic 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.

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.

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 [z/OS DFSMSdss Diagnosis](#).

Example: 5695DF175

Procedure: The component identification number for DFSMSdss is 5695DF175; see [“Release-level keyword”](#) on page 156.

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: RA10

Procedure: The keyword for z/OS V1R10 DFSMSdss is RA10; see [“Type-of-failure and function keywords”](#) on page 157.

Type-of-failure and function keywords

Select one keyword from [Table 14 on page 157](#) that best describes the type of failure and see the topic 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 14. Summary of Type-of-Failure Keywords		
Keyword	Type of Failure	Topic
ABENDxxx	DFSMSdss ends abnormally because of a system-detected error. Note: The ABENDxxx keyword does not apply to the DFSMSdss stand-alone restore program.	“ABENDxxx” on page 157
MSGADRnnnt	An error is related to a DFSMSdss message.	“MSGADRnnnt” on page 158
WAIT	The program does not seem to be doing anything.	“WAIT” on page 158
LOOP	The program is doing something repetitively.	“LOOP” on page 159
INCORROUT	Output from the program is incorrect or missing.	“INCORROUT” on page 160
DOC	Documentation of the program is in error.	“DOC” on page 160
PERFM	Program performance is degraded.	“PERFM” on page 161

ABENDxxx

Use this procedure when DFSMSdss ends abnormally. However, if the abend is user abend "0001," skip this topic, and go directly to [“MSGADRnnnt” on page 158](#).

Note: Because there are no abends in the DFSMSdss stand-alone restore program, this procedure does not apply to it.

Example: 5695DF175 RA10ABENDxxx *function*

Procedure: To determine the keywords for type-of-failure and function, follow these instructions:

1. Replace the xxx in the ABENDxxx 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 161](#).

- 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)-mmmm(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 “2.a” on page 157. For example:

```
ADR101I (ttt)-mmmm(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 ABENDxxx type-of-failure keyword. Refer to [“Module keyword” on page 161](#) to identify the maintenance level of the module that failed.

MSGADRnnnt

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 RA10 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 ADR*mmmmm* with the module identifier from the message (*mmmmm* in the following example).

```
ADR013I (ttt)-mmmm(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 DFSMSdss stand-alone restore program, the module keyword is always ADRDMPRS. If you select this keyword, see [“Maintenance-level keyword” on page 163](#).

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 163](#).

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 RA10 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, SYSDUMP, or SYSDUMP 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)-mmmm(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)-mmmm(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. See [“Module keyword” on page 161](#).
- d. Check prior messages for an ADR101I message with a task identifier that matches the one that you found in step [“3.a” on page 159](#). For example:

```
ADR101I (ttt)-mmmm(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 161](#).

Guideline: For the DFSMSdss stand-alone restore program, the module keyword is always ADRDMPRS. If you select this keyword, see [“Maintenance-level keyword” on page 163](#).

LOOP

Use this procedure when some part of the program repeats endlessly. If a message repeats endlessly, use the MSGADRNnt keyword.

Example: 5695DF175 RA10 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)-mmmm(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)-mmmm(yy), date_and_time TASK COMPLETED
```

- c. Use CNTRL as the function keyword if no tasks are active; see [“Module keyword” on page 161](#).
- 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 [“3.a” on page 159](#). For example:

```
ADR101I (ttt)-mmmm(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 161.

Guideline: For the DFSMSdss stand-alone restore program, the module keyword is always ADRDMPRS. If you select this keyword, see “Maintenance-level keyword” on page 163.

INCORROUT

Use this procedure if you expect output but do not receive it, or if the output is not what you expected.

Examples:

```
5695DF175 RA10 INCORROUT MSGADR nnnt ADR mmmm OC yy function
or
5695DF175 RA10 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” on page 158 through “3” on page 158 in “MSGADRnnnt” on page 158, 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
CGCREATED
COMPRESS
CONSOLIDATE
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; see “Using the IBM Support Center” on page 163.

DOC

Use this procedure when you encounter incorrect or missing information in a DFSMSdss publication. For a minor publication error, use the form for readers' comments at the back of the publication. If the error is serious and of general concern to other users, continue with the procedure described here.

Example: 5695DF175 RA10 DOC xxxnnnnnnnnnn

Procedure: To determine the keywords for type-of-failure and document number, follow these steps:

1. Use DOC as the type-of-failure keyword.

-
2. Specify 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, for document order number SC35-0423-09, specify the following:

```
DOC SC35042309
```

-
3. Locate the page of the error in the document and prepare a description of the problem. If you submit an APAR, include this information in the error description.
-

You have now obtained the keywords for type-of-failure and document number; see [“Using the IBM Support Center”](#) on page 163.

PERFM

Use this procedure if performance is less than expectations and the problem cannot be corrected by system tuning.

Example: 5695DF175 RA10 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  
CGCREATED  
COMPRESS  
CONSOLIDATE  
CONVERTV  
COPY  
COPYDUMP  
DEFRAG  
DUMP  
PRINT  
RELEASE  
RESTORE
```

You have now obtained the keywords for type-of-failure and function; see [“Using the IBM Support Center”](#) on page 163.

Module keyword

This topic applies only to the ABENDxxx, 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 ADDRIO1, as indicated in the PRB/CDE. Its normal state is waiting for the return from the WAIT SVC (X'01') in ADDRIO1.
- The subtasks for the DFSMSdss functions are ADRBLDSA, ADRCGCR, ADRCPYD, ADRPRNT, ADREFRAG, ADRDDTFP, ADRDTS, ADRDTS, ADRDDDS, ADRCMPRO, ADRLSE0, ADRTDFP, ADRTDDS, or ADKRVOL, 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: Perform the following.

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 determine whether one is not in its normal state, as described in this topic.

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/yyHDZ1A10 aparnum
.
.
ptfnum, yr.day, hh:mm:ss
```

where:

ADRxxxxx

Module name

mm/dd/yy

Maintenance date

HDZ1A10

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 this search. You can usually find these registers in the first supervisor request block (SVRB) after the PRB for the failing task.

Examples:

- a. 5695DF175 RA10 ABEND *xxx function module*
- b. 5695DF175 RA10 WAIT *function module*
- c. 5695DF175 RA10 LOOP *function module*

-
3. If you already know the PTF number, see [“Using the IBM Support Center” on page 163](#). Otherwise, see [“Maintenance-level keyword” on page 163](#).
-

Maintenance-level keyword

Use this keyword to identify the maintenance level of the module that failed.

Procedure: Perform the following.

1. Use the PTF number as the maintenance level keyword. The PTF number is found when you find the module identifier (see Step [“2” on page 162](#) in [“Module keyword” on page 161](#)).
-
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. Contact the IBM Support Center; see [“Using the IBM Support Center” on page 163](#).

Using the 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.

ASAP

Automatic Software Alert Process. This facility allows the user to be alerted when critical service information becomes available on a list of products selected by the user.

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.

ETR

Electronic Technical Response. Through this facility, the user may electronically report problems and ask appropriate technical questions about IBM products. Problem reports and questions are answered electronically. Optionally, problem reports may be answered through voice contact at the request of the user. Submit nondefect-related, nontechnical questions to the question and answer (Q&A); queue in Canada on a Severity 3, priority 3 basis.

FIXCAT

A fix category identifies groups of software fixes (PTFs). A fix category may identify software fixes that are required to support a particular hardware device, support compatibility with a new software release, or to provide a software function.

SRCHSERVICE

Online database of current authorized program analysis report (APAR) and program temporary fix (PTF) information with extensive search capability.

SRD

Service Request and Delivery facility. This facility provides a means for election ordering and delivery of corrective services including PTFs and APARs.

VPL

View Program Listings. Online database of module listings for non-OCO modules distributed via 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 database feature, Info/MVS. The database divides itself into several logical files of related or similar information, such as IBM ServiceLink.

Chapter 11. ACS routine information

This topic contains general-use Programming Interface and Associated Guidance Information.

This topic lists the variables you can use for automatic class selection (ACS) routines during DFSMSdss copy, restore, and CONVERTV operations, and describes the processing of ACS routines. This information is provided for guidance purposes only. It is not associated with any interface provided by DFSMSdss.

Messages generated by ACS routines are not printed by DFSMSdss unless the ACS routine returns a non-zero return code.

Related reading: For information about writing ACS routines, see [Chapter 23, “Application programming interface,”](#) on page 577.

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 15 on page 165](#) are passed to the ACS routines.

Table 15. 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' unless an application has specified an ACS environment in EI22ACSEN.
&ACSENVIR2	The sub-environment in which the ACS routine was invoked. The valid values are the values for &ACSENVIR as well as FLASHCPY and blank.
&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 the RACF component of z/OS Security Server is installed).
&DD	DDNAME
&DEF_DATACLAS	Default data class name (available only if the RACF component of z/OS Security Server is installed).
&DEF_MGMTCLAS	Default management class name (available only if the RACF component of z/OS Security Server is installed).
&DEF_STORCLAS	Default storage class name (available only if the RACF component of z/OS Security Server is installed).
&DSN	Data set name.
&DSNTYPE	Data set name type (for example: EXT, HFS, LIBRARY, PDS, or null).
&DSORG	Data set organization.
&DSOWNER	Owner or group considered to be the data set owner (available only if the RACF component of z/OS Security Server is installed).

Table 15. 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.
(continued)

Variable Name	Description
&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 167 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 167 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 product, such as z/OS Security Server, 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 16 on page 166](#) are passed to the ACS routines.

Table 16. 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 unless an application has specified an ACS environment in EI22ACSEN. Set to 'CONVERT' for CONVERTV.
&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.

Table 16. Variables Passed to ACS Routines during DFSMSdss Restore and CONVERTV Processing (continued)

Variable Name	Description
&APPLIC	Application identifier associated with the data set (available only if the RACF component of z/OS Security Server is installed).
&DEF_DATACLAS	Default data class name (available only if the RACF component of z/OS Security Server is installed).
&DEF_MGMTCLAS	Default management class name (available only if the RACF component of z/OS Security Server is installed).
&DEF_STORCLAS	Default storage class name (available only if the RACF component of z/OS Security Server is installed).
&DSN	Data set name.
&DSNTYPE	Data set name type (for example: EXT, HFS, LIBRARY, PDS, or null).
&DSORG	Data set organization.
&DSOWNER	Owner or group that is considered to be the data set owner (available only if the RACF component of z/OS Security Server 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 167 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 167 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 product, such as z/OS Security Server, 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.

Chapter 12. Dumping and restoring Linux for IBM Z partitions and volumes

You can include Linux volumes in an existing z/OS backup solution that uses DFSMSdss. If so, you can use DFSMSdss to dump Linux volumes to tape or to a *direct access storage device* (or *DASD*).

This backup solution is intended for the following environments:

- A z/OS-centric environment with z/OS running in several logical partitions (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 can perform the DFSMSdss processing.

In this section, the following topics describe how to use DFSMSdss to dump and restore Linux for IBM Z partitions and volumes, and also how to use a z/OS system to back-up Linux partitions that are attached to a Linux for IBM Z image:

- [“Preparing to work with Linux volumes” on page 169](#)
- [“Backing up a Linux volume with partitions” on page 171](#)
- [“Using DFSMSdss dump and restore commands” on page 172.](#)

This information is intended for storage administrators who are familiar with Linux for IBM Z. To do this work, you require root authority on Linux and must be authorized to run DFSMSdss batch jobs through RACF, or a functionally equivalent security product.

Note:

The examples in this topic were tested with:

- DFSMSdss Release 10
- OS/390 Version 2 Release 10
- z/OS Version 1 Release 1.

The procedures might not work with other versions of DFSMS.

Preparing to work with Linux volumes

This topic describes the following requirements for using DFSMSdss to dump and restore Linux for IBM Z partitions and volumes.

Understanding the hardware environment

An operating system can run on the processor in one of the following modes:

BASIC

A single operating system image that owns the entire processor or all processors.

LPAR

Depending on the model, a processor can be divided into as many as 60 logical partitions, with 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 IBM Z, all on the same hardware.

Choosing VOLSERs for Linux volumes

In a z/OS environment, DASD is divided into logical units known as *volumes*. You can define a volume to any size, up to a maximum size supported by z/OS environment.

Volumes are further divided into fixed-size *tracks*. The device's 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 (known as the *volser*).

When choosing a volser for a volume, follow these rules:

- The volser is six characters long. z/OS accepts fewer than six characters for the volser, but Linux requires that volumes have a six character volser.
- The volser uses uppercase, alphanumeric characters (A-Z, 0-9), and special characters (\$, #, @).

Formatting and partitioning Linux volumes

You must format and partition a volume before Linux can use it. Specifically, you must format the Linux volumes in the compatible disk layout (cdl) using `dasdfmt` version 1.0, and partition them using `fdasd` version 1.0.

This topic describes these steps in more detail:

- [“Using dasdfmt to format a Linux volume” on page 170](#)
- [“Using fdasd to partition a Linux volume” on page 170.](#)

Using dasdfmt to format a Linux volume

The default disk layout for `dasdfmt` is cdl. Volumes formatted in the original Linux disk layout (ldl) are not compatible with z/OS, and as such, cannot be backed up by z/OS.

The following example shows 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 result should appear similar to the following screen:

```
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.
```

In the example, a 3339 cylinder volume is attached to Linux at address 198. The volser is LNX200, and its block size is 4096 bytes. Linux requires the volser to be six characters in length. The disk label, VOL1, indicates that z/OS can process the volume.

A classic Linux volume has a disk label of LNX1; z/OS cannot process a volume with this disk label.

Using fdasd to partition a Linux volume

After you have formatted a volume with `dasdfmt` in the compatible disk layout, you must partition it before Linux can use it. Use the `fdasd` program to partition the volume.

The fdasd 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 extended count-key-data (ECKD) DASD instead of on hard drives. With fdasd, you can create up to three partitions on a volume and you can set the size of each partition. The partitions appear to z/OS as data sets.

When creating partitions with fdasd that you want to back-up with z/OS, observe the following rules:

- 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. Doing so renames the partitions and the data sets.

Obtaining authorization for Linux volumes

For Linux, you need root authority to mount and unmount the partitions, and 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 treats the Linux partitions as data sets. Also, you can prevent unauthorized access of the Linux partitions by z/OS applications and users by using RACF or an equivalent security product.

Backing up a Linux volume with partitions

z/OS treats a Linux partition (such as `/dev/dasd/0198/part1`) as a data set. The data set is named `LINUX.Vvolser.PART000x.NATIVE` for a data partition, or `LINUX.Vvolser.PART000x.SWAP` for a swap partition.

Here, *volser* is the volume serial number assigned to the volume when dasdfmt formatted the volume. fdasd can change the volser, too. The volser must be unique for z/OS 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.VLN200.PART0001.NATIVE`, where LNX200 is the volser of the volume.



Attention: If a partition is mounted read/write while undergoing a dump operation, data written to the partition during the dump operation might not be included in the backup. 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. Instead, 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 mounted read/write, DFSMSdss can back up your data, but the data might be inconsistent. By unmounting or remounting a partition read-only, you can help to ensure that all of your data is 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 use this naming convention:

Data Set Names	Partition Names
<code>LINUX.Vvolser.PART0000.type</code>	<code>/dev/dasd/yyyy/part1</code>
<code>LINUX.Vvolser.PART0001.type</code>	<code>/dev/dasd/yyyy/part2</code>
<code>LINUX.Vvolser.PART0002.type</code>	<code>/dev/dasd/yyyy/part3</code>

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.

Do not rename the data sets. fdasd expects the 24th character to be an 'N' or an 'S'. Otherwise, fdasd cannot recognize the partition type.

Using DFSMSdss dump and restore commands

This topic contains examples of batch jobs that use DFSMSdss functions to dump and restore Linux for IBM Z partitions and volumes. To submit these jobs for processing, you can either submit them to z/OS from your TSO/E user ID, or FTP them to a z/OS system from a Linux system, as described in [“Submitting JCL batch jobs to a z/OS system using FTP”](#) on page 177.

This topic contains the following examples for your reference:

- [“Example 1. DUMP FULL”](#) on page 172
- [“Example 2. DUMP FULL with CONCURRENT COPY”](#) on page 173
- [“Example 3. DUMP DATASET”](#) on page 173
- [“Example 4. COPY FULL”](#) on page 174
- [“Example 5. COPY FULL COPYVOLID ALLEXCP”](#) on page 174
- [“Example 6. RESTORE FULL”](#) on page 175
- [“Example 7. RESTORE DATASET”](#) on page 175
- [“Example 8. COPYDUMP”](#) on page 177.

Note:

1. IBM recommends using only the ADRDSSU keywords, which are shown between //SYSIN and /* in the examples in this section.
2. You must include ALLEXCP among the keywords you specify for any DFSMSdss DUMP batch job or COPY batch job that processes Linux cdl volumes.
3. For information about JCL rules and syntax, see the following publications:
 - [z/OS MVS JCL User's Guide](#)
 - [z/OS MVS JCL Reference](#).

Example 1. DUMP FULL

You can use DFSMSdss to dump the entire contents of a volume to tape or DASD, and have DFSMSdss restore the dump at a later time. Here, you can use the DFSMSdss DUMP FULL command. After you create a dump of the boot volume, you can restore the volume to multiple volumes as a way of making identical copies of the basic Linux system.

On the DUMP FULL command, include the keyword ALLEXCP to cause DFSMSdss to process all of the data set or partition, even if unused. ALLEXCP is required; your data is not backed up if you do not specify it.

[Figure 7 on page 172](#) shows an example of the JCL (the Linux volume has volser LNX200).

```
//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
/*
```

Figure 7. Sample JCL for dumping the contents of a volume.

You can also dump to two or more output tapes at the same time, for example, if you wanted a backup and a copy of the backup for storage off-site.

[Figure 8 on page 173](#) shows an example of the JCL.


```
//LXD2D1XX JOB , 'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT02 EXEC PGM=ADDRSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNK200,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
/*
```

Figure 8. Sample JCL for dumping two or more output tapes at the same time.

Example 2. DUMP FULL with CONCURRENT COPY

You can use the volume being dumped without the dump being affected, much sooner than with the normal full volume dump. You might want to use this method when the Linux partitions that are being backed up need to be available in read/write mode.

On the DUMP FULL command, include the keyword CONCURRENT.

Figure 9 on page 173 shows an example of the JCL.

```
//LXD2D2BB JOB , 'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT02 EXEC PGM=ADDRSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNK200,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
/*
```

Figure 9. Sample JCL for DUMP FULL with CONCURRENT COPY.

The dump will reflect whatever data was present when the concurrent copy job started.

When you see message ADR734I as shown in the following example, 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 LNK200. SERIALIZATION FOR THIS DATA IS RELEASED IF DFSMSDSS
HELD IT. THE INTERMEDIATE RETURN CODE IS 0000.
```

Example 3. DUMP DATASET

You can dump individual partitions by using physical processing. Doing might 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).

Figure 10 on page 174 shows an example of the JCL.

```
//LXD2J1BB JOB , 'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT03 EXEC PGM=ADDRSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNK200,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
/*
```

Figure 10. Sample JCL for DUMP DATASET.

You can also dump all of the Linux partitions. [Figure 11 on page 174](#) shows an example of the JCL.

```
//LXD2J2BB JOB , 'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=0M,
//      MSGCLASS=H,CLASS=A
//STEPT03 EXEC PGM=ADDRSSU
//SYSPRINT DD SYSOUT=*
//DASDIN   DD UNIT=3390,VOL=SER=LNK200,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
/*
```

Figure 11. Sample JCL for dumping all of the Linux partitions.

Example 4. COPY FULL

You can make a full volume copy of a volume. Doing so would allow you, for example, to populate a new server with a standard configuration.

The COPY FULL function can also be useful because of FlashCopy for Enterprise Storage Server (ESS) devices, or SnapShot for RAMAC Virtual Array (RVA) devices. DFSMSdss attempts to use the fastest copy method possible before using the traditional data movement methods. FlashCopy or SnapShot make a virtually instantaneous copy of the volume, allowing you to continue using the volume.

Using FlashCopy requires the volumes to be in the same ESS that supports FlashCopy Version 2 (data set FlashCopy), or in the same logical subsystem in an ESS that supports FlashCopy Version 1, but not support FlashCopy Version 2 (data set FlashCopy) or later functions.

Similarly, when using SnapShot, the volume on which create a copy must be within the same subsystem as the source volume. In an RVA device, the four SSIDs that are defined to the RVA are considered to be within the same subsystem.

After the COPY FULL command completes, the data on LNX900 is the same as LNX200, except that the volser is LNX900. [Figure 12 on page 174](#) shows an example of the JCL.

```
//LXD2C1BB JOB , 'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//      MSGCLASS=H,CLASS=A
//STEPT02 EXEC PGM=ADDRSSU
//SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,VOL=SER=LNK200,DISP=OLD
//TARGET   DD UNIT=3390,VOL=SER=LNK900,DISP=OLD
//SYSIN    DD *
COPY FULL INDDNAME(SOURCE) OUTDDNAME(TARGET) -
      ALLEXCP
/*
```

Figure 12. Sample JCL for making a full volume copy of a volume.

Example 5. COPY FULL COPYVOLID ALLEXCP

You can create a backup copy of a Linux volume.

On the DFSMSdss COPY FULL command, include the COPYVOLID keyword to cause DFSMSdss to copy the volser to the new volume. When the copy operation completes, the two volumes are identical, including the volser. Because z/OS allows only one volume with a particular volser to be online at a time, DFSMSdss varies offline the volume that was the target of the copy.

On the COPY FULL command, include the keyword ALLEXCP to cause DFSMSdss to process all of the data set or partition, even if unused. ALLEXCP is required; your data is not backed up if you do not specify ALLEXCP.

Figure 13 on page 175 shows an example of the JCL.

```
//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=LNK200,DISP=OLD
//TARGET   DD UNIT=3390,VOL=SER=LNK900,DISP=OLD
//SYSIN    DD *
COPY FULL INDDNAME(SOURCE) OUTDDNAME(TARGET) -
          COPYVOLID ALLEXCP
/*
```

Figure 13. Sample JCL for creating a backup copy of a Linux volume.

Example 6. RESTORE FULL

You can restore a full volume from a dump taken by DFSMSdss. You would use this command when you want a Linux volume from a DFSMSdss dump that was created earlier.

Specify the RESTORE command with the FULL keyword. This action restores the entire contents from the original volume.

Figure 14 on page 175 shows an example of the JCL.

```
//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=LNK200,DISP=OLD
//SYSIN    DD *
RESTORE FULL INDDNAME(SOURCE) OUTDDNAME(TARGET) PURGE
/*
```

Figure 14. Sample JCL for restoring a full volume from a DFSMSdss dump.

In Figure 14 on page 175, observe the following:

- The Linux volume is *volser* LNK200
- The dump that was stored in data set TDS.DUMP200 is being restored to LNK200
- The SOURCE DD statement identifies where DFSMSdss is to find the information to restore
- The TARGET DD statement identifies where DFSMSdss is to restore the volume information and data.

You can overwrite any data sets that are on the LNK200 volume currently with unexpired dates when you specify PURGE. All Linux partitions are permanent, and thus have "never expire" dates.

Example 7. RESTORE DATASET

You can restore individual partitions or 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.

Figure 15 on page 176 shows an example of the JCL.

```
//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=LNK200,DISP=OLD
//SYSIN    DD *
RESTORE INDDNAME(SOURCE) OUTDDNAME(TARGET) -
        DATASET(INCLUDE(LINUX.**.NATIVE)) REPLACE
/*
```

Figure 15. Sample JCL for restoring individual partitions or data sets.

In [Figure 15 on page 176](#), notice that the keyword REPLACE is specified instead of PURGE. For data set level restores, REPLACE causes DFSMSdss to replace any existing data sets 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 obtain the backup version. Also in [Figure 15 on page 176](#), the INCLUDE statement indicates that DFSMSdss is 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, do 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.

To restore a deleted partition or a partition that never existed, use fdasd, which 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 or partition, the data is placed correctly and you do not lose any partitions. The reason is that the name of the second partition is the same as the restored first partition.

For example, if you delete /dev/dasd/xxxx/part1 (known to z/OS as LINUX.VLNK200.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, too. After fdasd deletes LINUX.VLNK200.PART0000.NATIVE, it renames LINUX.VLNK200.PART0001.NATIVE to LINUX.VLNK200.PART0000.NATIVE. If you then use DFSMSdss to restore the first partition (named LINUX.VLNK200.PART0000.NATIVE), you will lose the second partition.

You can also use the RENAMEUNCONDITIONAL keyword to change the names of the data sets that you are restoring. [Figure 16 on page 176](#) shows an example of the JCL.

```
//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=LNK300,DISP=OLD
//SYSIN    DD *
RESTORE INDDNAME(SOURCE) OUTDDNAME(TARGET) -
        DATASET(INCLUDE(LINUX.VLNK200.PART0001.NATIVE))
        RENAMEUNCONDITIONAL(LINUX.VLNK200.PART0001.NATIVE, -
        LINUX.VLNK300.PART0001))
/*
```

Figure 16. Sample JCL for renaming data sets to be restored.

If you want to change the data set, change only the volser or the last character of the partition name (PART000x). If you change anything else, Linux might not recognize the partition.

Assume that you have three partitions on a volume in which:

- Partition 1 contains programs
- Partition 2 contains data
- Partition 3 is a swap partition.

Now suppose that someone with root authority accidentally deletes the programs on Partition 1. As a result, you need to restore the backup versions of those programs, but leave the data partition (Partition 2) alone. You might not have to restore Partition 3 because it is swap space.

Figure 17 on page 177 shows an example of the JCL you might use to restore 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=ADDRSSU //SYSPRINT DD SYSOUT=*
//SOURCE   DD UNIT=3390,DISP=OLD,DSN=TDS.DUMP200
//TARGET   DD UNIT=3390,VOL=SER=LNK200,DISP=OLD
//SYSIN    DD *
RESTORE INDDNAME(SOURCE) OUTDDNAME(TARGET) -
        DATASET(INCLUDE(LINUX.VLNK200.PART0000.NATIVE)) REPLACE
/*
```

Figure 17. Sample JCL for restoring only one partition of a volume.

Example 8. COPYDUMP

With DFSMSdss, you can copy Linux volume dumps. You might want to do this if you need to have copies of your dump tapes for disaster recovery purposes. Some installations, for example, keep an on-site backup tape in addition to an off-site copy of the dump.

To copy a Linux volume dump, use the DFSMSdss COPYDUMP command. You can copy a dump immediately after creating it, or you can make a copy of an older dump.

Figure 18 on page 177 shows an example of the JCL.

```
//LXDRP1AA JOB , 'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
//          MSGCLASS=A,CLASS=A
//STEPT03 EXEC PGM=ADDRSSU
//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)
/*
```

Figure 18. Sample JCL for copying Linux volume dumps.

Submitting JCL batch jobs to a z/OS system using FTP

You can use FTP to submit your JCL batch jobs to a z/OS system from a Linux image. When you FTP to the z/OS system, enter the `site file=jes` command. The FTP server on the z/OS system should route any file that it receives to the job entry subsystem (JES) for execution. (Your login id must have sufficient authority to run DFSMSdss batch jobs, as mentioned in [“Obtaining authorization for Linux volumes”](#) on page 171).

Then, “put” your JCL batch jobs to the z/OS system. Save your jobs as text files that do not exceed 80 characters in length.

Using DFSMSdss stand-alone services

You can use DFSMSdss stand-alone services to create an IPL-able (that is, bootable) image on tape. With the IPL-able image, you can use the DFSMSdss stand-alone restore program to restore a Linux volume that was backed up by DFSMSdss. You can do this without having to start z/OS.

For a list of devices you can use with the DFSMSdss stand-alone restore program, see the topic on DFSMSdss stand-alone services in [“DFSMSdss stand-alone services”](#) on page 515.

Chapter 13. Format of the DFSMSdss dump data set

This topic describes the formats of the DFSMSdss dump data set, and data areas ADRBMB ([Table 17 on page 180](#)) and ADRTAPB ([Table 19 on page 181](#)).

For information about ADRUFO, refer to [z/OS DFSMS Installation Exits](#).

For information about ADREID0, refer to [“ADREID0 data area” on page 609](#).

Data set and volume backup format

- For a physical dump, the volume record may also contain an encryption record following the volume header when encryption was performed during the dump. The volume record may also contain an extended volume record if any data was dumped from an EAV, if zCompression was used, or the block size of the output dump data set on tape is greater than 65,520 bytes. The extended volume record follows the:
 - Encryption record, if encryption was performed
 - Volume record, if encryption was not performed.
- 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. The tape header record may also contain an extended volume record if zCompression was used, or if the block size of the output dump data set on tape is greater than 65,520 bytes. The extended tape header record follows the:
 - Encryption record, if encryption was performed
 - Tape header record, if encryption was not performed.
- 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 180](#).
3. Track 0: dump of track 0 of cylinder 0.
4. VSAM volume data set (VVDS) track records, if VSAM or SMS-managed 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.

Note:

1. For each data set, repeat items “3” on page 179 through “7” on page 180.
2. For each volume of the data set, repeat items “4” on page 179 and “6” on page 179.

Every record on a DFSMSdss dump data set is described by the “ADRTAPB data area” on page 181.

ADRBMB data area

Table 17. ADRBMB Mapping Macro					
Offsets		Type	Len	Name (Dim)	Description
Dec	Hex				
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
Note:					
0	(0)	STRUCTURE	*	ADRCMB	CHUNK MAP BLOCK
0	(0)	CHARACTER	12	CMHDR	CHUNK MAP HEADER
0	(0)	CHARACTER	4	CMID	BLOCK EYECATCHER
4	(4)	ADDRESS	4	CMNPTR	PTR TO NEXT CHUNKMAP
8	(8)	UNSIGNED	4	CMNCHNK	NUMBER OF CHUNKS MAPPED
8	(8)	UNSIGNED	2	CMNCHKTP	SHIFTED NUMBER OF CHUNKS
10	(A)	CHARACTER	2	*	RESERVED BY THIS SEGMENT
12	(C)	BITSTRING	0	CMBITMAP	BITMAP LAYOUT, CONTIG BITS FOR EACH 21 CYL CHUNK TO BE PROCESSED
Note:					
0	(0)	STRUCTURE	*	ADRTSMB	TRACK SUBMAP
0	(0)	CHARACTER	8	TSMHDR	TRACK SUBMAP HEADER
0	(0)	CHARACTER	4	TSMID	TRACK SUBMAP EYECATCHER
4	(4)	SIGNED	4	TSMCHKCC	CYL ADDR OF CHUNK START
8	(8)	BIT(*)	*	TSMBITMP	TRACK SUB BITMAP

ADRBMB constants

Table 18. ADRBMB Mapping Macro				
Length	Type	Value	Name	Description
4	CHARACTER	CMBB	ADRCMID	BLOCK IDENTIFIER
4	DECIMAL	197	CMAKSEG#	MAX NUM OF BITMAP
4	DECIMAL	315	CHKTRKS	15*21 TRACKS / CHUNK

Table 18. ADRBMB Mapping Macro (continued)

Length	Type	Value	Name	Description
4	DECIMAL	21	CHUNKSZ	21 CYLS PER CHUNK
4	CHARACTER	TSMB	ADRTSMID	BLOCK IDENTIFIER
4	DECIMAL	40	CSTRKSZ	SIZE OF TRACK SUBMAP

ADRBMB cross-reference

Name	Hex Offset
ADRBMB	0
ADRCMB	0
ADRTSMB	0
BMBITMAP	12
BMHDR	0
BMID	0
BMNPTR	4
BMNTRK	8
CMBITMAP	12
CMHDR	0
CMID	0
CMNCHKTP	8
CMNCHNK	8
CMNPTR	4
TSMBITMP	8
TSMCHKCC	4
TSMHDR	0
TSMID	0

ADRTAPB data area

Table 19. ADRTAPB Mapping Macro

OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
Mapping of main prefix area (first thing in every record).					
0	(0)	STRUCTURE	16	ADRTAPB	
0	(0)	CHARACTER	6	DTPSCHK	
0	(0)	SIGNED	4	DTPSEQNO	SEGMENT SEQUENCE NUMBER
4	(4)	UNSIGNED	1	DTPNOSEG	NUM OF SEGMENTS PER RECORD
5	(5)	UNSIGNED	1	DTPSEGNO	SEGMENT NUM OF RECORD
6	(6)	UNSIGNED	2	DTPSEGLN	SEGMENT LENGTH INCL PREFIX
8	(8)	UNSIGNED	1	DTPPFXLN	LENGTH OF PREFIX(CONSTANT 16)
9	(9)	CHARACTER	1	DTPDMPID	TYPE OF DUMP ID

Table 19. ADRTAPB Mapping Macro (continued)

OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
		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 COMPATIBILITY BETWEEN V2.1, V2.2 AND V2.3
	 1111		*	RESERVED - DO NOT USE LAST 4 BITS IN DTPDMPID TO ENSURE DOWNWARD COMPATIBLTY
10	(A)	BIT(8)	1	DTPRCID1	RECORD IDENTIFIER 1
		1... ..		DTPVHDR	80 - VOLUME HEADER (SEE DTVOL)
		1... ..		DTPTHDR	- TAPE HEADER (see DTHDR)
		.1... ..		DTPBTM	40 - MAP OF DUMPED TRKS (see DTBTMR)
		.1... ..		DTPDSNL	- DS NAME/CATALOG LST (see DTLDN)
		..1.		DTPTRK0	20 - TRACK 0 (see DTTTRK)
		..1.		DTPDSHDR	- DATA SET HEADER (see DTDSDHDR)
		...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.		DTPVDS	02 - VVDS TRACK (see DTTTRK)
	1		DTPSPHDR	01 - SPHERE REC HEADER (see DTSPPHRE)
11	(B)	BIT(8)	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 NON-TRACK DATA SET (EG, 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 DUMPCOND VOL
	1.		DTPSELF	SELF DESC RECORDS EXIST
	1		*	RESERVED
12	(C)	CHARACTER	2	DTPVER#	ADRTAPB VERSION NUMBER
14	(E)	CHARACTER	2	*	RESERVED
16	(10)	CHARACTER	0	DTPBODY	START OF REMAINDER OF RECORD
Rest of volume header record (follows ADRTAPB).					
0	(0)	STRUCTURE	46	DTVOL	
0	(0)	CHARACTER	4	DTVTOCB	VTOC BEGINNING CCHH

Table 19. ADRTAPB Mapping Macro (continued)

OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
4	(4)	CHARACTER	4	DTVTOCE	VTOC ENDING CCHH
8	(8)	CHARACTER	4	DTVOLSZ	VOLUME SIZE(DS4DEVSZ)
8	(8)	UNSIGNED	2	DTVLOGCY	# OF CYLINDERS IN VOLUME
10	(A)	UNSIGNED	2	DTVTRKCY	# 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 & 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	DEVTYPE OF VOLUME(UCBTBYT4)
35	(23)	UNSIGNED	1	DTVMODNO	MODEL NUMBER
36	(24)	BIT(8)	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 fmt
	 1...		DTVFCMP	08 - FILE COMPRESSED
	1..		DTVUNLCD	04 - UNALLOCATED SPACE DUMPED
	1.		DTVLVF	02 - OTHER LOGICAL VOLUMES MAY FOLLOW THIS LOGICAL VOLUME
	1		DTVVDS	01 - VVDS DATASET DUMPED BEFORE VTOC
37	(25)	UNSIGNED	2	DTVBMSZ	BITMAP SIZE IN WORDS
39	(27)	BIT(8)	1	DTVIND2	VOLUME TYPE INDICATORS
		1...		DTVLNVI	80-NONVSAM DATA SETS NOT ON VOLUME
		.1...		DTVLVI	40-VSAM DATA SETS NOT ON VOLUME
		..1.		DTVGNI	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.		DTVFHCOMP	HARDWARE COMPRESS USED
	1		*	UNUSED
40	(28)	UNSIGNED	2	DTVLEN	LEN OF HEADER (SEGMENT LEN - PREFIX
42	(2A)	UNSIGNED	1	DTVVERNO	DFDSS VERSION NUMBER
43	(2B)	UNSIGNED	1	DTVLVLNO	DFDSS MODIFIC. NUMBER
44	(2C)	UNSIGNED	1	DTVVXTNO	NUMBER OF VVDS EXTENTS
45	(2D)	UNSIGNED	1	DTVVXTOF	OFFSET TO VVDS EXTENT
Mapping of additional volume information (follows DTVOL).					

Table 19. ADRTAPB Mapping Macro (continued)

OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
0	(0)	STRUCTURE *		DTVEND	FIELDS FROM DTVOL START
0	(0)	CHARACTER	0	*	
Mapping of VVDS extents (in DTVEND).					
0	(0)	STRUCTURE	10	DTVXTNT(*)	VVDS EXTENTS, IN DSCB FORMAT
0	(0)	CHARACTER	10	DTVXLLEN	
Rest of log data set dump tape header (follows ADRTAPB).					
0	(0)	STRUCTURE	18	DTHDR	
0	(0)	CHARACTER	8	DTHTIMD	DATE & 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		DTHLGDS#	EXTEND NUM-DS AREA USED
	 1111		*	RESERVED
9	(9)	UNSIGNED	2	DTHLEN	HEADER LEN
11	(B)	UNSIGNED	1	DTHVERNO	DFDSS VERSION NUMBER
12	(C)	UNSIGNED	1	DTHLVLNO	DFDSS MODIFIC. NUMBER
13	(D)	UNSIGNED	2	DTHBLKSZ	MAX BLKSIZE
15	(F)	UNSIGNED	2	DTHNDS	# DS 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
	 1...		DTHEFSAM	EF SAM DS HAVE DTTTRK
	111		*	RESERVED
Rest of data set name/catalog list (follows ADRTAPB).					
0	(0)	STRUCTURE	45	DTLDSN	
0	(0)	UNSIGNED	1	DTLLEN	LENGTH OF DATA SET NAME
1	(1)	CHARACTER	44	DTLCAT	CATALOG NAME
Rest of data set header record (follows ADRTAPB).					
0	(0)	STRUCTURE	106	DTDHDR	
0	(0)	UNSIGNED	1	DTDLEN	LENGTH OF DATA SET NAME
1	(1)	UNSIGNED	1	DTDCA TLN	LENGTH OF CATALOG NAME
2	(2)	CHARACTER	2	DTDDSORG	DATA SET ORGA (FROM F1)
4	(4)	CHARACTER	1	DTDOPTCD	DS OPTION CODE (FROM F1)

Table 19. ADRTAPB Mapping Macro (continued)

OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
5	(5)	CHARACTER	1	DTDNVOL	NUMBER VOLS FOR DATA SET
6	(6)	BIT(8)	1	DTDIND	DATA SET INDICATOR
		1... ..		DTDIPWD	PASSWORD SUPPLIED 1=YES
		.1... ..		DTDTPWD	TYPE PASSWORD 0=D.S. 1=CATALOG
		..1.		DTDIRACF	RACF PROFILE
		...1 1...		DTDRAFCF	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 VOL CNT (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® & 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 ADTL INFO
	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
Mapping of sphere information (in DTSAIXS).					
Sphere record for catalog filter dump (follows ADRTAPB).					
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)	BIT(8)	1	DTSPATHF	PATH INFO FLAG
		1... ..		DTSPATHI	IF SET, PATH OWNERID IS CONTAINED IN THIS BLOCK (SEE DTSPHON FOR DESCRIPTION)

Table 19. ADRTAPB Mapping Macro (continued)

OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
		.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 DECIM
99	(63)	UNSIGNED	2	DTSERDAY	12-BITS DAY OF IN DECIMAL & 4-BIT SIGN CHAR THAT ALLOWS TO BE UNPACKED
101	(65)	UNSIGNED	1	DTSEXCNY	CENTURY BYTE IN DECIMAL
Mapping of security info table (in DTSPATHD) .					
0	(0)	STRUCTURE	52	DTSPASSW	SECURITY INFO TABLE
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	ADDR OF AUTHORIZATION MOD
50	(32)	SIGNED	2	D TSAUTHR	AUTHORIZATION REC LENGTH
Common data format data set attributes (follows DTDHDR).					
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	0	*	LAST
Mapping for additional data (follows DTCDFATT).					
0	(0)	STRUCTURE	*	DTDSAIR	
0	(0)	SIGNED	4	DTDSAIDL	ADD. DATA LENGTH
4	(4)	CHARACTER	*	DTDSAID	ADD. DATA FOLLOWS

Table 19. ADRTAPB Mapping Macro (continued)					
OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
Rest of volume definition record (follows ADRTAPB).					
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	#BYTES/TRK (TRACK CAPACITY WITH OVERHEAD).
16	(10)	UNSIGNED	2	DTMLOGCY	# CYLINDERS PER VOLUME
18	(12)	UNSIGNED	2	DTMTRKCY	# TRACKS PER VOLUME
20	(14)	UNSIGNED	2	DTMMAXCB	MAX COMPRESS BUF 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...		DTMWOT	RLS time stamps are BWO
	1..		DTM_TCT_COMPR	1=TCT compression used
22	(16)	BIT(11) POS(6)	2	*	Unused
24	(18)	UNSIGNED	1	DTM#VVRS	# OF VVRS/NVRS DUMPED
25	(19)	UNSIGNED	1	DTM#DSCB	# OF DSCBS DUMPED
26	(1A)	UNSIGNED	1	DTM#EXT	# OF EXTENTS DUMPED
27	(1B)	CHARACTER	1	DTMMODNO	MODEL NUMBER
Mapping of the DSCBs, extents and VVRs (follows DTMVOL).					
0	(0)	STRUCTURE	*	DTMVDATA	DSCBS (1&2) FOR DS, FOLLOWED BY EXTENT LIST, FOLLOWED BY VVRS OR NVR DATA SET CLUSTER.
0	(0)	CHARACTER	0	*	
Rest of record that maps the tracks dumped (follows ADRTAPB).					
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#	# OF TRKS MAPPED IN BMBB SEGMENT
8	(8)	UNSIGNED	2	DTBTRK#P	# OF TRKS MAPPED IN BMBB SEGMENT
10	(A)	CHARACTER	2	*	RESERVED
Mapping of the bitmap data (follows DTBTMR).					
0	(0)	STRUCTURE	*	DTBBITM	BITMAP
0	(0)	CHARACTER	0	*	
Rest of track record (follows ADRTAPB). The first segment of the track is preceded by DTTTRK. The second and subsequent track segments are not preceded by DTTTR, but are preceded by ADRTAPB. The remaining track image data begins immediately after ADRTAPB .					

Table 19. ADRTAPB Mapping Macro (continued)

OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
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 TRK
2	(2)	CHARACTER	1	DTTTRKID	TRACK INDICATORS
		1... ..		DTTIOER	I/O ERROR ON TRACK
		.1... ..		DTTTRVVF	LAST REC ON TRK IS OVFLD REC (maintained for restore compatibility only, do not set)
		..1.		DTTTCMP	IF ON, TRACK COMPRESSED
		...1		DTTVFRST	FIRST VVDS RECORD
	 1...		DTTINVT	INVALID TRACK FORMAT
	1..		DTTSTAT	USER STATISTICAL RECORD
3	(3)	UNSIGNED	4	DTTCCHH	CCHH OF TRACK 2@SVD
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)	UNSIGNED	1	DTTXDLEN	EXTRA ENC DATA LEN
12	(C)	CHARACTER	4	*	RESERVED
16	(10)	CHARACTER	8	DTTRODAT	RCRD 0 DATA, ONLY ON 1ST SEG
24	(18)	CHARACTER	0	DTTBODY	R1- RN RECORDS ON TRACK
Count field format (in track image data).					
0	(0)	STRUCTURE	8	DTTCKD	CNT, KEY & DATA FIELDS ON TRK
0	(0)	CHARACTER	8	DTTCNT	COUNT FIELD
0	(0)	CHARACTER	5	DTTCCCHR	CCHHR OF RECORD
0	(0)	UNSIGNED	4	DTTCCCHH	CCHH OF RECORD 2@SVD
4	(4)	UNSIGNED	1	DTTCRCRD	R OF RECORD
5	(5)	UNSIGNED	1	DTTKELEN	KEY LENGTH
6	(6)	UNSIGNED	2	DTTDATLN	DATA LENGTH
Key or Data field mapping (follows count field).					
0	(0)	STRUCTURE	*	DTTKD	KEY AND DATA FIELDS
0	(0)	CHARACTER	0	*	
Rest of record that maps the trailer record (follows ADRTAPB).					
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
0	(0)	STRUCTURE	4	DTSDHDR	
0	(0)	UNSIGNED	2	DTSDLEN	LEN OF SELF DESC REC
2	(2)	CHARACTER	1	DTSDTYPE	TYPE OF SELF DSC REC
3	(3)	CHARACTER	1	DTSDIND1	SELF DESC REC FLAGS

Table 19. ADRTAPB Mapping Macro (continued)

OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
		1... ..		DTSDLAST	LAST SELF DESC REC
		.111 1111		*	RESERVED
4	(4)	CHARACTER	0	DTSDEND	START SELF DESC REC ON A WORD BOUNDARY
0	(0)	STRUCTURE	356	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
4	(4)	CHARACTER	16	DTCDATAS	SAMPLE DATA
20	(14)	CHARACTER	64	DTCRSAL	RSA LABEL
84	(54)	CHARACTER	256	DTCRDKL	RSA CIPHD DATA LABEL
340	(154)	SIGNED	4	DTCICNT	ITERATION COUNT
344	(158)	CHARACTER	8	DTCSALT	SALT FOR ADRPWKEY
352	(160)	SIGNED	4	DTCRDKLN	RSA CIPHD DATA LAB LN@DDE
Compression dictionary block. Immediately follows a self-describing record header					
0	(0)	STRUCTURE	16	DTCDCT	FOLLOWS A DTS DHDR
0	(0)	CHARACTER	16	DTCFLDS	
0	(0)	SIGNED	4	DTCMAXD	MAX DICT SIZE
4	(4)	CHARACTER	12	*	RESERVED
0	(0)	STRUCTURE	*	DTCDICT	EXPANSION DICTIONARY AFTER DTCDCT
0	(0)	CHARACTER	0	*	AREA
Extended volume record found in first tape record					
0	(0)	STRUCTURE	52	DTSDEVOL	EXTENDED VOL SD REC
0	(0)	SIGNED	4	DTELOGCYL	# OF CYLINDERS - EAV
4	(4)	UNSIGNED	2	DTELCYL	LARGE UNIT REGION
6	(6)	CHARACTER	2	*	RESERVED
8	(8)	UNSIGNED	4	DTCBITM#	SIZE OF CHUNK MAP
12	(C)	CHARACTER	4	*	RESERVED
16	(10)	UNSIGNED	8	DTEVBLOCKSIZE	OUTPUT BLOCKSIZE
16	(10)	UNSIGNED	4	DTEVHIBLSZ	HIGH WORD BLKSIZE
20	(14)	UNSIGNED	4	DTEVBLKSZ	LOW WORD BLKSIZE WHEN DTVBLKSZ=X'FFFE'
24	(18)	BIT(8)	1	DTEFLGS	EXT VOLUME FLAG BYTE
		.1... ..		DTERESET	RESET SPECIFIED ON DUMP
		..1.		DTECYLMG	CYLINDER MANAGED SPACE

Table 19. ADRTAPB Mapping Macro (continued)

OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
		...1 ...		DTEVZCOMP	ZCOMPRESS SPECIFIED
		...1		DTEAPPMETA	API - Meta Object
		... 1...		*	RESERVED
	1..		DTE_TCT_COMPR	1=TCT compression used
	11		*	RESERVED
		...1 1111		*	RESERVED
25	(19)	CHARACTER	27	*	RESERVED
Extended tape record found in first tape record					
0	(0)	STRUCTURE	52	DTSDEHDR	EXT TAPE SD RECORD
0	(0)	UNSIGNED	8	DTEHBLOCKSIZE	OUTPUT BLOCKSIZE
0	(0)	UNSIGNED	4	DTEHHIBLSZ	HIGH WORD BLKSIZE
4	(4)	UNSIGNED	4	DTEHBLKSZ	LOW WORD BLKSIZE WHEN DTHBLKSZ=X'FFFE'
8	(8)	BIT(8)	1	DTETFLGS	EXT TAPE FLAG BYTE
		1...		DTETZCOMP	ZCOMPRESS SPECIFIED
		.111 1111		*	RESERVED
9	(9)	CHARACTER	3	*	RESERVED
12	(C)	BIT(64)	8	DTEDS#	EXTENDNUM DSETS AREA. USED IF DTHLGDS# IS ON
12	(C)	UNSIGNED	4	DTEDS#HI	HIGH BOUNDARY COUNT
20	(14)	UNSIGNED	4	DTEDS#LO	LOW BOUNDARY COUNT
76	(4C)	CHARACTER	32	*	RESERVED
0	(0)	STRUCTURE	64	BFRPREFIX	BUFFER PREFIX AREA@V22H
0	(0)	UNSIGNED	2	BPSCOC	OFFSET IN BUFFER FROM BEGINNING OF R0 DATA TO COUNT FIELD IN ERROR
2	(2)	UNSIGNED	2	BPSCOD	OFFSET IN BUFFER FROM BEGINNING OF R0 DATA TO DATA FIELD IN ERROR
4	(4)	SIGNED	4	BPHWMRK	ADDRESS OF NEXT AVAILABLE BYTE (HIGH WATER MARK OF USED STORAGE)
8	(8)	CHARACTER	56	BPWORKA	WORK AREA FOR OPTIMIZE@V22H
64	(40)		0	BPBODY	LOCATION OF REST OF BUFFER
0	(0)	STRUCTURE	*	DSAUTHR	AUTHORIZATION RECORD.
0	(0)	CHARACTER	0	*	
0	(0)	STRUCTURE	68	DTCHKSUM	CHECKSUM RECORD
0	(0)	CHARACTER	1	DTCKTYPE	TYPE OF CHECKSUM
		1...		DTCK_MD5	MD5 HASH
		.111 1111		*	UNUSED
1	(1)	CHARACTER	3	*	UNUSED
4	(4)	CHARACTER	64	DTCKHASH	HASH VALUE
4	(4)	CHARACTER	16	DTCK_MD5HASH	MD5 HASH VALUE

Table 19. ADRTAPB Mapping Macro (continued)					
OFFSET		Type	Len	Name (Dim)	Description
Dec	Hex				
0	(0)	STRUCTURE	24	DTDSHDRE	DATA SET HEADER EXTENSION
0	(0)	CHARACTER	2	DTDS_ACTUAL_CNT	ACTUAL VOLUME COUNT
0	(0)	UNSIGNED	1	DTDS_VCDT	DATA COMPONENT VOLUME COUNT
1	(1)	UNSIGNED	1	DTDS_VCTI	INDEX COMPONENT VOLUME COUNT
2	(2)	CHARACTER	2	DTDS_FLAGS	DATA SET HEADER EXTENSION FLAGS
		1		DTDS_APPMETA	1=APPMETA OBJECT STORED
4	(4)	CHARACTER	20	*	RESERVED
0	(0)	STRUCTURE	*	DTLDSCMP	COMPRESSED DATA SET NAME
0	(0)	CHARACTER	0	*	
Mapping of sphere information (in DTSAIXS).					
0	(0)	STRUCTURE	*	DTSAIXS	AIX SPHERE INFO
0	(0)	CHARACTER	0	*	
0	(0)	STRUCTURE	*	DTSPATHD	START PATH SECURITY INFO
0	(0)	CHARACTER	0	*	
Self-describing Record constants cipher block record					
"1"	"CHAR HEX"	"01"	DTSDCIPH	CIPHER BLOCK SD REC	
Compression dictionary record					
"1"	"CHAR HEX"	"02"	DTSDCDCT	COMPRESSION DICT SELF-DESC REC TYPE	
Extended addressable volume record					
"1"	"CHAR HEX"	"03"	DTSDVOLE	EXTENDED VOLUME SELF-DESC REC TYPE	
Extended data set record					
"1"	"CHAR HEX"	"04"	DTSDHDRE	TAPE HDR EXTENSION SELF-DESC REC TYPE	
Checksum record					
"1"	"CHAR HEX"	"05"	DTSDCHKSUM	CHECKSUM SELF-DESC REC TYPE	
Data set header extension record					
"1"	"CHAR HEX"	"06"	DTSDDSHDR	DATA SET HEADER EXTENSION	
"2"	"DECIMAL"	"1"	DTPVERC	Version	

ADRTAPB constants

Table 20. 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
1	CONSTANT	X'03'	DTSDVOLE	EXTENDED VOLUME SELF-DESC REC TYPE

Table 20. ADRTAPB Mapping Macro (continued)

Length	Type	Value	Name	Description
1	CONSTANT	X'04'	DTSDHDRE	TAPE HEADER EXTENSION SELF-DESC REC TYPE
1	CONSTANT	X'05'	DTSDCHKSUM	CHECKSUM SELF-DESC REC TYPE
1	CONSTANT	X'06'	DTSDDSHDR	DATA SET HEADER EXTENSION

ADRTAPB cross-reference

Name	Hex Offset	Hex Value
ADRTAPB	0	
BFRPREFIX	0	
BPBODY	40	
BPHWMRK	4	
BPSCOC	0	
BPSCOD	2	
BPWORKA	8	
DSAUTHR	0	
DTBADDR	4	
DTBBITM	0	
DTBBMID	0	
DTBHDR	0	
DTBTMR	0	
DTBTRK#	8	
DTBTRK#P	8	
DTC#DIRB	8	
DTCBITM#	8	
DTCCA128	0	40
DTCCTDES	0	80
DTCDATAS	4	20
DTCDCT	0	
DTCDFATT	0	
DTCDFIND	4	
DTCDICT	0	
DTCETDES	0	20
DTCFLAG1	0	
DTCFLDS	0	
DTCHURPN	0	
DTCICNT	154	
DTCIPHB	0	
DTCMAXD	0	
DTCRDKL	54	
DTCRDKLN	160	
DTCRSAL	14	
DTCSALT	158	

Name	Hex Offset	Hex Value
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	
DTSDSM	69	01
DTDSHDR	0	
DTDSHDRE	0	
DTDSMS	6	01
DTDSPER	6	02
DTDS_ACTUAL_CNT	0	
DTDS_VCTD	0	
DTDS_VCTI	1	
DTDS_FLAGS	2	
DTDS_APPMETA	2	80
DTDTPWD	6	40
DTDVCTD	67	
DTDVCTI	68	
DTDVOLCT	67	
DTEAPPMETA	18	10
DTEDSBLKSZ	4	
DTEDSBLOCKSIZE	0	

ADRTAPB Data Area

Name	Hex Offset	Hex Value
DTEDSHIBLSZ	0	
DTELCYL	4	
DTELOGCYL	0	
DTE_TCT_COMPR	18	04
DTEVBLKSZ	14	
DTEVBLOCKSIZE	10	
DTEVHIBLSZ	10	
DTHBLKSZ	D	
DTHDAY	0	
DTHDR	0	
DTHEFSAM	11	08
DTHFCMP	11	80
DTHFHCMP	11	10
DTHGNVI	8	80
DTHGT64K	8	20
DTHGVI	8	40
DTHIND1	11	
DTHIND2	8	
DTHLEN	9	
DTHLVLNO	C	
DTHNDS	F	
DTHSFER	11	20
DHTIMD	0	
DHTIME	4	
DTHUNLCD	11	40
DTHVERNO	B	
DTLCAT	1	
DTLDSCMP	0	
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	
DTM_TCT_COMPR	16	04
DTMTIME	16	10
DTMTRKCP	C	

Name	Hex Offset	Hex Value
DTMTRKCY	12	
DTMVDATA	0	
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	
DTPDSHDR	A	20
DTPDSNL	A	40
DTPDTRLR	A	04
DTPDUMPC	B	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	
DTPSELF	B	02
DTPSEQNO	0	
DTPSPHDR	A	01
DTPTHDR	A	80
DTPTRK0	A	20
DTPVER#	C	
DTPVHDR	A	80
DTPVOLD	A	10
DTPVTOC	A	10
DTPVTRLR	A	04
DTPVVD	A	02
DTRRECNT	0	c
DTRSERL	0	
DTRTLR	0	

ADRTAPB Data Area

Name	Hex Offset	Hex Value
DTSAIXNM	0	
DTSAIXS	0	
DTSAUTHR	32	
DTSAUTNM	2A	
DTSCNTLP	8	
DTSCODEN	20	
DTSEDEDS	0	
DTSDEND	4	
DTSEVOL	0	
DTSDHDR	0	
DTSDIND1	3	
DTSDLAST	3	80
DTSDLEN	0	
DTSDTYPE	2	
DTSERDAY	63	
DTSEXCNY	65	
DTSEXPRI	62	
DTSEYEAR	62	
DTSINFO	0	
DTSLEN	0	
DTSMSTRP	0	
DTSNMAT	28	
DTSPASSW	0	
DTSPATHA	58	
DTSPATHD	0	
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
DTREADP	18	
DTSUPDAT	10	
DTTBODY	18	
DTTCCHH	0	
DTTCCHH	3	
DTTCCHR	0	
DTTCKD	0	
DTTCNT	0	

Name	Hex Offset	Hex Value
DTTCRCRD	4	
DTTDATLN	6	
DTTHDR	0	
DTTINVT	2	08
DTTIOER	2	80
DTTKD	0	
DTTKELEN	5	
DTTLRCNT	7	
DTTRODAT	10	
DTTSTAT	2	04
DTTCMP	2	20
DTTRK	0	
DTTRKID	2	
DTTRKLN	0	
DTTROVF	2	40
DTTVFRST	2	10
DTTXDLEN	B	
DTVBLKSZ	C	
DTVBMSZ	25	
DTVCPVOL	24	10
DTVCVAF	24	20
DTVDAY	17	
DTVDEVTY	1F	
DTVEND	0	
DTVFCMP	24	08
DTVFHCMP	27	02
DTVGVI	27	20
DTVGVI	27	10
DTVIND1	24	
DTVIND2	27	
DTVLN	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

Name	Hex Offset	Hex Value
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	

z/OS UNIX file format

For a z/OS UNIX file backup, the format of the DFSMSdss dump data set contains the following data in the following sequence:

1. Path header record
2. UNIX file header record – describes the source working directory
3. File extended ACL entry (if any) – describes the source working directory
 - a. There can be more than one of these records.
4. List of potential UNIX files record
5. UNIX file header record
6. File extended ACL entry (if any)
 - a. There can be more than one of these records
7. Data bytes records (for symbolic link or regular file types only)
8. 2 UNIX file trailer records

Note:

1. For each UNIX file, repeat items 5-8

z/OS UNIX file backups utilize version 2 of the ADRTAPB. For description of every record that was introduced in the version 2 of the ADRTAPB, refer to ADRTAPB version 2 section. The version can be identified by looking at the DTPVER# field in the ADRTAPB structure at offset (0x0C). If DTPVER# = 2, then a version 2 ADRTAPB is being used.

ADRTAPB version 2 data area

1	ADRTAPB					
OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION	
0	(0)	STRUCTURE	64	BFRPREFIX	BUFFER PREFIX AREA@V22H	
0	(0)	UNSIGNED	2	BPSCOC	OFFSET IN BUFFER FROM BEGINNING OF R0 DATA TO COUNT FIELD IN ERROR	
2	(2)	UNSIGNED	2	BPSCOD	OFFSET IN BUFFER FROM BEGINNING OF R0 DATA TO DATA FIELD IN ERROR	
4	(4)	SIGNED	4	BPHWMRK	ADDRESS OF NEXT AVAILABLE BYTE	

```

      8      (8) CHARACTER      56  BPWORKA      (HIGH WATER MARK OF USED
      64     (40) CHARACTER      0   BPCBODY      STORAGE)
                                           WORK AREA FOR
                                           OPTIMIZE@V22H
                                           LOCATION OF REST OF BUFFER

```

Mapping of main prefix area (first thing in every record).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	32	ADRTAPB	
0	(0)	CHARACTER	6	DTPSCHK	
0	(0)	SIGNED	4	DTPSEQNO	SEGMENT SEQUENCE NUMBER
4	(4)	UNSIGNED	1	DTPNOSEG	NUM OF SEGMENTS PER RECORD
5	(5)	UNSIGNED	1	DTPSEGNO	SEGMENT NUM OF RECORD
6	(6)	UNSIGNED	2	DTPSEGLN	SEGMENT LENGTH INCL PREFIX
8	(8)	UNSIGNED	1	DTPPFXLN	LENGTH OF PREFIX VERS < 2 = CONSTANT(16) OR VERS 2 = CONSTANT(30)
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 COMPATIBILITY BETWEEN V2.1, V2.2 AND V2.3
	 1...		DTPPATH	08 - UNIX FILE DUMP
	111		*	Available
10	(A)	BIT(8)	1	DTPRCID1	RECORD IDENTIFIER 1 80 - DUMP HEADER
		1... ..		DTPVHDR	- VOLUME HEADER (SEE DTVOL)
		1... ..		DTPTHDR	- TAPE HEADER (see DTHDR)
		1... ..		DTPUHDR	- UNIX DUMP HEADER (see DTFHDR)
		.1... ..		DTPBTM	40 - DUMP HEADER - MAP OF DUMPED TRKS (see DTBTMR)
		.1... ..		DTPDSNL	- DS NAME/CATALOG LST (see DTLDNS)
		.1... ..		DTPFNL	- UNIXFILE NAME LIST
		.1... ..		DTPTRK0	20 - - TRACK 0 (see DTTTRK)
		.1... ..		DTPDSHDR	- DATA SET HEADER (see DTDSDR)
		.1... ..		DTPFLHDR	- UNIX FILE HEADER
		...1		DTPVTOC	10 - - VTOC TRACK (see DTTTRK)
		...1		DTPVOLD	- VOLUME DEFINITION (see DTMVOL)
		...1		DTPACL	- UNIXFILE ACLS
	 1...		DTPDATA	08 - DATA TRACK (see DTTTRK)
	 1...		DTPFDATA	- UNIXFILE BYTES
	1..		DTPVTRLR	04 - VOLUME TRAILER (see DTRTLR)
	1..		DTPDTRLR	- DS TRAILER (see DTRTLR)
	1..		DTPFTRLR	- UNIXFILE TRAILER
	1..		DTPVVDS	02 - VVDS TRACK (see DTTTRK)
	1		DTPSPHDR	01 - SPHERE REC HEADER (see DTSPPHRE)
11	(B)	BIT(8)	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 NON-TRACK DATA SET (EG, 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 DUMPCOND VOL
	1..		DTPSELF0	SELF DESC RECORDS EXIST
	1		*	RESERVED
12	(C)	CHARACTER	2	DTPVER#	ADRTAPB VERSION NUMBER
14	(E)	CHARACTER	2	*	RESERVED
16	(10)	STRUCTURE	16	DTPV2	Extend to Version 2
16	(10)	UNSIGNED	4	SEGLN	Segment Length

20	(14)	CHARACTER	12	*	Available
32	(20)	CHARACTER	0	DTPBODY	START OF REMAINDER OF RECORD

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	16	DTPVER2	TAPBVERSION 2 Expansion
0	(0)	UNSIGNED	4	SEGLEN	Segment Length
4	(4)	CHARACTER	12	*	Available

REST OF VOLUME HEADER RECORD (follows ADRTAPB).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	46	DTVOL	
0	(0)	CHARACTER	4	DTVTOCB	VTOT BEGINNING CCHH
4	(4)	CHARACTER	4	DTVTOCE	VTOT ENDING CCHH
8	(8)	CHARACTER	4	DTVOLSZ	VOLUME SIZE (DS4DEVST)
8	(8)	UNSIGNED	2	DTVLOGCY	# OF CYLINDERS IN VOLUME
10	(A)	UNSIGNED	2	DTVTRKCY	# 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	VTOT INDICATORS
23	(17)	CHARACTER	8	DTVTIME	DATE & 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	DEVTYPE OF VOLUME(UCBTBYT4)
35	(23)	UNSIGNED	1	DTVMODNO	MODEL NUMBER
36	(24)	BIT(8)	1	DTVIND1	VOLUME TYPE INDICATORS
		1... ..		DTVVRT	80 - VIRTUAL VOLUME
		.1... ..		DTVMINI	40 - MINI DISK
		.1... ..		DTVCVAF	20 - VOLUME HAS INDEXED VTOT
		...1... ..		DTVCPVOL	10 - CP volume fmt
		... 1... ..		DTVFCMP	08 - FILE COMPRESSED
	1... ..		DTVUNLCD	04 - UNALLOCATED SPACE DUMPED
	1... ..		DTVLVF	02 - OTHER LOGICALVOLUMES MAY FOLLOW THIS LOGICALVOLUME
	1... ..		DTVVVDS	01 - VVDS DATASET DUMPED BEFORE VTOT
37	(25)	UNSIGNED	2	DTVBMSZ	BITMAP SIZE IN WORDS
39	(27)	BIT(8)	1	DTVIND2	VOLUME TYPE INDICATORS
		1... ..		DTVLNVI	80-NONVSAM DATA SETS NOT ON VOLUME
		.1... ..		DTVLVI	40-VSAM DATA SETS NOT ON VOLUME
		..1... ..		DTVGNI	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 COMPRESS USED
	1		*	UNUSED
40	(28)	UNSIGNED	2	DTVLEN	LEN OF HEADER (SEGMENT LEN - PREFIX
42	(2A)	UNSIGNED	1	DTVVERNO	DFDSS VERSION NUMBER
43	(2B)	UNSIGNED	1	DTVLVLNO	DFDSS MODIFIC. NUMBER
44	(2C)	UNSIGNED	1	DTVXTNO	NUMBER OF VVDS EXTENTS
45	(2D)	UNSIGNED	1	DTVXTOF	OFFSET TO VVDS EXTENT

Mapping of additional volume information (follows DTVOL).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTVEND	FIELDS FROM DTVOL START
0	(0)	CHARACTER	0	*	

Mapping of VVDS extents (in DTVEND).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	10	DTVXTNT(*)	VVDS EXTENTS, IN DSCB FORMAT
0	(0)	CHARACTER	10	DTVXLN	

REST OF LOG DATA SET DUMP TAPE HEADER (follows ADRTAPB).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	18	DTHDR	

0	(0)	CHARACTER	8	DTHTIMD	DATE & 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
		...1		DTHLGDS#	DATA SETS ON VOLUME
	 1111		*	EXTEND NUM-DS AREA USED
9	(9)	UNSIGNED	2	DTHLEN	RESERVED
11	(B)	UNSIGNED	1	DTHVERNO	HEADER LEN
12	(C)	UNSIGNED	1	DTHLVNO	DFDSS VERSION NUMBER
13	(D)	UNSIGNED	2	DTHBLKSZ	DFDSS MODIFIC. NUMBER
15	(F)	UNSIGNED	2	DTHNDS	MAX BLKSIZE
17	(11)	CHARACTER	1	DTHIND1	# DS IN LIST
		1... ..		DTHFCMP	INDICATORS
		.1..		DTHUNLCD	FILE COMPRESSED
		.1.		DTHSFER	UNALLOCATED SPACE DUMPED
		...1		DTHFCMP	SPHERE OPTION
	 1...		DTHFSAM	HARDWARE COMPRESSION
	111		*	EF SAM DS HAVE DTTTRK
					RESERVED
OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
=====	=====	=====	=====	=====	=====
REST OF DATA SET NAME/CATALOG LIST (follows ADRTAPB).					
=====	=====	=====	=====	=====	=====
OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
=====	=====	=====	=====	=====	=====
0	(0)	STRUCTURE	45	DTLDSN	
0	(0)	UNSIGNED	1	DTLLEN	LENGTH OF DATA SET NAME
1	(1)	CHARACTER	44	DTLCAT	CATALOG NAME
Mapping of compressed data set name (follows DTLDSN).					
=====	=====	=====	=====	=====	=====
OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
=====	=====	=====	=====	=====	=====
0	(0)	STRUCTURE	*	DTLDSCMP	COMPRESSED DATA SET NAME
0	(0)	CHARACTER	0	*	
REST OF DATA SET HEADER RECORD (follows ADRTAPB).					
=====	=====	=====	=====	=====	=====
OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
=====	=====	=====	=====	=====	=====
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 VOLS FOR DATA SET
6	(6)	BIT(8)	1	DTDIND	DATA SET INDICATOR
		1... ..		DTDIPWD	PASSWORD SUPPLIED 1=YES
		.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 VOL CNT (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 & 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 ADTL INFO
	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		DTSDM	USE SYSTEM DATA MOVER
SPHERE RECORD FOR CATALOG FILTER DUMP (follows ADRTAPB).					
=====	=====	=====	=====	=====	=====

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	4	DTSPHERE	
0	(0)	SIGNED	4	DTSLEN	LENGTH OF SPHERE RECORD

Mapping of AIX sphere information (follows DTSPHERE).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	D TSAIXS	AIX SPHERE INFO
0	(0)	CHARACTER	0	*	

Mapping of sphere information (in D TSAIXS).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	102	DTSINFO	SPHERE INFORMATION
0	(0)	CHARACTER	44	D TSAIXNM	AIX NAME
44	(2C)	CHARACTER	44	DTSPATHN	PATH NAME
88	(58)	CHARACTER	1	DTSPATHA	PATH ATTRIBUTE
89	(59)	BIT(8)	1	DTSPATHF	PATH INFO FLAG
		1... ..		DTSPATHI	IF SET, PATH OWNERID IS CONTAINED IN THIS BLOCK (SEE DTSPTHON FOR DISCRIPTION)
		.1... ..		DTSPATHE	PATH EXPIRATION DATE IS CONTAINED IN THIS BLOCK (SEE DTSPTHEP FOR DISCRIPTION). IF NOT SET, IGNORE THE EXPIR FIELDS.
		..1.		DTSPATHP	PATH HAS PASSWORD THAT IS CONTAINED IN THIS BLOCK (SEE DTSEXPIR FOR DISCRIPTION). IF EXISTS, BEGINS AT DTSPATHD.
		...1 1...		DTSRACF	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 DECIM
99	(63)	UNSIGNED	2	DTSERDAY	12-BITS DAY OF IN DECIMAL & 4-BIT SIGN CHAR THAT ALLOWS TO BE UNPACKED
101	(65)	UNSIGNED	1	DTSEXCNY	CENTURY BYTE IN DECIMAL

Mapping of path security information (follows DTSINFO).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTSPATHD	START PATH SECURITY INFO
0	(0)	CHARACTER	0	*	

Mapping of security info table (in DTSPATHD).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	52	DTSPASSW	SECURITY INFO TABLE
0	(0)	CHARACTER	8	D TSMSTRP	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	D TSAUTNM	ADDR OF AUTHORIZATION MOD
50	(32)	SIGNED	2	D TSAUTHR	AUTHORIZATION REC LENGTH

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	D SAUTHR	AUTHORIZATION RECORD.
0	(0)	CHARACTER	0	*	

COMMON DATA FORMAT DATA SET ATTRIBUTES (follows DTDSHDR).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
-------------------	---------------	------	--------	------------	-------------

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
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	0	*	LAST

Mapping for additional data (follows DTCDFATT).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTDSAIR	
0	(0)	SIGNED	4	DTDSIDL	ADD. DATA LENGTH
4	(4)	CHARACTER	*	DTDSID	ADD. DATA FOLLOWS

REST OF VOLUME DEFINITION RECORD (follows ADRTAPB).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
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	#BYTES/TRK (TRACK CAPACITY WITH OVERHEAD).
16	(10)	UNSIGNED	2	DTMLOGCY	# CYLINDERS PER VOLUME
18	(12)	UNSIGNED	2	DTMTRKCY	# TRACKS PER VOLUME
20	(14)	UNSIGNED	2	DTMMAXCB	MAX COMPRESS BUF 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 BW0
22	(16)	BIT(11) POS(6)	2	*	unused
24	(18)	UNSIGNED	1	DTM#VVRS	# OF VVRS/NVRS DUMPED
25	(19)	UNSIGNED	1	DTM#DSCB	# OF DSCBS DUMPED
26	(1A)	UNSIGNED	1	DTM#EXT	# OF EXTENTS DUMPED
27	(1B)	CHARACTER	1	DTMMODNO	MODEL NUMBER

Mapping of the DSCBs, extents, and VVRs (follows DTMVOL).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTMVDATA	DSCBS (1&2) FOR DS, FOLLOWED BY EXTENT LIST, FOLLOWED BY VVRS OR NVR DATA SET CLUSTER.
0	(0)	CHARACTER	0	*	

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTMVDATA	DSCBS (1&2) FOR DS, FOLLOWED BY EXTENT LIST, FOLLOWED BY VVRS OR NVR DATA SET CLUSTER.
0	(0)	CHARACTER	0	*	

REST OF RECORD THAT MAPS THE TRACKS DUMPED (follows ADRTAPB).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
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#	# OF TRKS MAPPED IN BMBB SEGMENT
8	(8)	UNSIGNED	2	DTBTRK#P	# OF TRKS MAPPED IN BMBB SEGMENT
10	(A)	CHARACTER	2	*	RESERVED

Mapping of the bitmap data (follows DTBTMR).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTBBITM	BITMAP
0	(0)	CHARACTER	0	*	

REST OF TRACK RECORD (follows ADRTAPB).

The first segment of the track is preceded by DTTRK.
The second and subsequent track segments are not preceded by DTTRK, but are preceded by ADRTAPB. The remaining track

image data begins immediately after ADRTAPB.

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	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 TRK
2	(2)	CHARACTER	1	DTTTRKID	TRACK INDICATORS
		1... ..		DTTIOER	I/O ERROR ON TRACK
		.1... ..		DTTTROVF	LAST REC ON TRK IS OVFLD REC (maintained for restore compatibility only, do not set)
		..1.		DTTTCMP	IF ON, TRACK COMPRESSED
		...1		DTTVFRST	FIRST VVDS RECORD
	 1...		DTTINVT	INVALID TRACK FORMAT
	1..		DTTSTAT	USER STATISTICAL RECORD
3	(3)	UNSIGNED	4	DTTCCHH	CCHH OF TRACK 2@SVD
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)	UNSIGNED	1	DTTXDLEN	EXTRA ENC DATA LEN
12	(C)	CHARACTER	4	*	RESERVED
16	(10)	CHARACTER	8	DTTR0DAT	RCRD 0 DATA, ONLY ON 1ST SEG
24	(18)	CHARACTER	0	DTTBODY	R1- RN RECORDS ON TRACK

Count field format (in track image data).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	8	DTTCKD	CNT, KEY & DATA FIELDS ON TRK
0	(0)	CHARACTER	8	DTTCNT	COUNT FIELD
0	(0)	CHARACTER	5	DTTCCHHR	CCHHR OF RECORD
0	(0)	UNSIGNED	4	DTTCCCHH	CCHH OF RECORD 2@SVD
4	(4)	UNSIGNED	1	DTTCRCRD	R OF RECORD
5	(5)	UNSIGNED	1	DTTKELEN	KEY LENGTH
6	(6)	UNSIGNED	2	DTTDATLN	DATA LENGTH

Key or Data field mapping (follows count field).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTTKD	KEY AND DATA FIELDS
0	(0)	CHARACTER	0	*	

REST OF RECORD THAT MAPS THE TRAILER RECORD (follows ADRTAPB).

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
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
OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	4	DTSDHDR	
0	(0)	UNSIGNED	2	DTSDLEN	LEN OF SELF DESC REC
2	(2)	CHARACTER	1	DTSDTYPE	TYPE OF SELF DSC REC
3	(3)	CHARACTER	1	DTSDIND1	SELF DESC REC FLAGS
		1... ..		DTSDLAST	LAST SELF DESC REC
		.111 1111		*	RESERVED
4	(4)	CHARACTER	0	DTSDEND	START SELF DESC REC ON A WORD BOUNDARY
OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	356	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
4	(4)	CHARACTER	16	DTCDATAS	SAMPLE DATA

20	(14)	CHARACTER	64	DTCRSAL	RSA LABEL
84	(54)	CHARACTER	256	DTCRDKL	RSA CIPH DATA LABEL
340	(154)	SIGNED	4	DTCICNT	ITERATION COUNT
344	(158)	CHARACTER	8	DTCsalt	SALT FOR ADPRWKEY
352	(160)	SIGNED	4	DTCRDKLN	RSA CIPH DATA LAB LN@DDE

COMPRESSION DICTIONARY BLOCK. Immediately follows a
self-describing record header

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	16	DTCDCCT	FOLLOWS A DTSDHDR
0	(0)	CHARACTER	16	DTCFLDS	
0	(0)	SIGNED	4	DTCMAXD	MAX DICT SIZE
4	(4)	CHARACTER	12	*	RESERVED

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTCDDICT	EXPANSION DICTIONARY AFTER DTCDCCT
0	(0)	CHARACTER	0	*	AREA

EXTENDED VOLUME RECORD FOUND IN FIRST TAPE RECORD

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	52	DTSDEVOL	EXTENDED VOL SD REC
0	(0)	SIGNED	4	DTELOGCYL	# OF CYLINDERS - EAV
4	(4)	UNSIGNED	2	DTELCYL	LARGE UNIT REGION
6	(6)	CHARACTER	2	*	RESERVED
8	(8)	UNSIGNED	4	DTCBITM#	SIZE OF CHUNK MAP
12	(C)	CHARACTER	4	*	RESERVED
16	(10)	BIT(64)	8	DTEVBLOCKSIZE	BLOCKSIZE
16	(10)	UNSIGNED	4	DTEVHIBLSZ	HIGH WORD BLKSIZE

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
20	(14)	UNSIGNED	4	DTEVBLKSZ	LOW WORD BLKSIZE WHEN DTVBLKSZ=0
24	(18)	BIT(8)	1	DTEFLGS	EXT VOLUME FLAG BYTE
		1... ..		DTERESET	RESET CODED ON DUMP
		.1... ..		DTECYLMG	CYLINDER MGD SPC
		..1... ..		DTEVZCOMP	ZCOMPRESSION USED
		...1 1111		*	UNUSED
25	(19)	CHARACTER	27	*	RESERVED

EXTENDED TAPE RECORD FOUND IN FIRST TAPE RECORD

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	52	DTSDEHDR	EXT TAPE RECORD
0	(0)	BIT(64)	8	DTEHBLOCKSIZE	BLOCKSIZE
0	(0)	UNSIGNED	4	DTEHHIBLSZ	HIGH WORD BLKSIZE
4	(4)	UNSIGNED	4	DTEHBLKSZ	LOW WORD BLKSIZE WHEN DTHBLKSZ=0
8	(8)	BIT(8)	1	DTETFLGS	EXT TAPE FLAG BYTE
		1... ..		DTETZCOMP	ZCOMPRESSION USED
		.111 1111		*	UNUSED
9	(9)	CHARACTER	3	*	UNUSED
12	(C)	BIT(64)	8	DTEDS#	ExtendNum Dsets area Used if DTHLGDS# is ON
12	(C)	UNSIGNED	4	DTEDS#HI	Hi boundary count
16	(10)	UNSIGNED	4	DTEDS#LO	Lo boundary count
20	(14)	CHARACTER	32	*	RESERVED

CHECKSUM RECORD CONTAINING CHECKSUM OF DATA UP TO THIS ONE

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	68	DTCHKSUM	CHECKSUM RECORD
0	(0)	CHARACTER	1	DTCKTYPE	TYPE OF CHECKSUM
		1... ..		DTCK_MD5	MD5 HASH
		.111 1111		*	UNUSED
1	(1)	CHARACTER	3	*	UNUSED
4	(4)	CHARACTER	64	DTCKHASH	HASH VALUE
4	(4)	CHARACTER	16	DTCK_MD5HASH	MD5 HASH VALUE

DATASET HEADER EXTENTION

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
-------------------	---------------	------	--------	------------	-------------

0	(0)	STRUCTURE	24	DTDSHRE	DATASET HDR EXTENTION@MCA
0	(0)	CHARACTER	2	DTDS_ACTUAL_CNT	ACTUAL VOLUME COUNT
0	(0)	UNSIGNED	1	DTDS_VCTD	DATA COMP ACTUAL CNT
1	(1)	UNSIGNED	1	DTDS_VCTI	INDEX COMP ACTUAL CNT@MCA
2	(2)	CHARACTER	2	DTDS_FLAGS	DS HDR EXT FLAGS
		1... ..		DTDS_APPMETA	1=APPMETA OBJ STORED
2	(2)	BIT(15) POS(2)	2	*	AVAILABLE
4	(4)	CHARACTER	20	*	AVAILABLE

UNIX FILE SUPPORT STRUCTURES

! UNIX FILE DUMP TAPE HEADER

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	1059	DTFHDR	
0	(0)	CHARACTER	8	DTFTIMD	DATE & TIME/DAY OF DUMP
0	(0)	CHARACTER	4	DTFDAY	DAY
4	(4)	CHARACTER	4	DTFTIME	TIME
8	(8)	UNSIGNED	2	DTFLEN	HEADER LEN
10	(A)	UNSIGNED	1	DTFVERNO	DFDSS VERSION NUMBER
11	(B)	UNSIGNED	1	DTFLVLNO	DFDSS MODIFIC. NUMBER
12	(C)	BIT(64)	8	DTFBLKSIZE	BLOCKSIZE
12	(C)	UNSIGNED	4	DTFHIBLKSZ	HIGH WORD BLKSIZE
16	(10)	UNSIGNED	4	DTFLOBLKSZ	LOW WORD BLKSIZE
20	(14)	CHARACTER	8	DTFNUM	# FILES IN LIST
28	(1C)	CHARACTER	1	DTFIND1	INDICATORS
		1... ..		DTFZCOMP	zEDC COMPRESSION
		.111 1111		*	AVAILABLE
29	(1D)	CHARACTER	3	*	AVAILABLE
32	(20)	CHARACTER	1027	DTFWORKDIR	WORKINGDIRECTORY INFO
32	(20)	UNSIGNED	4	DTF_WDL	WORKINGDIRECTORY LENGTH
36	(24)	CHARACTER	1023	DTF_WD	WORKINGDIRECTORY

! UNIX FILE PATH LIST

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTFLST	
0	(0)	UNSIGNED	4	DTFLLEN	LENGTH OF PATH NAME
4	(4)	CHARACTER	*	DTFLNAME	PATH NAME

! UNIX FILE PATH HEADER

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTFLHDR	
0	(0)	UNSIGNED	4	DTFPLEN	LENGTH OF PATH NAME
4	(4)	CHARACTER	*	DTFPNAME	PATH NAME

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	250	*	
0	(0)	CHARACTER	250	DTFATTR	FILE ATTRIBUTES
250	(FA)	CHARACTER	0	DTFATTR_END	END OF FILE ATTRIBUTES

! UNIX FILE ACCESS CONTROL LISTS

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	*	DTFACL	
0	(0)	UNSIGNED	2	DTFACL#	NUMBER OF ENTRIES
2	(2)	UNSIGNED	2	DTFACLEN	LENGTH OF ENTRIES
4	(4)	UNSIGNED	1	DTFACLVERS	FACL VERSION
5	(5)	CHARACTER	1	DTFACLT	ACL TYPE=IFSP_FLAG2
		1... ..		DTFACCESS	ACCESS ACL
		.1... ..		DTFFMODEL	FILE MODEL
		.1... ..		DTFDMODEL	DIRECTORY MODEL
6	(6)	UNSIGNED	2	DTFACL#USERS	NUMBER OF USER ENTRIES
8	(8)	CHARACTER	*	DTFACLE	FILE ACL=IRRPFACL

! UNIX FILE DATA (BYTE STREAM)

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	16	DTFBYTES	
0	(0)	CHARACTER	16	DTFBHDR	HEADER FOR BYTES SEGMENT
0	(0)	CHARACTER	1	DTFBIND1	BYTE SEGMENT INDICATORS
		1... ..		DTFBCMP	ON = BYTES COMPRESSED
		.111 1111		*	
1	(1)	CHARACTER	3	*	AVAILABLE

4	(4)	CHARACTER	8	DTFBOFF	OFFSET INTO FILE
12	(C)	UNSIGNED	4	DTFBLEN	COUNT OF BYTES
16	(10)	CHARACTER	0	DTFBODY	BYTES 1 THRU DTFBLEN

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	12	DTFTLR	
0	(0)	CHARACTER	8	DTFBCNT	NUMBER OF BYTES DUMPED
8	(8)	CHARACTER	4	*	

AVAILABLE 0 2

ADRTAPB constants

ADRTAPB cross-reference

1 CROSS REFERENCE

NAME	HEX OFFSET	HEX VALUE	LEVEL
=====	=====	=====	=====
ADRTAPB	0		1
BFRPREFIX	0		1
BPBODY	40		2
BPHWMRK	4		2
BPSCOC	0		2
BPSCOD	2		2
BPWORKA	8		2
DSAUTHR	0		1
DTBADDR	4		3
DTBBITM	0		1
DTBBMID	0		3
DTBHDR	0		2
DTBTMR	0		1
DTBTRK#	8		3
DTBTRK#P	8		4
DTC#DIRB	8		2
DTCBITM#	8		2
DTCCA128	0	40	3
DTCCTDES	0	80	3
DTCDATAS	4		2
DTCDCCT	0		1
DTCDFATT	0		1
DTCDFIND	4		2
DTCDICT	0		1
DTCETDES	0	20	3
DTCFLAG1	0		2
DTCFLDS	0		2
DTCHKSUM	0		1
DTCHURPN	0		2
DTCICNT	154		2
DTCIPHB	0		1
DTCK_MD5	0	80	3
DTCK_MD5HASH	4		3
DTCKHASH	4		2
DTCKTYPE	0		2
DTCMAXD	0		3
DTCRDKL	54		2
DTCRDKLN	160		2
DTCRSAL	14		2
DTCSALT	158		2
DTDAIXSP	69	80	3
DTDALIAS	6	04	3
DTDCAT	F		2
DTDCATLN	1		2
DTDCDF	69	40	3
DTDDSN	38		2
DTDDSORG	2		2
DTDIND	6		2
DTDIND2	69		2
DTDIPWD	6	80	3
DTDIRACF	6	20	3
DTDLEN	0		2
DTDNOIDX	69	04	3
DTDNTALL	69	10	3
DTDNVOL	5		2

1 CROSS REFERENCE

NAME	HEX OFFSET	HEX VALUE	LEVEL
=====	=====	=====	=====
DTDOPTCD	4		2
DTDPDSE	69	20	3
DTDPDSET	69	02	3
DTDPDSEX	4	80	3
DTDPWD	7		2
DTDRACFD	6	10	4
DTDRACFG	6	08	4
DTDRACFP	6	18	3
DTDS_ACTUAL_CNT	0		2
DTDS_APPMETA	2	80	3
DTDS_FLAGS	2		2
DTDS_VCTD	0		3
DTDS_VCTI	1		3
DTDSAI	69	08	3
DTDSAID	4		2
DTDSAIDL	0		2
DTDSAIR	0		1
DTSDSM	69	01	3
DTDSHDR	0		1
DTDSHDRE	0		1
DTDSMS	6	01	3
DTDSPER	6	02	3
DTDSHDRE	0		
DTDS_ACTUAL_CNT		0	
DTDS_VCTD			0
DTDS_VCTI			1
DTDS_FLAGS			2
DTDTPWD	6	40	3
DTDVCTD	67		3
DTDVCTI	68		3
DTDVOLCT	67		2
DTECYLMG	18	40	3
DTEDS#	C		2
DTEDS#HI	C		3
DTEDS#LO	10		3
DTE_TCT_COMPR	18		04
DTEAPPMETA	18		10
DTEFLGS	18		2
DTEHBLKSZ	4		3
DTEHBLOCKSIZE	0		2
DTEHHIBLKSZ	0		3
DTELCYL	4		2
DTELOGCYL	0		2
DTERESET	18	80	3
DTETFLGS	8		2
DTETZCOMP	8	80	3
DTEVBLKSZ	14		3
DTEVBLOCKSIZE	10		2
DTEVHIBLKSZ	10		3
DTEVZCOMP	18	20	3
DTF_WD	24		3
DTF_WDL	20		3
DTFACCESS	5	80	3
DTFACL	0		1
DTFACL#	0		2
DTFACL#USERS	6		2
DTFACLE	8		2
DTFACLEN	2		2
DTFACLT	5		2
DTFACLVERS	4		2
DTFATTR	0		2
DTFATTR_END	FA		2

1 CROSS REFERENCE

NAME	HEX OFFSET	HEX VALUE	LEVEL
=====	=====	=====	=====
DTFBCMP	0	80	4
DTFBCNT	0		2
DTFBHDR	0		2
DTFBIND1	0		3
DTFBLEN	C		3
DTFBLKSIZE	C		2
DTFBODY	10		2
DTFBOFF	4		3

	DTFBYTES	0		1
	DTFDAY	0		3
	DTFDMODEL	5	20	3
	DTFFMODEL	5	40	3
	DTFHDR	0		1
	DTFHIBLSZ	C		3
	DTFIND1	1C		2
	DTFLEN	8		2
	DTFLHDR	0		1
	DTFLLEN	0		2
	DTFLNAME	4		2
	DTFLOBLKSZ	10		3
	DTFLST	0		1
	DTFLVLNO	B		2
	DTFNUM	14		2
	DTFPLEN	0		2
	DTFPNAME	4		2
	DTFTIMD	0		2
	DTFTIME	4		3
	DTFTLR	0		1
	DTFVERNO	A		2
	DTFWORKDIR	20		2
	DTFZCOMP	1C	80	3
	DTHBLKSZ	D		2
	DTHDAY	0		3
	DTHDR	0		1
	DTHEFSAM	11	08	3
	DTHFCMP	11	80	3
	DTHFHCMP	11	10	3
	DTHGNVI	8	80	3
	DTHGT64K	8	20	3
	DTHGVI	8	40	3
	DTHIND1	11		2
	DTHIND2	8		2
	DTHLEN	9		2
	DTHLGDS#	8	10	3
	DTHLVLNO	C		2
	DTHNDS	F		2
	DTHSFER	11	20	3
	DHTIMD	0		2
	DHTIME	4		3
	DTHUNLCD	11	40	3
	DTHVERNO	B		2
	DTLCAT	1		2
	DTLDSCMP	0		1
	DTLDSN	0		1
	DTLLEN	0		2
1	CROSS REFERENCE			
	NAME	HEX OFFSET	HEX VALUE	LEVEL
	=====	=====	=====	=====
	DTM#DSCB	19		2
	DTM#EXT	1A		2
	DTM#VVRS	18		2
	DTM_TCT_COMPR	16		04
	DTMBWOT	16	08	3
	DTMCVAF	16	20	3
	DTMDEVTY	6		2
	DTMIND	16		2
	DTMLOGCY	10		3
	DTMMAXCB	14		2
	DTMMINI	16	40	3
	DTMMODNO	1B		2
	DTMTIME	16	10	3
	DTMTRKCP	C		3
	DTMTRKCY	12		3
	DTMVDATA	0		1
	DTMVIRT	16	80	3
	DTMVOL	0		1
	DTMVOLSZ	C		2
	DTMVSERL	0		2
	DTPACL	A	10	5
	DTPBODY	20		2
	DTPBTM	A	40	3
	DTPBWQE	B	08	3
	DTPDASDD	9	20	3
	DTPDATA	A	08	3
	DTPDDDSP	B	40	3
	DTPDDISP	B	80	3
	DTPDMPID	9		2
	DTPDSHDR	A	20	4

	DTPDSNL	A	40	4
	DTPDTRLR	A	04	4
	DTPDUMPC	B	04	3
	DTPENDKR	B	10	3
	DTPENDNT	B	20	3
	DTPFDATA	A	08	4
	DTPFLHDR	A	20	5
	DTPFNL	A	40	5
	DTPFTRLR	A	04	5
	DTPFULD	9	80	3
	DTPLOGCL	9	10	3
	DTPNOSEG	4		3
	DTPPARTD	9	40	3
	DTPPATH	9	08	3
	DTPPFXLN	8		2
	DTPRCFL1	B		2
	DTPRCID1	A		2
	DTPSCHK	0		2
	DTPSEGLN	6		2
	DTPSEGN0	5		3
	DTPSELF0	B	02	3
	DTPSEQN0	0		3
	DTPSPHDR	A	01	3
	DTPTHDR	A	80	4
	DTPTRK0	A	20	3
	DTPUHDR	A	80	5
1	CROSS REFERENCE			

NAME	HEX OFFSET	HEX VALUE	LEVEL
=====	=====	=====	=====
DTPVER#	C		2
DTPVER2	0		1
DTPVHDR	A	80	3
DTPVOLD	A	10	4
DTPVTOC	A	10	3
DTPVTRLR	A	04	3
DTPVVDS	A	02	3
DTPV2	10		2
DTRRECNT	0		3
DTRSERL	0		2
DTRTLR	0		1
D TSAIXNM	0		2
D TSAIXS	0		1
D TSAUTHR	32		2
D TSAUTNM	2A		2
D TSCNTLP	8		2
D TSCODEN	20		2
D TSDHDR	0		1
D TSDEND	4		2
D TSDDEVOL	0		1
D TSDHDR	0		1
D TSDIND1	3		2
D TSDLAST	3	80	3
D TSDLEN	0		2
D TSDTYPE	2		2
D TSERDAY	63		4
D TSEXCNY	65		3
D TSEXP IR	62		3
D TSEYEAR	62		4
D T SINFO	0		1
D T SLEN	0		2
D TSMSTRP	0		2
D T SNUMAT	28		2
D TSPASSW	0		1
D TSPATHA	58		2
D TSPATHD	0		1
D TSPATHE	59	40	3
D TSPATHF	59		2
D TSPATHI	59	80	3
D TSPATHN	2C		2
D TSPATHP	59	20	3
D TSPHERE	0		1
D TSPRACF	59	18	3
D TSPTHEP	62		2
D TSPTHON	5A		2
D TSRACFD	59	10	4
D TSRACFG	59	08	4
D T SREADP	18		2
D T SUPDAT	10		2
D TTBODY	18		2
D TCCCHH	0		4

1	DTTCCHH	3		3
	DTTCCHHR	0		3
	DTTCKD	0		1
	DTTCNT	0		2
1	CROSS REFERENCE			
	NAME	HEX OFFSET	HEX VALUE	LEVEL
	=====	=====	=====	=====
	DTTCRCRD	4		4
	DTTDLN	6		3
	DTTHDR	0		2
	DTTINVT	2	08	4
	DTTIOER	2	80	4
	DTTKD	0		1
	DTTKELEN	5		3
	DTTLRCNT	7		3
	DTTR0DAT	10		2
	DTTSTAT	2	04	4
	DTTTCMP	2	20	4
	DTTTRK	0		1
	DTTTRKID	2		3
	DTTTRKLN	0		3
	DTTTR0VF	2	40	4
	DTTVFRST	2	10	4
	DTTXDLEN	B		3
	DTVBLKSZ	C		2
	DTVBMSZ	25		2
	DTVCPVOL	24	10	3
	DTVCVAF	24	20	3
	DTVDAY	17		3
	DTVDEVTY	1F		2
	DTVEND	0		1
	DTVFCMP	24	08	3
	DTVFCMP	27	02	3
	DTVGNI	27	20	3
	DTVGVI	27	10	3
	DTVIND1	24		2
	DTVIND2	27		2
	DTVLEN	28		2
	DTVLNVI	27	80	3
	DTVLOGCY	8		3
	DTVLVF	24	02	3
	DTVLVI	27	40	3
	DTVLVLNO	2B		2
	DTVMACB	E		2
	DTVMINI	24	40	3
	DTVMODNO	23		2
	DTVOL	0		1
	DTVOLSZ	8		2
	DTVSERL	10		2
	DTVSMS	27	08	3
	DTVSMSI	27	04	3
	DTVTIMD	17		2
	DTVTIME	1B		3
	DTVTOCB	0		2
	DTVTOCE	4		2
	DTVTRKCY	A		3
	DTVUNLCD	24	04	3
	DTVVERNO	2A		2
	DTVVIRT	24	80	3
	DTVVTOCI	16		2
	DTVVVDS	24	01	3
	DTVVXLEN	0		2
1	CROSS REFERENCE			
	NAME	HEX OFFSET	HEX VALUE	LEVEL
	=====	=====	=====	=====
	DTVVXTNO	2C		2
	DTVVXTNT	0		1
	DTVVXTOF	2D		2
	SEGLN	10		3
	SEGLN			

Chapter 14. DFSMSDss patch area

This topic describes how to customize certain DFSMSDss functions by setting flags in ADRPATCH, a module in the DFSMSDss load module (ADDRSSU). You can set the flags in ADRPATCH dynamically through the DFSMSDss SET PATCH auxiliary command, or you can set them in ADRPATCH permanently through the AMASPZAP program.

The SET PATCH offset=value command specifies that DFSMSDss set the patch byte, at the specified offset to the specified value. You must have READ access authorization to that profile to use the SET PATCH command. Your installation can limit use of the SET PATCH command with the RACF FACILITY class profile STGADMIN.ADR.PATCH.

The following sample JCL sets the flags in ADRPATCH. The specific offsets and values of the flags are explained in the sections that follow. For the mapping of the flags in ADRPATCH, see the ADRPTCHB data area description in [Table 21 on page 236](#).

Using RESET with CLONE

The RESET keyword, when used for UNIX file processing, specifies that DFSMSDss resets the last backup date field (ATTRREFTIME64) for regular files after the dump is complete. By default, the RESET keyword is ignored when the CLONE keyword is also specified with the DUMP command.

Setting the byte at offset X'5F' in ADRPATCH to a nonzero value causes DFSMSDss to perform RESET processing when both the RESET and CLONE keywords are specified with the DUMP command. In this case, the last backup date field is reset to the time of which CLONE 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 CLONE permanently, modify the sample JCL (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN      DD *
NAME ADDRSSU ADRPATCH
VER  5F      00
REP  5F      FF
```

If the clone dump is unsuccessful for any reason after the last backup date field (ATTRREFTIME64) has been reset, the last backup date field (ATTRREFTIME64) for the regular file 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 CLONE. Rather than using this patch to accomplish incremental backup in DFSMSDss with clone, use DFSMSHsm instead.

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 ADDRSSU 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. For information about the SET command, see [“Controlling task processing”](#) on page 508.

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 (see [“Sample JCL”](#) on page 213) 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 (see [“Sample JCL”](#) on page 213) 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 ADRLIP1 within ADRDSSU. The displacement of label ADRLIP1 is used in setting the slip trap.

```
//AMBLIST JOB...
//*
//*****
//*
//* SAMPLE JCL TO LOCATE THE DISPLACEMENT OF LABEL ADRLIP1
//* WITHIN ADRDSSU. THE DISPLACEMENT OF ADRLIP1 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=(ADRSSU,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.

Note:

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 (see [“Sample JCL” on page 213](#)) 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 (see [“Sample JCL” on page 213](#)) 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 (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN      DD *
  NAME  ADDRDSU  ADPATCH
  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 ADPATCH can be used to give that resource owner authority to the SMS constructs.

To use ADPATCH to set the revoked RESOWNER on, set the byte at offset X'11' in module ADPATCH to a X'FF'. You can use the SET PATCH command to set the patch byte dynamically or modify the sample JCL (see [“Sample JCL” on page 213](#)) as follows to set the patch byte permanently:

```
//SYSIN      DD *
  NAME  ADDRDSU  ADPATCH
  VER    11      00
  REP    11      FF
```

Note:

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 218](#).

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 ADPATCH 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.

Note:

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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
NAME ADDRDSU ADPATCH
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 ADPATCH. 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
NAME ADDRDSU ADPATCH
VER 13 00
REP 13 FF
```

Overwriting existing objects during logical data set DUMP to an object store cloud

During logical data set DUMP operations to an object storage cloud, DFSMSdss will perform a check to see if a backup using the same object prefix already exists in the specified container. If a backup already exists then DFSMSdss will fail the backup.

You can allow DFSMSdss to overwrite existing backups but all preexisting objects with the specified object prefix will be deleted. Use this patch only if you are willing to tolerate that all preexisting objects with the specified object prefix will be lost.

The function is affected by setting the flag at offset X'5D' in ADPATCH. The settings are listed as follows:

X'00'

DFSMSdss functions normally. DFSMSdss looks for a backup in the object storage cloud and container with the specified object prefix. If a backup using the same object prefix exists then DFSMSdss fails the backup.

Any setting other than X'00'

DFSMSdss bypasses the check for existing objects. If a backup exists in the specified object storage cloud, container and object prefix then the objects will be overwritten.

To set the flag to on dynamically, use the SET PATCH command.

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 ADPATCH. 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
      NAME ADRDSSU ADPATCH
      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 ADPATCH. 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
      NAME ADRDSSU ADPATCH
      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 ADPATCH. 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 (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN DD *
NAME ADDRDSU ADPATCH
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 ADPATCH. 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 (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN DD *
NAME ADDRDSU ADPATCH
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 ADPATCH 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 (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN DD *
NAME ADDRDSU ADPATCH
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 ADPATCH 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
  NAME  ADDRDSU  ADPATCH
  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 ADPATCH 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
  NAME  ADDRDSU  ADPATCH
  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 ADPATCH 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
      NAME  ADDRDSU  ADPATCH
      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 ADPATCH 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 20nn or 21nn and that were dumped without the fix for OW00780 are restored with an expiration date of 19nn.

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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
      NAME  ADDRDSU  ADPATCH
      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 ADPATCH 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
NAME  ADDRDSU  ADPATCH
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 uncatolog the data set until after all data sets are dumped.

This function is affected by setting the flag at offset X'1E' in ADPATCH. 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 uncatolog 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 uncatologing 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
NAME  ADDRDSU  ADPATCH
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 ADPATCH 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 (see [“Sample JCL” on page 213](#)) 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 ADDRSSU ADPATCH
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 ADPATCH 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 ADPATCH 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 ADPATCH 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 ADPATCH 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 (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN DD *
NAME ADDRDSU ADPATCH
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 ADPATCH. 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 (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN DD *
NAME ADDRDSU ADPATCH
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 ADPATCH. 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
  NAME  ADDRDSU  ADPATCH
  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 ADPATCH 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
  NAME  ADDRDSU  ADPATCH
  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 ADPATCH. 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.

Note:

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 (see “Sample JCL” on page 213) 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.

Note:

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 (see “Sample JCL” on page 213) 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 ADPATCH. 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 than X'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 (see [“Sample JCL”](#) on page 213) as follows:

```
//SYSIN DD *
      NAME ADDRDSU ADPATCH
      VER   42      00
      REP   42      FF
```

Changing the default DEFRAG processing of checkpointed data sets (OW20285)

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 ADPATCH. 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 (see [“Sample JCL”](#) on page 213) as follows:

```
//SYSIN DD *
      NAME ADDRDSU ADPATCH
```


VER	43	00
REP	43	FF

Setting the percentage to overallocate target data set space (OW27837)

When restoring or copying an extended format NONVSAM data set, DFSMSdss may not be able to calculate the exact amount of space the target data set will require. Therefore, it is possible for the target data set to be underallocated, causing RESTORE or COPY to fail with an MSGADR910E RC40000004 reason 0000000D. To prevent underallocating the target, a patch byte has been provided. The hexadecimal value set in this patch byte is the percentage by which an extended format non VSAM data set should be overallocated. The patch is at offset X'44' in ADPATCH.

Note: Set the patch only for those data sets that fail to restore or copy. Reset the patch to zero after the failing data set has been restored or copied, so that other extended format non VSAM data sets are not needlessly overallocated. The percentage of overallocated space required may vary by data set. In other words, some data sets may require an overallocation of 10%, and some may require additional space, so adjust the overallocation percentage as needed.

To use ADPATCH to overallocate the target data set by 10%, modify the sample JCL (see [“Sample JCL”](#) on page 213) as follows:

```
//SYSIN DD *
      NAME ADDRDSU ADPATCH
      VER   44      00
      REP   44      0A
```

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 ADPATCH. 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 (see [“Sample JCL”](#) on page 213) as follows:

```
//SYSIN DD *
      NAME ADDRDSU ADPATCH
      VER   45      00
      REP   45      FF
```


Changing creation date default settings during 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 DS1CREDIT field.

DFSMSDss provides four patch bytes that allow you to change the method of setting the DS1CREDIT field. These patch bytes apply only to data sets that are allocated by DFSMSDss (not preallocated 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 (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
NAME ADDRDSU ADRPATCH
VER 46 00
REP 46 FF
VER 48 00
REP 48 FF
```

Note:

1. As an alternative, you can use the SET PATCH command to set the patch bytes dynamically
2. For a preallocated target that has F8/F9 DSCBs, the F9 DSCB field DS9CREAT is set ON, and is being scratched and reallocated, these patch byte settings will not be honored. These types of data sets have a field in the F9 DSCB which corresponds to the number of milliseconds past midnight in which the data set was created (DS9TIME). In order for the DS9TIME to be usable, the creation date and the time past midnight of the preallocated target will be preserved as its value prior to scratching.

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 sample JCL (see [“Sample JCL” on page 213](#)) as shown below:

```
//SYSIN DD *
NAME ADDRSSU ADPATCH
VER 4B 00
REP 4B FF
```

Changing the default maximum number of active parallel subtasks

During processing in PARALLEL mode, DFSMSdss limits the number of active subtasks to avoid exhausting virtual storage in the DFSMSdss address space. When the number of active parallel subtasks reaches the DFSMSdss determined value, DFSMSdss waits for any executing subtasks to complete before scheduling a new subtask.

This function is affected by setting the one byte field at offset X'4C' in ADPATCH. The settings are listed below:

X'00'

DFSMSdss functions normally as described above. The maximum number of active parallel subtasks is determined by DFSMSdss.

Any setting from X'01' to X'FF'

Specifies the maximum number of active subtasks that DFSMSdss should allow to execute in parallel.

To override the default dynamically, you can use the SET PATCH command to set the patch byte at offset X'4C' to any value from X'01' through X'FF'. To override the default permanently (for example, to allow a maximum of X'32' active parallel subtasks), modify the sample JCL (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN DD *
NAME ADDRSSU ADPATCH
VER 4C 00
REP 4C 32
```

Changing the default initialization processing during DUMP with FCWITHDRAW (OA18929)

For a DUMP FULL or DUMP TRACKS operation, DFSMSdss invokes ICKDSF to initialize the source volume of the DUMP operation at the end of the dump processing when all of the following conditions are met:

- FCWITHDRAW is specified
- The VTOC tracks on the source volume of the DUMP operation are the target of a FlashCopy relationship or the volume is dump conditioned
- The TRACKS keyword, if specified, designates one extent range that represents the entire volume
- The volume is not a VM-format volume (CP volume)
- The volume supports data set FlashCopy or space efficient FlashCopy.

You can control this function by setting the flag at offset X'4D' in ADPATCH. The valid settings are:

X'00'

DFSMSdss functions as described in this topic.

Any setting other than X'00'

DFSMSdss does not initialize the source volume when a DUMP FCWITHDRAW operation completes.

To set the flag on dynamically, use the SET PATCH command to set the patch byte at offset X'4D' to a value of X'FF'. To set the flag to on permanently, modify the sample JCL (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN DD *
NAME ADDRSSU ADPATCH
```

VER	4D	00
REP	4D	FF

Note: For a space efficient volume, DFSMSDss issues an informational message when all of the following conditions are true:

- This patch byte is set
- The volume is the target of a FlashCopy relationship
- FCWITHDRAW is specified for the FlashCopy relationship.

The message indicates that the physical space on the space efficient volume remains allocated. To release the physical space, you must initialize the volume through the ICKDSF INIT command.

By default, for a DUMP FULL or DUMP TRACKS operation, DFSMSDss will not perform an FCWITHDRAW when the volume could not be initialized using ICKDSF and issues an ADR288W. A patch option is being provided to allow DFSMSDss to continue issuing the FCWITHDRAW to the source volume of a DUMP FULL/TRACKS operation and prevents the ADR288W from being issued.

You can control this function by setting the flag at offset X'5E' in ADPATCH. The valid settings are:

X'00'

DFSMSDss functions as described in this topic.

Any setting other than X'00'

DFSMSDss performs an FCWITHDRAW when DFSMSDss was unable to initialize the source volume when a DUMP FCWITHDRAW operation completes.

To set the flag on dynamically, use the SET PATCH command to set the patch byte at offset X'5E' to a value of X'FF'. To set the flag to on permanently, modify the sample JCL (see [“Sample JCL”](#) on page 213) as follows:

```
//SYSIN DD *
NAME ADDRDSU ADPATCH
VER 5E 00
REP 5E FF
```

For more information about FCWITHDRAW and volume initialization processing, see [“FCWITHDRAW”](#) on page 407.

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 SET PATCH 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)

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 ADPATCH. The settings are:

X'00'

If the FASTREPLICATION keyword is not specified, then DFSMSdss processes as if the FASTREPLICATION(PREFERRED) keyword was specified.

Any setting other than X'00'

If the FASTREPLICATION keyword is not specified, 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 (see [“Sample JCL”](#) on page 213) as follows:

```
//SYSIN DD *
      NAME  ADDRDSU  ADPATCH
      VER   4F      00
      REP   4F      FF
```

Tuning hardware assisted compression (OA13300)

You can 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 the threshold value. To change the number of poor compressions allowed before rebuilding the compression, you can modify the value at offset X'50' using the SET PATCH command. The valid 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.

To modify the target compression threshold, you can set the value at offset X'51', using the SET PATCH 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.

Resetting the data-set-changed indicator during physical full or partial RESTORE operation (OA20907)

The data-set-changed indicator (DS1DSCHA) bit is automatically turned off during a physical full volume or tracks RESTORE operation. You can reset the indicator by setting the flag at offset X'52' in ADRPATCH. The valid settings are:

X'00'

DFSMSDss turns off the DS1DSCHA bit during a physical full volume or tracks RESTORE operation.

Any setting other than X'00'

DFSMSDss does not reset the DS1DSCHA indicator during a physical full volume or tracks RESTORE operation.

To set the flag on dynamically, use the SET PATCH command. To set the flag on permanently, modify the sample JCL (see [“Sample JCL”](#) on page 213) as follows:

```
//SYSIN DD *
      NAME  ADDRDSU  ADRPATCH
      VER   52       00
      REP   52       FF
```

Note: If you use an application program interface (API) to invoke DFSMSDss, this patch is not used and the DS1DSCHA bit is reset as usual.

Requesting that DFSMSDss double-check data set high used RBA values for LDS data sets

You can use the patch byte at offset X'53' to request that DFSMSDss is to do a sequential scan of the VSAM volume data set to re-find the VSAM volume record for a linear data set, and compare its Data Set High Used RBA value with the one obtained earlier in DFSMSDss processing. The valid settings are:

X'00'

DFSMSDss does not try to re-obtain the VSAM record for a linear data set during a logical data set COPY.

Any setting other than X'00'

DFSMSdss tries to re-obtain the VSAM volume record for a linear data set during a logical data set COPY and then compares the values of the Data Set High Used RBA. If they are not the same, DFSMSdss issues an ADR432E message along with other diagnostic information. Additionally, DFSMSdss issues ADR898D message to the system that requests a PRINT of VVDS on the specified volume. Processing of any data sets that receive the ADR432E message fails. DFSMSdss does not delete these data sets even if you specify the DELETE keyword.

To set the flag on dynamically, use the SET PATCH command. To set the flag on permanently, modify the sample JCL (see “Sample JCL” on page 213) as follows:

```
//SYSIN DD *
      NAME  ADRDSSU  ADRPATCH
      VER    53      00
      REP    53      FF
```

Note: If you use this Patch byte, DFSMSdss processing might take a long time, especially when the VSAM volume data set is large, because DFSMSdss might scan the entire VSAM volume data set to find the VSAM volume record.

Enabling or disabling use of the catalog search interface for data set name filtering

You can use the patch byte at offset X'54' to enable or disable the use of the Catalog Search Interface (CSI) during DFSMSdss catalog filtering for logical data set COPY, DUMP or RELEASE commands. DFSMSdss catalog filtering is performed whenever no input volumes are specified on the COPY, DUMP or RELEASE commands. Catalog filtering can use either:

- Catalog search interface (CSI)
- Generic catalog locates.

DFSMSdss provides a patch byte to enable or disable the use of the CSI during DFSMSdss catalog filtering. Modify the value at offset X'54' using the SET PATCH command. The valid settings are as follows.

X'11'

DFSMSdss uses the CSI to convert generic filter criteria to a list of data sets that are cataloged.

Any other value

DFSMSdss uses generic catalog locates to generate a list of data sets that are cataloged.

Requesting that DFSMSdss restore the VM-formatted volume that was DUMPed by z/OS V1R10 before OA27531 was applied.

You can use the patch byte at offset X'55' to request that DFSMSdss restore the VM formatted volume that was DUMP on a z/OS V1R10 system before OA27531 was applied.

Note that this patch byte is only valid for RESTORE tracks operations, and the patch byte is honored only when the SET PATCH command sets it on, dynamically. You must also ensure that the DUMP is a VM formatted volume made with the CPVOLUME keyword specified.

The settings are:

X'00'

DFSMSdss functions normally.

Any setting other than X'00'

The RESTORE will treat the contents of the DUMP as a VM formatted volume.

Adding timestamps to messages

You can use the patch byte at offset X'58' to request that DFSMSdss add timestamps to certain message classes as follows:

X'00'

Specific messages

X'80'

Informational messages

X'40'

Warning messages

X'20'

Error messages

X'10'

Terminating messages

The values may be added together to get combinations of messages. For example, if you want timestamps on warning and error messages, set the value to X'60'.

Enabling building appropriate channel programs

DFSMSdss tape read channel programs are built to read the maximum possible amount of data that any version of DFSMSdss can create. DFSMSdss provides a patch byte to modify how DFSMSdss builds channel programs to read tape blocks. You can control this function by setting the flag at offset X'59' in ADPATCH. The valid settings are:

X'00'

DFSMSdss continues to build channel programs to read the maximum amount of data that can be created by DFSMSdss.

Any other value

DFSMSdss builds channel programs to read the maximum block size written to the dump data set that the restore is being performed on.

To set the flag on dynamically, use the SET PATCH command to set the patch byte at offset X'59' to a value of X'FF'. To set the flag on permanently, modify the sample JCL (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN DD *
NAME ADRDSSU ADPATCH
VER 59 00
REP 59 FF
```

Requesting that DFSMSdss attempt to fix ESDSes with corrupted RDFs

Use the flag at offset X'57' to request that DFSMSdss attempt to fix entry sequenced data sets (ESDSes) that have record descriptor fields (RDFs) that incorrectly indicate repeated records. The values are:

X'00'

DFSMSdss operates normally

Any other value

DFSMSdss attempts to fix the ESDSes during logical data set RESTORE processing when VSAM I/O is needed to restore the data set. DFSMSdss examines each RDF when using VSAM I/O to put records into an ESDS. If the RDF indicates incorrectly that it describes repeated records, it is modified to describe a single record. Message ADR925W is issued to indicate that modifications were done during the restore of the data set. The ADR925W message is also written to the console for use by automation.

To set the flag on dynamically, use the SET PATCH command to set the flag at offset X'57' to a value of X'FF'. To set the flag permanently, modify the sample JCL (see [“Sample JCL” on page 213](#)) as follows:

```
//SYSIN DD *
NAME ADRDSSU ADPATCH
```


VER	57	00
REP	57	FF

ADRPTCHB data area

Table 21 on page 236 lists the mapping of flags in ADRPTCHB.

Table 21. ADRPTCHB-Mapping Macro					
Offsets		Type	Len	Name (Dim)	Description
Dec	Hex				
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	VARIABLE USED FOR ADTL INFO XT@NA03602 IN ADRTDSC & IN ADRFDSRL
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 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
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

Table 21. ADRPTCHB-Mapping Macro (continued)

Offsets		Type	Len	Name (Dim)	Description
Dec	Hex				
Note: The following 6 fields starting at offset X'20' are intended for installations to "adjust" target PDSE DS1SCAL3 that may have been incorrectly altered by PE-OW04199. The logic is: IF 2nd_qty > PBCYLHI THEN 2nd_qty = PBCYLQTY. Any nonzero value in PBCYLHI, PBTRKHI, or PBBLKHI activates 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.		PBVCCUM	VOLCOUNT(N(NN))
		...1		PBVCCANY	VOLCOUNT(ANY)
	 1111	*		UNUSED
65	(41)	UNSIGNED	1	PBVCCVAL	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 DEFRACTION TO MOVE CHECKPOINT INDICATED DATA SET
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

Table 21. ADRPTCHB-Mapping Macro (continued)

Offsets		Type	Len	Name (Dim)	Description
Dec	Hex				
72	(48)	UNSIGNED	1	PBCOCRED	DS1CREDIT COPY OLD 0 = USE OLD DATE 1 = USE TODAY'S DATE
73	(49)	UNSIGNED	1	PBCNCRED	DS1CREDIT COPY NEW 1 = USE OLD DATE 0 = USE TODAY'S DATE
74	(4A)	UNSIGNED	1	*	RESERVED
75	(4B)	UNSIGNED	1	PBVPDSE	VALIDATE PDSE 1 = ENABLE VALIDATION 0 = DISABLE VALIDATION
76	(4C)	UNSIGNED	1	PBMXPTSK	MAXIMUM NUMBER OF ACTIVE PARALLEL SUBTASKS X'01' THROUGH X'FF' = USER SPECIFIED THRESHOLD VALUE X'00' = DFSMSDSS DETERMINED THRESHOLD
77	(4D)	UNSIGNED	1	PBNOINIT	CHANGE DEFAULT INIT PROCESSING DURING DUMP FCWITHDRAW 1 = DO NOT INIT VOLUME 0 = INIT DUMP SOURCE VOLUME WHEN APPLICABLE
78	(4E)	UNSIGNED	1	PBMOVLL	OVERRIDE DEFRAG TO MOVE LINKLIST-INDICATED DATA SET
79	(4F)	UNSIGNED	1	PBFCDEF	OVERRIDE FAST REPLICATION DEFAULT SETTING TO NONE
80	(50)	UNSIGNED	1	PBDCTBD	REBUILD COMPRESSION DICTIONARY AFTER THIS MANY BAD COMPRESSIONS
81	(51)	UNSIGNED	1	PBCMPH	POOR COMPRESSION THRESHOLD VALUE 1-100
82	(52)	UNSIGNED	1	PBDSCHA	OVERRIDE RESET OF DSCHA BIT ON FULL OR PART RESTORE 0 = DS1DSCHA RESET 1 = DSCHA NOT RESET
83	(53)	UNSIGNED	1	PBCHKVVR	DOUBLE CHECK THE DSHURBA VALUE FOR LDS DATA SETS
84	(54)	UNSIGNED	1	PBCSIEN	ENABLE USAGE OF THE CATALOG SEARCH INTERFACE FOR GENERIC FILTER
85	(55)	UNSIGNED	1	PBCPVOL	1 = TREAT RESTORE AS A CPVOL - ONLY BY THE SET PATCH CMD
86	(56)	UNSIGNED	1	PBRESDS	TREAT ESDS AS LDS WHEN ATTEMPTING LOGICAL RESTORE OF INVALID DB2 ESDS
87	(57)	UNSIGNED	1	PBMSGTIME	ADD TIMESTAMPS TO MESSAGES IN EACH OF THE VARIOUS CLASSES
88	(58)	UNSIGNED	3	*	Reserved
91	(5B)	UNSIGNED	1	PBEATRV	Set EATTR=OPT for non-VSAM allocations
92	(5C)	UNSIGNED	1	PBZCFAIL	SET for ZCOMPRESS RESTORE ERROR
93	(5D)	UNSIGNED	1	PBCLWDCNTR	When creating a backup to an object storage cloud allow existing objects to be overwritten.
94	(5E)	UNSIGNED	1	PBBYFCWDEN	FCWD ENHANCEMENT BYPASS
95	(5F)	UNSIGNED	1	PBCLONERESE T	ALLOW RESET WITH CLONE
96	(60)	UNSIGNED	1	PBVTOCERR	Info message on VTOC error
97	(61)	UNSIGNED	1	PBSGWARN	Omit warning message for unavailable SG volumes

Table 21. ADRPTCHB-Mapping Macro (continued)

Offsets		Type	Len	Name (Dim)	Description
Dec	Hex				
98	(62)	UNSIGNED	1	PBCLWDPROV	Check for CDA provider file regardless of CDAPROVIDERFILE keyword's presence
99	(63)	UNSIGNED	1	PBOMITWARN	Omit specified warning message.

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	
PBCLWDCNTR	5D	
PBCMPTH	51	
PBCNCRED	49	
PBCNREFD	3B	
PBCOCRED	48	
PBCOREFD	3A	
PBCYLHI	20	
PBCYLQTY	24	
PBDCTBD	50	
PBDSCHA	52	
PBDUPKEY	9	
PBEOFNO	1D	
PBEX2000	1C	
PBFCDEF	4F	
PBFUDGE	44	
PBINVTOK	3E	
PBMISCNT	1A	
PBMOVLL	4E	
PBMSCIOK	42	
PBMXPTSK	4C	
PBNACSAC	3D	
PBNACSAU	16	

Name	Hex Offset	Hex Value
PBNMVNCV	14	
PBNOINIT	4D	
PBNORLS	45	
PBNO482E	19	
PBNRACFI	13	
PBNSLECT	17	
PBRESERV	10	
PBRESV60	43	
PBRETRY#	F	
PBREVOKE	11	
PBRNCRED	47	
PBRNREFD	39	
PBROCREC	46	
PBROREFD	38	
PBSRCVOL	3F	
PBTIMOUT	A	
PBTRKHI	28	
PBTRKQTY	2C	
PBUVSPRE	8	
PBVCVAL	41	
PBVDPDSE	4B	
PBVOLOPT	40	
PBWAIT#	E	
PBWAITFG	D	
PBWFUMOD	4A	
PBX99365	1B	
PTCHARRY	8	
PTCHEYE	0	

Overriding the EATTR attribute during logical COPY

During logical data set COPY operations, DFSMSdss uses the EATTR attribute of the source data set to determine the EATTR attribute of the target. If the source has an EATTR value of NS or NO, then the target data set that DFSMSdss allocates during the COPY will result in propagating the same EATTR attribute.

DFSMSdss provides a patch byte to allow you to override the EATTR attribute of the target data set to EATTR=OPT. 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'5B' using the SET PATCH command. The settings are:

X'00'

DFSMSdss functions normally. Data sets allocated during logical data set COPY are allocated with the same EATTR attribute as the source.

Any setting other than X'00'

Logical data set COPY command processing allocates the target data set with EATTR=OPT when the target data set is not preallocated or scratched and reallocated when the source EATTR attribute is set to NS or NO.

Recommendation: Restrict the use of this capability with the following RACF resources in the FACILITY class.

Table 22. RACF Resources to Restrict Use of This Patch Byte

Function	Resource	Required Access
Who can dynamically set patch bytes with the SET PATCH command. The patch byte is honored only when the SET PATCH command sets it on dynamically.	STGADMIN.ADR.PATCH	READ
Use of the COPY command.	STGADMIN.ADR.COPY	READ

Overriding the ADR310W warning message during full volume DUMP

During full volume DUMP operations and if there is a problem reading the source VTOC, ADR310W is issued and the overall return code is set to 4.

This might be expected when a backup is performed of a volume that is being replicated with Global Mirror or a similar offering.

DFSMSdss is providing this patch to allow you to override the warning message that causes the overall return code to be set to 4. When the patch is set, DFSMSdss instead issues ADR529I. This function is affected by setting the flag at offset X'60' in ADRPTCHB or with the SET PATCH command. The settings are:

X'00'

DFSMSdss functions normally.

Any setting other than X'00'

When DFSMSdss detects an issue reading the VTOC during full volume DUMP, it issues ADR529I instead of ADR310W. The overall return code is not changed.

Restore an extended-format data set that was dumped with the ZCOMPRESS keyword when zEDC services were not used

You can use the flag byte at offset X'5C' to restore an extended-format data set that was dumped with the ZCOMPRESS keyword when zEDC services were not used during the dump, and APAR OA47257 was not applied. Use this patch only if, at RESTORE time, the data set received error messages ADR908E and ADR910E RETURN CODE = 30000004, REASON CODE = 00000023. This patch byte is honored only when the SET PATCH command sets it to on dynamically when DFSMSdss is executed in batch. If DFSMSdss is called by an application, it honors the setting in ADRPATCH. The settings are:

X'00'

DFSMSdss functions normally

Any setting other than X'00'

Logical data set RESTORE of an extended-format data set that received the ADR908E and ADR910E RETURN CODE = 30000004, REASON CODE = 00000023 when dumped with a ZCOMPRESS failure restores properly.

Recommendation: This patch byte should only be used for RESTORE of single data sets that experienced the problem (error messages ADR908E and ADR910E RETURN CODE = 30000004, REASON CODE = 00000023). If used against properly dumped data sets, the RESTORE gives unexpected results.

Omitting the ADR826W warning message when the STORGRP keyword is specified

When the STORGRP keyword is used, if there are unavailable volumes defined to the storage group(s) specified, ADR826W is issued and the overall return code is set to 4. DFSMSdss is providing this patch to allow the user to override the warning message that causes the overall return code to be set to 4. When the patch is set, DFSMSdss will instead omit the ADR826W message. This function is affected by setting the flag at offset X'61' in ADRPTCHB or with the SET PATCH command. The settings are:

X'00'

DFSMSdss functions normally.

Any setting other than X'00'

When DFSMSdss detects an unavailable volume for a specified storage group, it will omit issuing ADR826W.

Check for a CDA Provider file regardless of the CDAPROVIDERFILE keyword's presence

The CDAPROVIDERFILE keyword must be specified in order for DFSMSdss to check for a CDA Provider file. A patch byte can be specified so that DFSMSdss will default to always check for a CDA provider file regardless of whether the CDAPROVIDERFILE keyword was specified or not.

The function is affected by setting the flag at offset X'62' in ADPATCH. The settings are as follows:

X'00'

DFSMSdss functions normally. DFSMSdss will only look for a CDA provider file when the CDAPROVIDERFILE keyword is specified.

Any setting other than X'00'

DFSMSdss will always check for a CDA provider file regardless of the CDAPROVIDERFILE keyword's presence.

To set the flag to on dynamically, use the SET PATCH command.

Omit specified warning message

When this patch byte is set one or more specific warning messages will be omitted.

DFSMSdss is providing this patch to allow the user to override specific warning messages that cause the overall condition code to be set to 4. When the patch is set, DFSMSdss will instead omit the warning message. This function is affected by setting the flag at offset X'63' in ADRPTCHB or with the SET PATCH command. The settings are:

X'00'

DFSMSdss functions normally.

X'80'

'80' bit is set.

When DFSMSdss deems surfacing ADR556W necessary for any reason code, the ADR556W message will be omitted.

Part 2. DFSMSdss Storage Administration Reference

Chapter 15. Specifying DFSMSdss commands

This topic is divided into the following sections:

- “[Command syntax](#)” on [page 245](#) describes syntax requirements.
- “[How many subkeywords are allowed?](#)” on [page 246](#) explains the allowable number of subkeywords that you can specify in the input (command) stream.
- “[Specifying subkeywords in a command data set](#)” on [page 246](#) explains how to use keywords in a command data set.
- “[How to read syntax diagrams](#)” on [page 246](#) explains the syntax conventions used in this book.
- “[JCL that you need](#)” on [page 248](#) summarizes the job control language (JCL) that you might need to use DFSMSdss.

Command syntax

You can write DFSMSdss commands in free form in columns 2 through 72 (inclusive). Any character in column 1 or beyond column 72 is ignored, causing unpredictable results when the task is processed. Syntax requirements are:

Commands:

A command must appear first, followed by its keywords. Each command must take up only one line, unless a continuation character is used to indicate continuation of the command on the next line. A command is separated from its keywords by one or more blanks, a comment, or both. For example:

```
DUMP FULL INDD(DASD1) OUTDD(TAPE1)
or
DUMP FULL INDD(DASD1) -
OUTDD(TAPE1)
```

Comments:

A comment is a string of characters that begins with a `/*` and ends with an `*/`. For example:

```
/*THIS IS A COMMENT */
```

A comment that does not begin and end on the same line must contain a continuation character, or syntax errors will result. For example:

```
/* THIS IS A MULTI -
   LINE COMMENT */
```

Separators:

A separator can be a comma (,), one or more blanks, or a comment. Separators shown in the syntax diagrams in this manual are always commas, but any of the three types can be used.

Keywords:

Keywords are parameters separated by one or more separators.

Subkeywords:

Subkeywords follow their associated keyword and are separated from them by a pair of enclosing parentheses. One or more blanks can precede and follow each parenthesis in the pair. For example:

```
REBLOCK( DATASET1 )
or
REBLOCK(DATASET1)
```

If two or more subkeywords are permissible for a single keyword, they are separated from one another by one or more blanks or by commas. Each comma can be preceded and followed by one or more blanks. For example:

```
REBLOCK(DATASET1 , DATASET2)
      or
REBLOCK(DATASET1,DATASET2)
      or
REBLOCK(DATASET1 DATASET2)
```

Continuation:

Continuation of a command is specified by a hyphen (-) as the right-most nonblank character, preceded by one or more blanks. If a continuation character is used, the following line is read as if it were part of the previous line. Since only one command is allowed per line, no additional commands may be included on the continued line. If no continuation character is used, the first word on the following line must be a command. For example:

```
COPY DATASET (INCLUDE(DATASET1)) ALLDATA(*) -
      CATALOG REBLOCK(DATASET1)
```

For examples of the continuation usage, see [“Continuation rules for IF-THEN-ELSE command sequencing” on page 512](#), using the IF-THEN-ELSE command sequence.

The absence of such a hyphen indicates the end of the command. If a keyword or subkeyword cannot fit on the remainder of a line, it can be started on that line, followed immediately by a plus sign (+) in column 72, and continued on the next line.

End of a command:

The end of a command can be specified by a semicolon (;). Everything to the right of the semicolon is ignored.

How many subkeywords are allowed?

DFSMSdss allows most command keywords to have a maximum of 255 subkeywords specified within the inline command stream. Exceptions to this rule are noted within individual command descriptions.

Specifying subkeywords in a command data set

Rather than specifying the data set names in the inline (command) stream, you can instead specify the ddname of a sequential data set or a member of a partitioned data set that contains the list of data set names. This allows you to specify more than 255 data set names.

The following keywords allow this ddname specification:

DATASET
(FILTERDD(ddn))

EXCLUDE
(DDNAME(ddn))

FILTERDD
(ddn)

PASSWORD
(ddn)

How to read syntax diagrams

To read syntax diagrams, follow one line at a time from the beginning to the end, and code everything you encounter on that line.

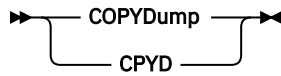
The following conventions apply to all syntax diagrams for DFSMSdss commands:

- Read the syntax diagrams from left to right and top to bottom.
- Each syntax diagram begins with a double arrowhead (➤➤) and ends with opposing arrows (➤➤).
- An arrow (➤➤) at the end of a line indicates that the syntax continues on the next line. A continuation line begins with an arrow (➤➤).

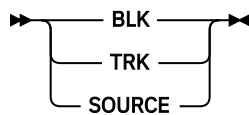
- Commands and keywords are shown in uppercase and lowercase letters. The uppercase portion is the minimum needed to code the command properly; the lowercase portion is optional. For example, COPYDump can be coded in any of the following ways: COPYD, COPYDU, COPYDUM, or COPYDUMP.

Note: Commands *must* be entered in uppercase. Lowercase is not recognized.

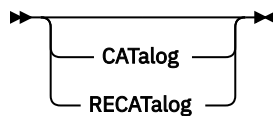
- Some commands and keywords have alternative abbreviations; these appear as part of the stack for that command or keyword. For example, the alternative abbreviation for COPYDump is CPYD.



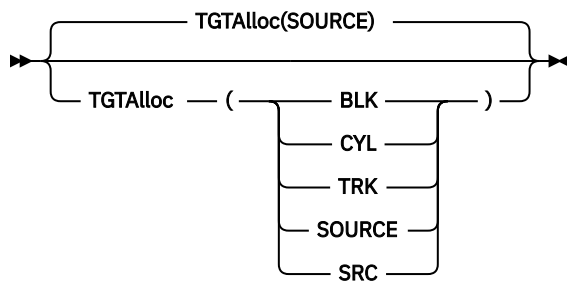
- Words in all lowercase letters represent information you supply. For example, *volser* or *ddn*.
- You must provide all items enclosed in parentheses, (), and you must include the parentheses.
- Where you can choose from two or more keywords, the choices are stacked one above the other. If one choice within the stack lies on the main path, you must choose a keyword. In the following example you must choose either BLK, TRK, or SOURCE.



- If one or more keywords are below the main path, they are optional. In the following example CATalog and RECATalog are optional keywords. You can choose one, or the other, or none.



- If a stack of keywords are below the main path and one keyword is above the main path, the use of the keyword is optional, and the above item is the default. In the following example, if no keywords are specified, the default TGTAlloc (SOURCE) is taken.

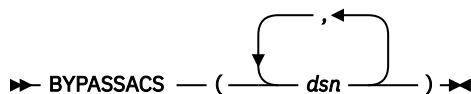


- The repeat symbol is shown below:



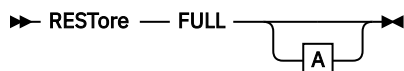
The repeat symbol appearing above keywords and variables indicates that you can specify those keywords and variables more than once. If a comma appears in the repeat symbol, you must separate repeated keywords or variables with a comma or any valid separator.

For example, after the keyword BYPASSACS, you can enter multiple data set names separated by commas.

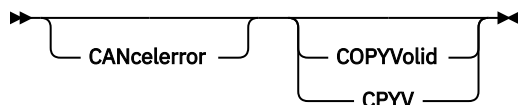


- Substitution blocks are used to simplify the diagrams. They indicate that blocks of the syntax diagram are located outside of the main diagram. You insert the keywords for that block where the symbol appears, and return to the main diagram to continue with the command.

In the following example the substitution block, **A**, points to a block of the syntax diagram that immediately follows the **FULL** keyword.



A: Optional Keywords Used With FULL:



The above example is equivalent to the following:



JCL that you need

The examples in this manual are shown with the necessary JCL. You may need the following JCL to use DFSMSdss:

Statement

Usage

JOB (required)

Initiates your job.

EXEC (required)

Specifies the program name (PGM=ADRDSSU) or, if the job control statements reside in a procedure library, the procedure name. See [“How to control DFSMSdss through PARM information in the EXEC statement”](#) on page 249 for additional information that can be entered in the PARM parameter of the EXEC statement.

SYSPRINT DD (required)

Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct-access device. If the DCB keyword LRECL is specified on the DD statement, it must be from 84 to 137 inclusive. If the BLKSIZE keyword is specified, it must be at least four greater than LRECL. If LRECL is less than 84, the return code is 8, and an error message is issued. If the specified LRECL is greater than 137, the LRECL and BLKSIZE are set to 137 and 141, respectively.

Note: If the SYSPRINT DD or a temporary message data set resides on a volume that is being processed by DFSMSdss and a secondary allocation is needed for it, the job may result in an S138 abend. This is because the DADSM EXTEND function attempts to enqueue on the volume's VTOC while DFSMSdss holds the enqueue on that VTOC. To avoid this situation, define the SYSPRINT DD to another volume or use the WORKUNIT or WORKVOL parameters, or both.

SYSIN DD (required)

Defines a command data set containing your DFSMSdss commands. It usually resides in the input stream. However, it can be defined as a blocked or unblocked sequential data set or as a member of a partitioned data set. Records must be fixed format, LRECL=80.

input DD (optional)

Defines the input (also called the source). The ddname, *input*, is supplied by you and is referred to by your DFSMSdss commands. This DD statement is not required for some operations. Do not specify the BUFNO keyword.

The following example shows an input DD statement that specifies a DASD volume:

```
//DASD DD UNIT=3380,VOLUME=(PRIVATE,SER=111111),DISP=OLD
```

See the reference under the INDDNAME, INDYNAM, DDNAME and DYNAM keywords of the various commands (for example, COPY or DUMP) for additional information.

output DD (optional)

Defines the output (also called the target). The ddname, *output*, is supplied by you and is referred to by your DFSMSdss commands. This DD statement is not required for some DFSMSdss operations. Do not specify the BUFNO keyword. Code a volume count when a new data set will reside on six or more volumes.

Note:

1. DISP=MOD is not supported for an output DD statement.
2. For dump operations, the output DD statement describes a sequential data set that DFSMSdss dumps the data to. If this dump data set resides on a DASD volume that is being processed by DFSMSdss and a secondary allocation is needed for it, the job may result in a S138 abend. This is because the DADSM EXTEND function attempts to enqueue on the volume's VTOC while DFSMSdss holds the enqueue on that VTOC.
3. For dump operations, the output DD statement can be directed to DUMMY. If you specify DD DUMMY, DFSMSdss does not perform data sets I/O. If you specify DELETE keyword with DD DUMMY, DFSMSdss deletes the input data sets as requested.
4. If multiple dumps are done in the same step, each dump command should have an output DD statement that refers to a unique JCL DD statement.

The following example shows an output DD statement that specifies a tape volume:

```
//TAPE DD UNIT=3480,VOLUME=SER=TAPE01,LABEL=(1,SL),  
// DISP=(NEW,CATLG),DSNAME=USER2.DUMP
```

See the reference under the OUTDDNAME and OUTDYNAM keywords of the various commands (for example, COPY or DUMP) for additional information.

filter DD (optional)

Defines a data set consisting of cards or card-image records that contains the filtering criteria (INCLUDE, EXCLUDE, and BY) to be used in a data set command. The ddname, *filter*, is supplied by you and is referred to by your DFSMSdss commands.

password DD (optional)

Defines a data set consisting of card-image records that contains data set names and their passwords. The ddname, *password*, is supplied by you and is referred to by your DFSMSdss commands.

For more information on coding JCL, refer to [z/OS MVS JCL Reference](#).

How to control DFSMSdss through PARM information in the EXEC statement

The EXEC statement for DFSMSdss can contain PARM information that is used by the program. You can use the following keyword parameters:

ABEND=nnn

nnn is a 3-digit decimal message number (ADR*nnnx*). If this is specified and this message is to be issued, DFSMSdss performs a user 0001 ABEND dump after issuing the message, the task stops and the return code is set to 8. To get a dump, include a DD statement for SYSABEND, SYSMDUMP, or SYSUDUMP. This keyword is provided for diagnostic purposes only.

AMSGCNT=nnnn

The abend message occurrence count that tells DFSMSdss to end abnormally on the *nth* occurrence of the message specified in ABEND=*nnn*. *nnnn* for AMSGCNT can be a number between 1 and 9999. The default is 1 (first occurrence). This keyword is provided for diagnostic purposes only.

DEBUG=FRMSG

This parameter causes DFSMSdss to issue an informational message during copy or defrag operations for which fast replication methods, such as SnapShot and FlashCopy, cannot be used. The informational message indicates why DFSMSdss could not use fast replication. This keyword is provided for diagnostic purposes only.

Guidelines:

1. When you specify this parameter, DFSMSdss interprets it as though you have specified the `DEBUG(FRMSG(SUMMARIZED))` keyword.
2. You can use the `DEBUG(FRMSG(MIN | SUM | DTL))` keyword with the `COPY` and `DEFRAG` commands. It is used when designating the use of fast replication methods. It is recommended that you convert your `DEBUG=FRMSG` parameter to this new keyword specification in your JCL jobs.

ESEPREAL=YES|NO

Specifies whether DFSMSdss preallocates the target extent space during a FlashCopy establish on thin-provisioned (ESE) volumes during `COPY` with `DELETE`, `DEFRAG`, or `CONSOLIDATE` operations. The default is `YES`. If `NO` is specified, the user must verify that sufficient space is available in the extent pool to allow the operation to complete.

LINECNT=nnnn

nnnn is a 1-digit to 4-digit number indicating the number of lines to be printed per page for the `SYSPRINT` data set. The default is 60. Specify `LINECNT=9999` to prevent a page ejection.

PAGENO=nnnn

nnnn is a 1-digit to 4-digit number indicating the starting page number to be used for the `SYSPRINT` data set. The default is 1.

SDUMP=nnn

nnn is a 3-digit decimal message number (`ADRnnnx`). If this is specified, DFSMSdss requests an SVC dump after issuing the message, and the task continues processing. The SVC dump is directed to a system defined data set (`SYS1.DUMPnn`) for later analysis. This keyword is provided for diagnostic purposes only.

SIZE=nnnnK

nnnn is a 1-digit to 4-digit number indicating the number of KB (1 KB equals 1024 bytes) of main storage to be used by DFSMSdss. This amount must be less than or equal to that specified by the `REGION` keyword. The default value is the region size.

SMSGCNT=nnnn

The sdump message occurrence count that tells DFSMSdss to request an SVC DUMP on the *n*th occurrence of the message specified in `SDUMP=nnnn`. *nnnn* for `SMSGCNT` can be a number between 1 and 9999. The default is 1 (first occurrence). This keyword is provided for diagnostic purposes only.

TMPMSGDS=YES|NO

This parameter indicates whether or not a temporary `SYSPRINT` message data set is to be allocated. When `NO`, `SYSPRINT` messages are buffered in an ESA Hiperspace and performance is improved. The default is `NO`.

TRACE=YES

When used during `DEFRAG` or `CONSOLIDATE` operation, this prints messages that indicate the relocated extents. This is a diagnostic tool.

TYPRUN=NORUN

For copy, dump, restore, compress, and release operations, only input data set selection is done without actually processing data sets. Printed output for the run indicates the data sets selected. For a defragmentation operation, the initial volume statistics are printed without actually relocating any extents. For a `CONVERTV` operation, a full report is produced, but no volumes or data sets are actually converted. For `CGCREATED` operation, only control card syntax checking is done. The task is not processed.

TYPRUN=SCAN

Only control card syntax checking is done. No tasks are processed.

USEEXCP=YES|NO

Specifies whether the access method used by DFSMSDss for DUMP output, RESTORE input and COPYDUMP operations is to be EXCP. If the backup is to or from tape, the default is NO. If the backup is to or from DASD, the default is YES, unless the backup data set is in the extended format. For COPYDUMP and RESTORE operations that specify USEEXCP=YES, if the backup is from a non-label tape or a standard label tape with a block size of zero in its tape header, the dump data set block size must be specified on the input tape's DD in order for EXCP processing to be used. If the block size is not specified, BSAM processing is used instead.

UTILMSG=YES|NO|ERROR

This parameter controls the output of messages from auxiliary programs invoked by DFSMSDss (including ICKDSF, IDCAMS, IEBCOPY, and IEHMOVE) to the DFSMSDss SYSPRINT output. When YES, informational, warning, and error messages from these auxiliary programs are copied to the DFSMSDss SYSPRINT output. When NO, messages are not copied to the output. When ERROR, messages are copied only if the auxiliary return program returns an error code to DFSMSDss. The default is ERROR.

WORKUNIT=workunit

You can supply an esoteric DASD unit name (for example, SYSDA), a generic DASD unit name (for example, 3380), or a specific DASD address. DFSMSDss passes this unit to dynamic allocation when temporary data sets are allocated. When WORKUNIT is specified by itself, the volumes to be processed by DFSMSDss should not fall within the esoteric group passed as the WORKUNIT name. If the esoteric name applies to the volumes to be processed by DFSMSDss, specify WORKVOL.

WORKVOL=volser

You can supply a volume serial number on which DFSMSDss should allocate temporary data sets. DFSMSDss passes the volume serial number to dynamic allocation. The volume serial number passed as a WORKVOL should not be the same as any volume being processed by DFSMSDss under the current task.

Note:

1. WORKUNIT, WORKVOL, or both parameters can be specified when invoking DFSMSDss.
2. When DFSMSDss invokes the IDCAMS utility to copy an ICF user catalog, the export data set is allocated as a permanent data set. The permanent data set cannot be placed on the volume specified by WORKUNIT or WORKVOL.
3. An esoteric unit name that requests virtual I/O can be used in the WORKUNIT parameter, but when DFSMSDss invokes the IEHMOVE utility during data set copy, the default dynamic allocation unit name is used (SYSALLDA).

XABUFF=ABOVE16|BELOW16

The I/O buffers used by DFSMSDss for COPY, COPYDUMP, DEFRAG, DUMP, PRINT, and RESTORE operations are to be above or below 16 megabytes virtual storage. The default is ABOVE16.

ZBUFF64R=YES|NO

The I/O buffers that DFSMSDss uses for COPY, COPYDUMP, DEFRAG, DUMP, PRINT, and RESTORE operations can be backed by real storage that is anywhere in 64-bit addressing, or below the 2-gigabyte bar. The ZBUFF64R EXEC parameter allows a user to indicate to DFSMSDss where the I/O buffers should be located.

DFSMSDss obtains I/O buffers that are backed by real storage anywhere in 64-bit addressing when you specify EXEC PARM='ZBUFF64R=YES'. The default is ZBUFF64R=YES, which results in all I/O buffers being obtained such that they can be backed above the 2-gigabyte bar.

DFSMSDss determines whether a tape device used in a job supports buffers above the 2-gigabyte bar, and uses buffers above the 2-gigabyte bar when all tape devices in the job support buffers above the 2-gigabyte bar. If any tape device in a job does not support buffers above the 2 gigabyte bar, DFSMSDss uses buffers below the 2-gigabyte bar for the entire job.

Examples of PARM information are:

```
// EXEC PGM=ADRDSSU,
// PARM='PAGENO=8,LINECNT=57,SIZE=500K,UTILMSG=YES'

// EXEC PGM=ADRDSSU,
// PARM='TYPRUN=SCAN,DEBUG=FRMSG,XABUFF=ABOVE16,UTILMSG=YES'

// EXEC PGM=ADRDSSU,
// PARM='ZBUFF64R=NO,UTILMSG=YES'
```

Related reading: For additional information about the PARM parameter of an EXEC statement, see the [z/OS MVS JCL Reference](#).

Related reading: For additional information about the ADRUFO parameter list and the ADRUIXIT exit, see the [z/OS DFSMS Installation Exits](#).

Examples of invoking DFSMSdss with JCL

The following are some examples of JCL job streams for invoking DFSMSdss.

Note: Throughout the examples in this manual, a DASD is presented as UNIT=3380 or UNIT=3390, and a tape device as UNIT=3480. This is only for illustration; you can specify any supported DASD and any supported tape device. Refer to the appropriate system generation manual for the device-type notation to be used in the UNIT parameter of a DD statement.

Moving a data set

The following example shows how to move a data set from one DASD volume to another using JCL and a DFSMSdss command. The source data set is deleted and the target data set is cataloged to reflect its new location.

```
//MYJOB JOB accounting information,REGION=nnM
//STEP1 EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=A
//DASD1 DD UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//DASD2 DD UNIT=3390,VOL=(PRIVATE,SER=222222),DISP=OLD
//SYSIN DD *
COPY DATASET(INCLUDE(MYDATSET)) -
LOGINDDNAME(DASD1) OUTDDNAME(DASD2) DELETE CATALOG
/*
```

Dumping a data set

Do not use a data set name that DFSMSdss will reference during execution. Otherwise, enqueue contention with the operating system initiator will occur. This example shows how to back up a DASD data set to tape:

```
//MYJOB JOB accounting information,REGION=nnM
//STEP1 EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=A
//DASD DD UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//TAPE DD UNIT=3480,VOL=SER=TAPE01,
// LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=MYDATSET.BACKUP
//SYSIN DD *
DUMP DATASET(INCLUDE(MYDATSET)) -
INDDNAME(DASD) OUTDDNAME(TAPE)
/*
```

Restoring a data set

Do not use a data set name that DFSMSdss will reference during execution. Otherwise, enqueue contention with the operating system initiator will occur. In this example, the data set (MYDATSET) that has been backed up on tape in the "Dumping a Data Set" example is restored to the original DASD volume on which it resided.


```
//MYJOB    JOB    accounting information,REGION=nnM
//STEP1    EXEC   PGM=ADRDSSU
//SYSPRINT DD    SYSOUT=A
//TAPE     DD     UNIT=3480,VOL=SER=TAPE01,
// LABEL=(1,SL),DISP=(OLD,KEEP),DSNAME=MYDATSET.BACKUP
//DASD     DD     UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//SYSIN    DD     *
RESTORE DATASET(INCLUDE(MYDATSET)) -
        INDDNAME(TAPE) OUTDDNAME(DASD) REPLACE
/*
```


Chapter 16. DFSMSdss filtering—choosing the data sets you want processed

This section is divided into the following subsections:

- “How DFSMSdss filters data sets” on page 255 describes how DFSMSdss filters data sets by data set name and by data set characteristic.
- “Filtering by data set names” on page 256 explains how to specify fully and partially qualified data set names for filtering. This section includes rules for spelling out qualifiers and examples of qualified data set names.
- “Filtering by data set characteristics” on page 258 lists the data set characteristics on which filtering can be done.
- “Standard catalog search order” on page 263 identifies the standard order in which DFSMSdss searches for a data set name.

See Chapter 25, “Data set attributes,” on page 641 for information about DFSMSdss and the way that it determines individual data set attributes.

How DFSMSdss filters data sets

DFSMSdss filters data sets in the following order:

1. DFSMSdss will first tentatively select data sets by applying INCLUDE criteria. If you do not specify inclusion criteria, you must specify EXCLUDE or BY. In this case, DFSMSdss tentatively selects all data sets (equivalent to *INCLUDE({})*).
2. If you specify exclusion criteria (EXCLUDE), DFSMSdss surveys the data sets that are selected with the inclusion criteria then rejects the data sets that fit any of the new criteria.{}
3. Prior to filtering by data set characteristics (BY), DFSMSdss may serialize the data sets that resulted from previous filtering. For RLS data sets a quiesce is performed for serialization.
4. DFSMSdss will then apply any data set characteristics (BY) criteria to the remaining data set list. DFSMSdss selects only those data sets that satisfy all BY criteria.

DFSMSdss lets you find out which data sets are selected by the filtering process without actually performing the requested operations. You can do this by specifying TYPRUN=NORUN on the JCL EXEC PARM field.

Virtual storage access method (VSAM) data set considerations

Considerations for VSAM data sets include the following:

- INCLUDE and EXCLUDE filtering is performed on the cluster names of the data sets.
- If you specify the BY criterion DSORG, MULTI, CATLG, or FSIZE, filtering is done at the cluster level. One exception is the DEFRA command, that filters at the VSAM component level. If you specify other BY criteria, the data components of the selected clusters are further filtered on those BY criteria by using information from the volume table of contents (VTOC). If a data component is selected, the index component for the cluster is also selected. Again, the one exception is the DEFRA command, that requires index components to pass all BY criteria.
- DFSMSdss supports the BY criterion EXPDT as it exists in the VTOC only.
- For a physical data set RESTORE, to prevent index components from being restored inadvertently, you must specify the fully qualified name of the cluster.
- For data set print operation, the fully qualified name of the data set to be printed is required. Support is at the component, not the cluster, level.

Filtering by data set names

A *fully qualified* data set name is one in which all qualifiers are spelled out completely. A *partially qualified* data set name is one in which asterisks (*) or percent signs (%) are used to represent qualifiers or parts of qualifiers.

Using an asterisk in partially qualified data set names

The **single asterisk**, *, is used in place of exactly *one* qualifier. In addition, it can be used to indicate to DFSMSdss that only *part* of a qualifier has been specified. For example, just the first, last, middle, or first and last parts.

When used with other qualifiers, the **double asterisk**, **, indicates either the nonexistence of leading, trailing, or middle qualifiers, or the fact that they play no role in the selection process.

The rules for using asterisks in a qualifier are:

- Two asterisks are the maximum permissible in a qualifier.
- If there are two asterisks in a qualifier, they must be the first and last characters.

Consequently, the following are permissible qualifiers:

**
A

The following are **not** permissible qualifiers:

**A*
*A*B*
*A*B
A*B*C

Using a percent sign in partially qualified data set names

The percent sign, %, acts as a place holder for a single character during data set name filtering.

The rules for using % in a qualifier are:

- Each % corresponds to exactly one character.
- % can be specified more than once, consecutively or in any level of the qualifier.
- A % cannot match a null ('') or a period ('.').
- Use of a % in filtering does not change any of the other filtering specifications for data set names.

Consequently, specifying I%M.A.%AT%%ET matches the data set names IAM.A.DATASET and IBM.A.BATTYET, but not IAM.A.DATASE, IAM.AA.DATASET, IAM.A.ATASET, or IM.A.DATASET.

Examples of fully and partially qualified data set names

The following are examples of fully and partially qualified data set names.

Fully qualified data set names:

USER2LD

The first and only qualifier is *USER2LD*.

SYS1.UTIL3.LOAD

The first qualifier is *SYS1*, the second, *UTIL3*, and the third, *LOAD*.

USER2.PROGRAM1.LIST

The first qualifier is *USER2*, the second, *PROGRAM1*, and the third, *LIST*.

Partially qualified data set names using **:

**** .LOAD**

All data sets whose last, or only, qualifier is *LOAD*.

SYS1.**

All data sets whose first, or only, qualifier is *SYS1*.

USER2..LIST**

All data sets whose first and last qualifiers are *USER2* and *LIST*, respectively, including *USER2.LIST*.

More partially qualified data set names:

***.LOAD**

All data sets with two qualifiers whose last qualifier is *LOAD*.

SYS1.*

All data sets with two qualifiers whose first qualifier is *SYS1*.

SYS1.*.LOAD

All data sets with three qualifiers whose first and last qualifiers are *SYS1* and *LOAD*, respectively.

SYS1.UT*.LIST

All data sets with three qualifiers whose first and last qualifiers are *SYS1* and *LIST*, respectively, and whose second qualifier begins with *UT*.

****.*LIB**

All data sets whose last, or only, qualifier ends with *LIB*.

****.*LIB***

All data sets whose last, or only, qualifier has *LIB* in it.

.*PLI*.

All data sets with three qualifiers whose second qualifier has *PLI* in it.

***.*.P*M**

All data sets with three qualifiers whose last qualifier begins with *P* and ends with *M*.

All data sets.

%.LIST

All data sets with one character in the first qualifier and *LIST* in the last qualifier.

USER%.*

All data sets with two qualifiers whose first qualifier is *USER* followed by some other character.

%*%

All data sets whose single qualifier consists of two or more characters.

Note: Single quotation marks around a data set name indicates the name is fully qualified. For example, specifying 'USER.%' selects a data set whose first qualifier is *USER* and whose second qualifier is the character '%'. This data set is selected using the filter *USER.** because the wildcard (*) matches the character %.

Relative generation filtering

DFSMSdss allows filtering on relative generations of a generation data group (GDG) data set in the INCLUDE, EXCLUDE, and REBLOCK data set name lists. A GDG is specified as *dsn(n)* where *dsn* is a fully or partially qualified base name without the last qualifier (*GggggVvv*), and *n* is the relative generation number or * for all generations.

Guideline: The last qualifier consists of the generation number (*Ggggg*) and the version number (*Vvv*).

The following are examples of relative generation filtering:

dsn(0)

For the current generation

dsn(-x)

For the xth prior generation

dsn(+x)

For the xth future generation

dsn(*)

For all generations.

For logical operations using catalog filtering, you must use one of the following search criteria:

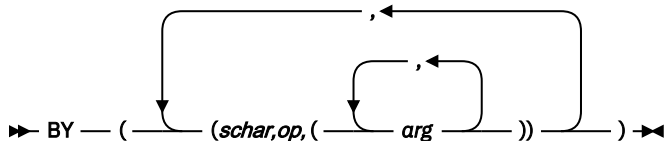
- The fully qualified data set name
- The partially qualified base name with the last qualifier (GggggVvv) not specified (for example, USER1.GDG.*)
- The above relative generation filtering

Partial qualification of the last qualifier (GggggVvv) is not supported unless a wildcard (* or %) is specified in the first character of the qualifier. For example, USER1.GDG.*V01 is valid, but USER1.GDG.G%%10* is not.

Note: Relative generation filtering is not applicable when using the CONVERTV, COPYDUMP, or PRINT functions. These functions do not allow filtering of any kind.

Filtering by data set characteristics

After DFSMSdss has tentatively selected data sets by applying INCLUDE and EXCLUDE criteria, you can apply BY criteria to further restrict the data sets finally chosen. You can, for example, use BY to select data sets by creation date, storage class, and a wide variety of other criteria. The BY keyword takes this form:



Where

Represents

schar

The selection characteristics:

Keyword

Criteria

ALLOC

Allocation type (cylinder, track, block, absolute track, or movable)

CATLG

Whether the data set is currently cataloged or not (using the standard catalog search order)

CREDT

Creation date (absolute or relative)

DSCHA

Whether the data-set-changed flag is on or off

DSORG

Data set organization (SAM, PAM, PDS, PDSE, BDAM, HFS, EXCP, VSAM or zFS)

EXPDT

Expiration date (absolute or relative). Data sets without expiration dates explicitly assigned to them are considered to have an expiration date of zero. If you wish to exclude these data sets from expiration date processing, you must specifically exclude them in your filtering list, that is, BY EXPDT NE 0000000.

EXTNT

Number of allocated or used extents for the entire data set on all the volumes on which it resides

FSIZE

Number of allocated or used tracks for the entire data set on all the volumes on which it resides (data set size)

MULTI

Whether the data set is singlevolume or multivolume (Single volume data sets that have been allocated but have never been opened and are not cataloged may be selected as multivolume)

REFDT

Last-referenced date (absolute or relative)

DATACLAS

Data class for SMS

MGMTCLAS

Management class for SMS

STORCLAS

Storage class for SMS

op

The operator:

Operator**Meaning****EQ or =**

Equal to

LE or <=

Less than or equal to

LT or <

Less than

GT or >

Greater than

GE or >=

Greater than or equal to

NE or !=

Not equal to

arg

An argument that qualifies the selection characteristic (*schar*).

Table 23 on page 259 summarizes the permissible combinations of *schar*, *op*, and *arg*.

Table 23. BY Keywords

schar	op	arg	Notes
ALLOC	EQ	CYL (cylinder allocation)	If MOV is picked, the data sets to be processed cannot be allocated as any of the following: <ul style="list-style-type: none"> • PSU (physical sequential unmovable) • POU (partitioned organization unmovable) • DAU (BDAM unmovable) • ABSTR (absolute tracks) COMPRESS command: If MOV is picked, only data sets allocated as POU cannot be compressed.
	NE	TRK (track allocation)	
		BLK (block length allocation)	
		ABSTR (absolute track allocation)	
		MOV (movable data sets)	

Table 23. BY Keywords (continued)

schar	op	arg	Notes
MULTI	EQ	YES (or 1) NO (or 0)	<p>YES: DFSMSDss processes only multivolume data sets. NO: DFSMSDss processes only singlevolume data sets.</p> <p>DEFRAG command: If a data set's volume sequence number is greater than one in the VTOC, DFSMSDss assumes it is multivolume. If this sequence number is 1, DFSMSDss assumes it is a single volume data set.</p> <p>COMPRESS command: Because DFSMSDss assumes the data set is single volume, you do not need to specify MULTI.</p> <p>Note: Single volume data sets that have been allocated but have never been opened and are not cataloged may be selected as multivolume.</p>
CATLG	EQ	YES (or 1) NO (or 0)	<p>YES: Only currently cataloged data sets are processed. NO: Only uncataloged data sets are processed.</p> <p>DUMP command: The CATLG filter is valid only when used with an input volume list (INDD, INDY, LOGINDD, LOGINDY, STORGRP).</p> <p>COPY command: The CATLG filter is valid only when used with an input volume list (INDD, INDY, LOGINDD, LOGINDY, STORGRP).</p> <p>RESTORE command: The target data set is tested to determine if it is cataloged or not.</p> <p>Note:</p> <ol style="list-style-type: none"> 1. Data sets cataloged through JCL are not cataloged until the job step ends. Therefore, when you use the (CATLG,EQ,NO) filter, you should also use a CREDIT or REFDT filter. The value used in the CREDIT or REFDT filter should be for data sets at least two days old: (CREDIT,LE,*, -2) or (REFDT,LE,*, -2). 2. A data set is considered uncataloged if it is not cataloged in the standard order of search or if it is cataloged in an unavailable catalog.
CREDIT EXPDT REFDT	LT GT EQ NE GE LE	<p>[yy]yyddd(n): 4-digit (or 2-digit) year and 3-digit day, modified by an optional 1-digit to 4-digit positive or negative number of days, n. The year values range from 1900 through 9999. For example, 1998100 (or 98100) is the 100th day of 1998 and 2000001 (or 00001) is the first day of 2000.</p> <p>*(,n): Current date (run date), modified by an optional 1-digit to 4-digit positive or negative number of days, n. For example, *, -5 is five days before this job is run.</p> <p>NEVER: Never-expire date (valid only for EXPDT). A never-expire data set has an expiration date in its VTOC entry of 99365, 99366, or 99999.</p>	<p>The preferred form of a date has a 4-digit year (yyyyddd). When you specify a date with a 4-digit year other than a date of all zeros, the smallest valid date before modification is 1900001. When you specify a date with the 2-digit year format (yyddd), a yy of 00 through 49 indicates years 2000 through 2049; a yy of 50 through 99 indicates years 1950 through 1999. You must use the 4-digit year format if you wish to filter on years 1900 through 1949 or on years higher than 2049.</p> <p>CREDIT and EXPDT: For a multivolume data set, the date from its first volume's VTOC is used for checking.</p> <p>EXPDT: When you specify a date of 99365 or 1999365, you are specifying the last day of 1999. When you specify EXPDT with NEVER, 99366, 1999366, 99999, 1999999, or 9999999, you ask DFSMSDss to look for all valid forms of never-expire dates (including 99365 in a data set's VTOC entry). Therefore, BY(EXPDT,EQ,1999365) selects no data sets. DFSMSDss considers the valid forms of never-expire dates as equal to each other and as greater than all other dates. A never-expire date cannot be modified and cannot be specified with the GT operator.</p> <p>REFDT: For a multivolume data set, the latest date from all its VTOCs is used for checking.</p> <p>RELEASE and DEFRAG commands: The date in the VTOC of the volume being processed is used for filtering both single- and multivolume data sets.</p>
DSCHA	EQ NE	YES (or 1) NO (or 0)	<p>For a multivolume data set, the value used for checking is 1 if any of the indicators from all of its VTOCs is 1. Otherwise, the value is 0.</p>

Table 23. BY Keywords (continued)

schar	op	arg	Notes
DSORG	EQ NE	SAM (all sequential data sets, including compressed and striped), PAM (partitioned data sets, including PDS and PDSE), PDS, PDSE, BDAM (all direct access data sets), VSAM (includes all VSAM data set types), zFS (zFS data sets), EXCP (applies to data sets not allocated as any of the listed organizations and not accessed by using any of the listed access methods)	COMPRESS command: Because DFSMSdss assumes data set organization, you do not need to specify it. The selective characteristic of DSORG can be specified more than once.
DATACLAS MGMTCLAS STORCLAS	EQ NE	An appropriate SMS class name.	

Table 23. BY Keywords (continued)

schar	op	arg	Notes
EXTNT FSIZE	LT GT EQ NE GE LE	nnnnnnnn (1-digit to 8-digit decimal number, from 0 to 99999999)	<p>nnnnnnnn is the number of used or allocated extents (EXTNT) or used or allocated tracks (FSIZE).</p> <p>RESTORE command: The data set that was dumped determines the number of used or allocated extents or tracks.</p> <p>DUMP and COPY commands:</p> <ul style="list-style-type: none"> For non-VSAM data sets: <ul style="list-style-type: none"> If ALLDATA or ALLEXCP is specified, FSIZE is equal to the allocated tracks, and EXTNT is equal to the allocated extents. If ALLDATA or ALLEXCP is not specified, FSIZE is equal to the used tracks, and EXTNT is equal to the used extents. <p>Note: A specification of FSIZE,EQ,0 can be used to select PDSE data sets that have no members (that is, have no used directory blocks), and a specification of FSIZE,NE,0 can be used to exclude PDSE data sets that have no members.</p> <ul style="list-style-type: none"> For VSAM data sets, FSIZE is always equal to the allocated tracks, and EXTNT is always equal to the allocated extents. <p>Logical data set COPY, DUMP, and RESTORE commands:</p> <ul style="list-style-type: none"> FSIZE criteria are applied once for the entire data set on all volumes that the data set resides on. For logical processing of HFS files with TYPRUN NORUN, FSIZE equals all of the allocated space. For logical processing of HFS files without TYPRUN NORUN, the used space represents the FSIZE unless ALLDATA was specified. EXTNT criteria are applied to non-VSAM data sets once for the data set on all volumes that the data set resides on. EXTNT criteria are applied to each VSAM data component on all volumes that the VSAM data component resides on. If a VSAM data component is selected by EXTNT filtering, the index component for the cluster is automatically selected. <p>Physical data set DUMP and RESTORE commands:</p> <ul style="list-style-type: none"> FSIZE criteria are applied once for each volume being processed of a non-VSAM or VSAM data set. For HFS data sets, FSIZE equates to the allocated space, not the space actually used. EXTNT criteria are applied once for each volume being processed of a non-VSAM data set or VSAM data component. If a VSAM data component residing on a volume is selected by EXTNT filtering, the index component (if any) residing on the same volume for the cluster is automatically selected. FSIZE and EXTNT can be used to select certain volumes of a multivolume data set without selecting all of the volumes that the data set resides on. <p>DEFRAG commands:</p> <ul style="list-style-type: none"> For VSAM data sets, EXTNT and FSIZE criteria are applied at the VSAM component level for the volume being processed only. EXTNT and FSIZE filtering are performed for both VSAM data and index components; VSAM index components are not automatically selected if the data component for the cluster is selected. For non-VSAM data sets, EXTNT and FSIZE criteria are applied for the volume being processed only. For HFS data sets, FSIZE equates to the allocated space, not the space actually used.

Note:

1. When multiple arguments are specified for an NE operation, DFSMSdss selects only those data sets not matching any of the arguments.
2. When multiple arguments are specified for an EQ operation, DFSMSdss selects those data sets matching any of the arguments.
3. BY criteria do not apply for the CONVERTV or COPYDUMP commands.

Some examples of the BY keywords

If you code

```
BY((ALLOC EQ CYL) (CATLG EQ YES))
```

you receive all cataloged data sets with cylinder allocation.

If you code

```
BY(FSIZE GE 100)
```

you receive all data sets whose size is greater than or equal to 100 tracks.

If you code

```
BY(DSORG EQ (PAM,SAM))
```

DFSMSDss selects all partitioned and sequential data sets.

Standard catalog search order

When catalogs are searched for a data set name, the standard order of the search is:

1. DFSMSDss first searches the catalogs specified with INCAT, if any. If INCAT and ONLYINCAT are specified, the following steps are skipped.
2. DFSMSDss searches the user catalog when the data set name meets one of the following criteria:
 - The data set is a qualified name and is the name of a user catalog.
 - The data set name is the same as the alias of a user catalog.
3. DFSMSDss searches the master catalog.

Broken data set considerations

Broken data sets are data sets that do not comply with defined IBM data set standards. This includes 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.

Chapter 17. DFSMSdss filtering—choosing the z/OS UNIX files you want processed

DFSMSdss requires that an absolute path name be specified for selecting files to be processed. Each Dump or Restore invocation supports up to 255 absolute paths. An absolute path is created by using a combination of the INCLUDE and WORKINGDIRECTORY keywords.

A single WORKINGDIRECTORY absolute path is required and is responsible for identifying a beginning location of files to be processed. Any files along the specified working directory path are not processed as part of the backup.

A maximum of 255 paths can be specified in the INCLUDE keyword. Each path in the INCLUDE keyword is concatenated to the WORKINGDIRECTORY path name in order to achieve an absolute path that DSS processes.

```
(INCLUDE('dir1/dir2/foo.txt')) WORKINGDIRECTORY('/u/usr/ernestof')
```

Would result in the absolute path: /u/usr/ernestof/dir1/dir2/foo.txt

Note: DFSMSdss does not provide wildcard support when processing UNIX files.

For information about DFSMSdss and the way that it determines individual UNIX file attributes, see Chapter 26, “z/OS UNIX file attributes,” on page 645.

Path name considerations

An *absolute path name* is a sequence that begins with a slash for the root, followed by one or more directory names separated with slashes, and ends with a directory name or a file name. The search for the file begins at the root and continues through the elements in the path name until it gets to the final name.

DFSMSdss uses a concatenation of the specified WORKINGDIRECTORY path and each path in the INCLUDE statement to generate an absolute path. The generated path name has the following constraints:

1. Path names should be specified within single quotes.
2. The absolute path can be up to 1023 characters in length. The length includes all directory names, separating slashes, and file name. This does not include the surrounding quotes.
3. File names are limited to 255 characters in length.
4. All valid path names, including path names that do not conform to the portable character set are supported.
5. The following DFSMSdss special characters need to be preceded by an escape character (\) in order to be specified as a path name character. Not escaping the following characters can result in undesired behavior:
 - a. Single quote (') – this character is used as a delimiter to determine the beginning and ending of a path name.
 - b. Spaces – this character is used as a delimiter for command word separation.
 - c. Semicolon (;) – this character is used for command termination.
 - d. Backslash (\) – this character is used as an escape character.
 - e. Asterisk – this character is a typical wildcard designator.

UNIX wildcard characters

DFSMSdss does not support UNIX wildcard characters and instead processes the absolute path(s) that are specified as a concatenation of the WORKINDIRECTORY and INCLUDE statements.

Specifying the following UNIX wildcard characters can result in undesired behavior. Their UNIX representation are listed for informational purposes only:

1. Asterisk (*) – matches any string, including an empty string.
2. Question mark (?) – matches a single character.

Note: The asterisk can be specified in a single scenario during Restore operations. Refer to the INCLUDE keyword for the Restore command for further details on this subject.

Chapter 18. Syntax - DFSMSDss function commands

This topic describes DFSMSDss *function* commands. Function commands specify operations, or "tasks," for DFSMSDss to perform. The following function commands specify such tasks:

- BUILDSEA
- CGCREATED
- CLOUDUTILS
- COMPRESS
- CONSOLIDATE
- CONVERTV
- COPY
- COPYDUMP
- DEFRAE
- DUMP
- PRINT
- RELEASE
- RESTORE
- SPACEREL

What DFSMSDss commands do

DFSMSDss commands can perform a variety of functions.

Building the stand-alone IPL-able core image

The DFSMSDss BUILDSEA command allows you to build the Stand-Alone Services IPL-able core image. The core image is used to IPL Stand-Alone Restore.

Using DUMP and RESTORE for backup and recovery

DFSMSDss can be used to back up data so that it can be recovered in the event of hardware failure, application failure, or user error. DFSMSDss can back up and recover data sets, entire volumes, or specific tracks. The DFSMSDss DUMP command is used to back up tracks, volumes, and data sets, whereas the RESTORE command is used to recover them. DFSMSDss may be used for the following backup functions:

- Back up data on direct-access storage devices (DASD).
- Restore data from the backup if the original becomes lost, damaged, or inadvertently changed.
- Back up application data for vital records purposes and disaster recovery.

For information about how to back up and restore Linux for IBM Z partitions and volumes, see [Chapter 12, "Dumping and restoring Linux for IBM Z partitions and volumes,"](#) on page 169.

Moving data with COPY

You must move data in order to replace storage devices, add storage capacity, and meet storage requirements. DFSMSDss allows you to perform the following data set movements:

- Move data sets from old to new DASD.
- Move data sets between Storage Management Subsystem (SMS) and non-SMS-managed volumes.
- Move data sets off a volume when requiring hardware maintenance.

- Move or copy data sets for other purposes.

The DFSMSdss COPY command performs data set, volume, and track movement.

Data sets may move from one DASD volume to another volume of either like or unlike device types. Where *like* devices have the same track capacity (3390 Model 2 and 3390 Model 3), unlike devices have different track capacities (3380 Model K and 3390 Model 3).

If you copy a full volume or range of tracks, however, the DASD must be of like device type. The user must specify the source and target volumes. This process permits only one source volume and one target volume.

Converting to and from Storage Management Subsystem (SMS) with CONVERTV

SMS allows you to match the needs of users' data (like data set organization, size, and format) to the characteristics of storage devices without requiring user knowledge or understanding of the installation's hardware configuration. With SMS, users can both store and retrieve data without awareness of space limitations, device characteristics, and volume serial numbers.

The DFSMSdss CONVERTV command can convert existing volumes to and from SMS management without moving data.

Managing space with COMPRESS, CONSOLIDATE, DEFRAG, and RELEASE

DFSMSdss provides features which consolidate free space on volumes, compress partitioned data sets, release unused data set space and combine data sets extents.

To help prevent out-of-space abends on new allocations you can use:

- The DFSMSdss COMPRESS command to compress partitioned data sets by consolidating unused space at the end of the data set.
- The DFSMSdss CONSOLIDATE command to combine as many extents of a data set into as few contiguous extents as possible .
- The DFSMSdss DEFRAG command to consolidate free space on a volume.
- The DFSMSdss RELEASE command to free unused space for use by other data sets within sequential, partitioned, and extended-format VSAM data sets.

Using COPY for partitioned data set (PDS) and partitioned data set extended (PDSE) conversions

The DFSMSdss COPY command can move or copy a PDS, then convert the PDS to a PDSE, and vice versa.

Copying DFSMSdss-produced dump data with COPYDUMP

The DFSMSdss COPYDUMP command can create a maximum of 255 copies of DFSMSdss-produced dump data.

Printing for diagnostic purposes with PRINT

DASD data may be printed to SYSPRINT or to a sequential data set, in print format. For data set printing, tracks are printed in a logical sequence reflecting the data set on the volume. It does not reflect the data set within the physical cylinder or head sequence.

The DFSMSdss PRINT command prints the following:

- A single-volume, non-virtual storage access method (non-VSAM) data set.
- A single-volume VSAM data set component.
- All or part of the VTOC.

- Ranges of tracks.

Managing extent space efficient volume space with SPACEREL

The DFSMSdss SPACEREL command allows you to release physical space associated with free space extents on the specified extent space efficient (ESE) volumes.

BUILDSA command for DFSMSdss

Use the BUILDSA command to build the IPL-able core image for the Stand-Alone Services program. With the BUILDSA command you can specify the device (card reader, tape drive, or DASD volume) from which Stand-Alone Services will be IPLed. You can also specify the operator console to be used for Stand-Alone Services.

The BUILDSA function allocates temporary data sets if needed for BUILDSA processing. These data sets are deleted when the BUILDSA operation is complete. System-generated, temporary data set names are used.

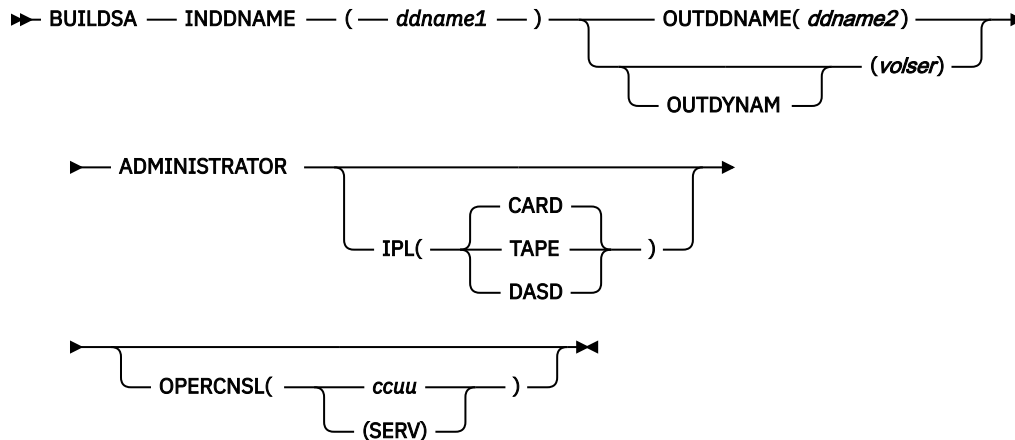
The BUILDSA function invokes the linkage editor (or Binder) utility. Specify UTILMSG=YES when you run the BUILDSA function, and keep the output as a debugging reference when you run the Stand-Alone Services program.

The BUILDSA function builds the IPL-able core image under the current operating system, and determines a record size based on whether the IPL is from card, tape, or DASD.

Note:

1. BUILDSA never allocates a core image data set in the EAS of an EAV.
2. The core image sequential data set is allocated with an EATTR value of NO. This will result in the data set being allocated in the track-managed region of a volume regardless of whether it is an EAV.
3. The Stand-Alone Services modules reside in target library SYS1.SADRYLIB after they are installed and accepted by System Modification Program (SMP) or System Modification Program Extended (SMP/E). If you name this data set something other than SYS1.SADRYLIB, then substitute that name for SYS1.SADRYLIB in the examples shown later in this section.
4. Stand-Alone Services does not support the creation of the core image on an SMS-managed volume.
5. To ensure that Stand-Alone Services is available when you run from DASD, do not delete the SYS1.ADR.SAIPLD.Vvolser data set or move it to another volume.
6. If you IPL from DASD and later change the volume serial number, you must rerun the BUILDSA function to create a new core image data set with the new volume serial number in the name.
7. Consider creating a password or providing other security protection for the SYS1.ADR.SAIPLD.Vvolser data set and for the Stand-Alone Services modules.
8. If you specify TYPRUN=NORUN with the EXEC statement, the BUILDSA task ends without processing input or output.

BUILDSA syntax



Explanation of BUILDSA command keywords

This section describes the keywords for the BUILDSA command.

ADMINISTRATOR



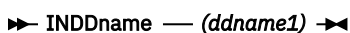
ADMINISTRATOR specifies that you are a DFDSS-authorized or DFSMSdss-authorized administrator for the BUILDSA command. If you are not authorized to use the ADMINISTRATOR parameter, the command is ended with an error message. If you are authorized, and you have access authorization to the output data set, you may create an IPL-able core image for the DASD volume. The ADMINISTRATOR parameter does not give you access to the input data sets, or to the output data sets for IPL(TAPE) or IPL(CARD).

To use the ADMINISTRATOR parameter, all of the following must be true:

- FACILITY class is active.
- Applicable FACILITY-class profile is defined.
- You have READ access to that profile.

For more details, see [“Understanding BUILDSA command authorization levels” on page 529](#).

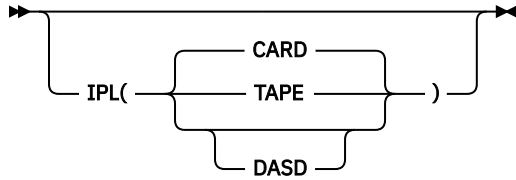
INDDNAME



INDDNAME specifies the DD card in the JCL that identifies the partitioned data set that contains the information that is needed to build the STAND-ALONE Services core image. This is the target library (SYS1.SADRYLIB) where the Stand-Alone Services modules are placed after the SMP or SMP/E install is complete.

ddname1 specifies the name of the DD statement that identifies a volume whose partitioned data set contains the input information.

IPL



IPL specifies the type of device from which Stand-Alone Services is to be IPLed. The system then creates the appropriate loader for the specified device type. If you do not specify the IPL parameter, the system creates a core image suitable for IPLing from a card reader (or a tape drive).

CARD specifies that the core image is to be used for IPLing from a card reader. As a result, the core image is created with BLKSIZE=80 and LRECL=80, and is placed in the data set specified by the OUTDD parameter. You can use this core image to IPL from a card reader (or a tape drive). If you specify CARD, you must also specify the OUTDDNAME parameter. Create the output data set with DSORG=PS, RECFM=F, BLKSIZE=80 and LRECL=80.

TAPE specifies that the core image is to be used for IPLing from a tape drive. As a result, the tape is created with a blocksize optimized for tape. IBM recommends that you specify TAPE when IPLing from a tape drive. If you specify TAPE, you must also specify the OUTDDNAME parameter. Create the output data set with DSORG=PS, RECFM=U, BLKSIZE=32760, and LRECL=32760.

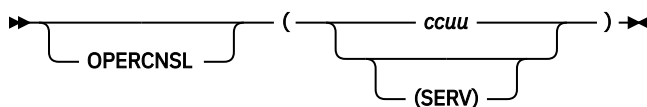
Note that you cannot use an encrypted tape as the source for an IPL. If you attempt to do so, DFSMSdss fails the BUILDSEA command.

DASD specifies that the core image is to be used for IPLing from a DASD volume. As a result, the core image is created with a record size optimized for DASD. If you specify DASD, you must also specify the OUTDYNAM parameter. The core image is placed in data set SYS1.ADR.SAIPLD.Vvolser, which is allocated by Stand-Alone Services in the volume specified on the OUTDYNAM parameter (that is, the volume from which the Stand-Alone Services program is to be IPLed). This core image can only be used to IPL from the DASD volume specified in the OUTDYNAM parameter.

Additionally, the BUILDSEA command invokes the ICKDSF REFORMAT command to place the IPLTEXT and bootstrap records on this volume. The BUILDSEA command provides ICKDSF with IPLTEXT necessary to IPL the Stand-Alone Services core image.

Note: Stand-Alone Services does not support the creation of the core image from an SMS-managed volume.

OPERCNSL



OPERCNSL specifies the device address to be used as the operator console when running Stand-Alone Services.

ccuu specifies that Stand-Alone Services attempt to use the device address as the operator console instead of using the device that generates the first interrupt. Specify a valid device that exists in the configuration where the Stand-Alone Services program will be executed. You can specify a 3-digit or 4-digit address.

SERV specifies that the ES/9000 service console be used as the Stand-Alone Services operator console instead of a unit address.

When OPERCNSL is not specified, Stand-Alone Services loads a wait-state, and then waits for the operator console to generate an interrupt. Limited verification of this parameter is performed during

processing of the BUILDSEA command. For more information about the predefined console, see [“Running stand-alone services with a predefined console”](#) on page 516.

OUTDDNAME

►► OUTDDname — (*ddname2*) ►◄

OUTDDNAME specifies the output location for the IPL-able core image.

ddname2 specifies the DD card in the JCL where the IPL-able core image is placed when IPL(TAPE) or IPL(CARD) is specified.

When IPL(TAPE) is specified, it becomes the physical sequential data set where the core image (including bootstrap) is placed to IPL from a tape drive.

When IPL(CARD) is specified, it becomes the physical sequential data set where the core image (including bootstrap) is placed to IPL from a card reader. If this is a DASD data set, you can then punch it to cards or use it in a VM virtual card reader for IPL.

Specify either OUTDDNAME or OUTDYNAM, not both. When OUTDDNAME is specified, IPL(CARD) is the default. You can specify the IPL(CARD) parameter or the IPL(TAPE) parameter.

OUTDYNAM

►► OUTDYnam — (*volser*) ►◄

OUTDYNAM specifies the output volume serial number for the DASD volume where the IPL-able core image is placed when IPL(DASD) is specified.

volser specifies the name of the DASD volume from which the Stand-Alone Services program will be IPLed.

The IPL bootstrap and IPLTEXT (needed to read in the core image) are placed on this volume. Additionally, Stand-Alone Services allocates data set SYS1.ADR.SAIPLD.Vvolser on this volume and places the core image into this data set. If the SYS1.ADR.SAIPLD.Vvolser data set already exists, Stand-Alone Services deletes and reallocates it.

Specify either OUTDDNAME or OUTDYNAM, not both. When OUTDYNAM is specified, IPL(DASD) must also be specified.

BUILDSEA command examples

Example 1: core image using the default parameters

In this example, the IPL parameter is not specified, so the default (CARD) is used. Stand-Alone Services is created with BLKSIZE=80, LRECL=80, a bootstrap, and is placed in a data set on volume 339001. This core image can be used to IPL from a tape or a card reader. You can either punch the Stand-Alone Services program to a card reader or use IEBGENER to copy the Stand-Alone Services program to tape. The OPERCNSL parameter is not specified; after Stand-Alone Services is IPLed, it loads a wait-state and then waits for the first interrupt to define the operator console.

Note: The DCB parameters must be coded as shown.

```

//BUILDSA JOB accounting information,REGION= nnnnK
//STEP1 EXEC PGM=ADRDSSU,PARM='UTILMSG=YES'
//SAMODS DD DSN=SYS1.SADRYLIB,DISP=SHR
//CARDDD DD DSN=ADRSA.IPLC,UNIT=3390,
// DISP=(NEW,KEEP),VOL=SER=339001,
// SPACE=(TRK,(40,5)),
// DCB=(DSORG=PS,RECFM=F,BLKSIZE=80,LRECL=80)
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
BUILDSA -
INDD(SAMODS) -
OUTDD(CARDDD)

/*

```

The following example copies the Stand-Alone Services core image (created in Example 1) to tape.

Note: The DCB parameters must be coded as shown.

```

//COPYSA JOB accounting information
//STEP1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=ADRSA.IPLC,DISP=SHR,VOL=SER=339001,UNIT=3390,
// DCB=(RECFM=F,BLKSIZE=80,LRECL=80)
//SYSUT2 DD DSN=DSSSA,DISP=(,KEEP),VOL=SER=T11002,LABEL=(,NL),
// DCB=(RECFM=F,BLKSIZE=80,LRECL=80),
// UNIT=3480
//SYSIN DD DUMMY

/*

```

Example 2: core image for IPL from tape

In this example, Stand-Alone Services is created for IPLing in stand-alone mode from a tape. The core image is then placed on an unlabeled tape. The OPERCNSL option is not specified; after Stand-Alone Services is IPLed, it loads a wait-state and then waits for the first interrupt to define the operator console.

Note: The DCB parameters must be coded as shown.

```

//BUILDSA JOB accounting information,REGION= nnnnK
//STEP1 EXEC PGM=ADRDSSU,PARM='UTILMSG=YES'
//SAMODS DD DSN=SYS1.SADRYLIB,DISP=SHR
//TAPEDD DD DSN=ADRSA.IPLT,UNIT=3480,LABEL=(,NL),
// DISP=(NEW,KEEP),VOL=SER=TAPE01,
// DCB=(DSORG=PS,RECFM=U,BLKSIZE=32760,LRECL=32760)
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
BUILDSA -
INDD(SAMODS) -
OUTDD(TAPEDD) -
IPL(TAPE)

/*

```

Example 3: core image for IPL from DASD

In this example, the core image is created for IPLing in stand-alone mode from the DASD with volume label IPLVOL. The core image, the IPL bootstrap, and IPLTEXT are all placed on volume IPLVOL in data set SYS1.ADR.SAIPLD.VIPLVOL. The OPERCNSL option is not specified; after Stand-Alone Services is IPLed, it loads a wait-state and then waits for the first interrupt to define the operator console.

```

//BUILDSA JOB accounting information,REGION= nnnnK
//STEP1 EXEC PGM=ADRDSSU,PARM='UTILMSG=YES'
//SYSPRINT DD SYSOUT=A
//SAMODS DD DSN=SYS1.SADRYLIB,DISP=SHR
//SYSIN DD *
BUILDSA -
INDD(SAMODS) -
OUTDYNAM(IPLVOL) -
IPL(DASD)

/*

```

Example 4: core image for IPL from DASD with OPERCNSL option

In this example, the core image is created for IPLing in stand-alone mode from the DASD with volume label IPLVOL. The core image, the IPL bootstrap, and IPLTEXT are all placed on volume IPLVOL in data set SYS1.ADR.SAIPLD.VIPLVOL. The OPERCNSL customization option is specified for the operator console definition; after Stand-Alone Services is IPLed, it attempts to use the device at address 0009 as the operator console instead of waiting for the first interrupt.

```
//BUILDSA JOB accounting information,REGION= nnnnK
//STEP1 EXEC PGM=ADDRSSU,PARM='UTILMSG=YES'
//SYSPRINT DD SYSOUT=A
//SAMODS DD DSN=SYS1.SADRYLIB,DISP=SHR
//SYSIN DD *
BUILDSA -
        INDD(SAMODS) -
        OUTDYNAM(IPLVOL) -
        IPL(DASD) -
        OPERCNSL(0009)
/*
```

CGCREATED command for DFSMSdss

The CGCREATED command signals that a FlashCopy Consistency Group has been formed or aborted. I/O activity can resume on the "frozen" FlashCopy source volumes previously established by specifying FCCGFREEZE on the COPY command.

The CGCREATED operation is processed at logical subsystem (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.

The freeze and thaw operations require the specified devices support the FlashCopy Consistency Group function. Appropriate LIC level is required on the ESS Model 800, the DS8000®, or the DS6000.

The CGCREATED command is restricted by RACF FACILITY-Class profile 'STGADMIN.ADR.CGCREATE'.

For additional information about creating consistent copies with FlashCopy Consistency Group, refer to [“Backing up volumes with FlashCopy consistency group” on page 129](#).

For additional information about FlashCopy Consistency Group, refer to [z/OS DFSMS Advanced Copy Services](#) and the IBM TotalStorage ESS User's Guide.

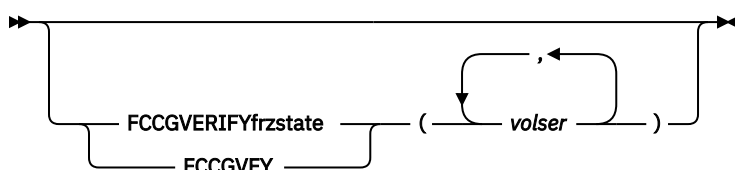
For additional information about RACF authorization, refer to [Chapter 5, “Protecting DFSMSdss functions,” on page 35](#).

For additional information about RACF FACILITY class profiles, refer to [z/OS Security Server RACF Security Administrator's Guide](#).

CGCREATED syntax



A: Optional keywords:



Explanation of CGCREATED command keywords

This section describes the keywords for the CGCREATED command.

ACCESSVOLUME

volser

Specifies the volume serial number of a CKD volume.

ACCESSVOLUME specifies one or more CKD access volumes to be used for a z/OS host to communicate with the storage facility and to direct the "Consistency Group Created" command to the logical subsystems where the access volumes reside. The volumes must be mounted and online. You can specify up to 255 volumes. You cannot specify a nonspecific volume serial number using an asterisk (*).

ACCESSVOLUME is a required keyword on the CGCREATED command. Because the CGCREATED command is processed at LSS level, only one volume needs to be specified for each LSS containing frozen volumes in the FlashCopy Consistency Group.

If the specified access volume does not reside in an LSS with Consistency Group timer enabled or if the storage facility does not support FlashCopy Consistency Group, the CGCREATED command will fail for the volume.

FCCGVERIFY

volser

Specifies the volume serial number of the FlashCopy Consistency Group verification volume.

FCCGVERIFYFRZSTATE specifies one or more volume serial numbers of the verification volumes that DFSMSdss will use 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, or select one volume from each LSS.

CLOUDUTILS command for DFSMSdss

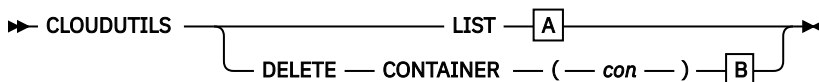
The CLOUDUTILS command can be used to list or delete elements of DFSMSdss created backups in an object-storage cloud. Use this command to:

1. Query which DFSMSdss created containers exist.
2. Query which DFSMSdss created backups exist within a container.
3. Delete one or more DFSMSdss backups within a container.
4. Delete a DFSMSdss backup container.

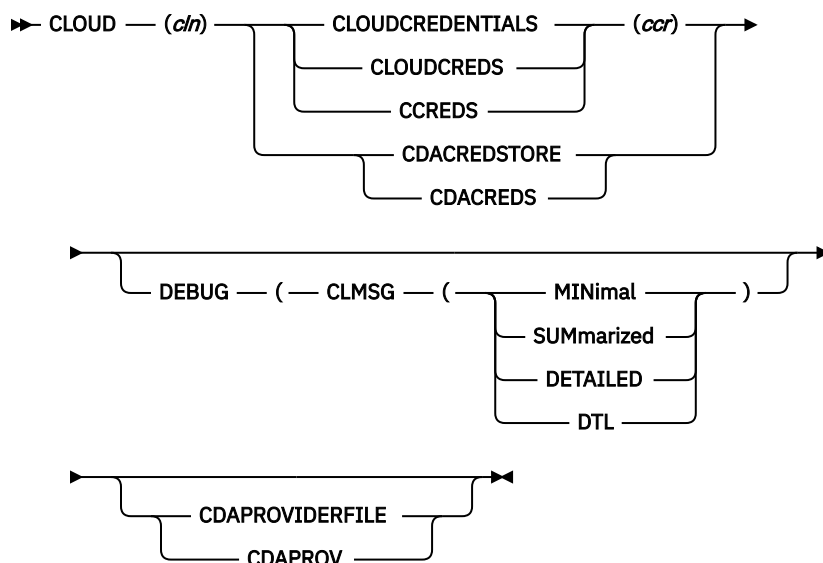
The CLOUDUTILS command does not operate at the object granularity level. Use it only to manage DFSMSdss backups.

When utilizing Cloud Data Access (CDA), the container name may be folded into lowercase.

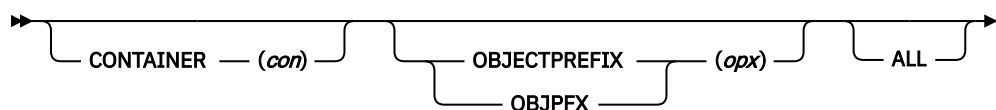
CLOUDUTILS syntax



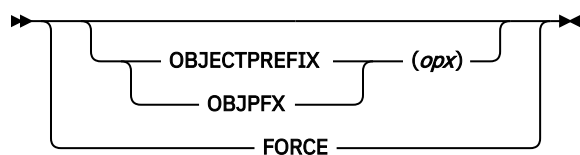
CLOUDUTILS command for DFSMSdss



A: Optional keywords with CLOUDUTILS LIST



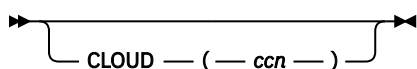
B: Optional keywords with CLOUDUTILS DELETE



Explanation of CLOUDUTILS command keywords

This section describes the keywords for the CLOUDUTILS command.

CLOUD



ccn

Specifies the name of either a CDA provider file or an SMS construct that identifies the cloud storage the dump to be written on.

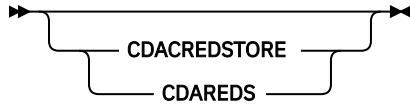
If the keyword **CDAPROVIDERFILE** is present, DFSMSdss will look for a z/OS Cloud Data Access (CDA) provider file with the name *ccn*.json. The provider file will not be used if it does not contain the key-value pair "enableDFSMSdss=YES".

If the keyword **CDAPROVIDERFILE** is absent, DFSMSdss will look for a network connection construct by the name *ccn* in the active SMS configuration.

Note:

1. The name can be up to 30 characters in length.
2. To specify CLOUD, RACF authorization may be required.

CDACREDSTORE



Specifies that the cloud credentials have been stored using the z/OS Cloud Data Access (CDA) Authorization Facility. When specified, DFSMSdss will request the cloud provider credentials from CDA using the cloud name that was specified in the CLOUD keyword.

Note:

1. Do not specify CLOUDCREDENTIALS if you are specifying CDACREDSTORE.

CDAPROVIDERFILE

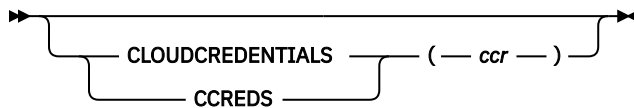


Specifies that DFSMSdss should look for a z/OS Cloud Data Access (CDA) provider file with the specified CLOUD name. If a valid CDA provider file does not exist or is empty, DFSMSdss will search for a network connection construct with the same name in the active SMS configuration.

Note:

- This keyword may be set to a default specification using ADPATCH offset X'62'. For more information on DFSMSdss patch bytes, see [Chapter 14, “DFSMSdss patch area,” on page 213](#)
- For more information about the CDA Provider File, see “Provider file” on page 12.

CLOUDCREDENTIALS



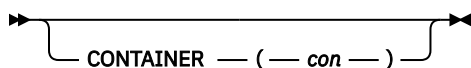
CCR

Specifies a credential (in EBCDIC) that is used when authenticating with a cloud. The credential can be up to 64 characters long. Valid characters are uppercase and lowercase letters A through Z, numerals 0-9, and the following characters: !@#\$%&*~_=:<>?|}. You cannot use embedded spaces, commas (,), forward slash (/), parentheses (()), or semicolons. DFSMSdss removes leading and trailing blanks.

Note:

1. Do not specify CDACREDSTORE when specifying CLOUDCREDENTIALS.
2. The credentials that are specified in your input command stream are not printed to the SYSPRINT output.
3. The credentials must be kept secure. If there are batch JCL jobs that specify this keyword, then those jobs should be in a data set or library that has limited access and controlled by a security product. Do not use this keyword if the credentials cannot be kept secure.

CONTAINER



con

Specifies the DFSMSdss created container.

A DFSMSdss specific prefix may be prepended to the specified container name, according to the cloud storage solution being used:

- When using a TCT cloud storage solution, when no prefix is specified the prefix prepended will be 'SYSZADR.'
- When using the Direct to Cloud storage solution, the prefix prepended will be 'SYSZADR-'
- If either 'SYSZADR.' or 'SYSZADR-' is specified as part of the container name, no prefix will be prepended regardless of the cloud storage solution used.

Note:

1. The name can be up to 128 characters in length. If the CDACREDSTORE keyword is used, the container name may be folded to lowercase.
2. The allowable characters are uppercase letters A-Z, numbers 0-9, special characters \$ @ # - _ , and . in the nonfirst position.

DELETE

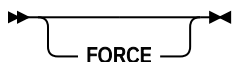
Specifies that the CLOUDUTILS command should perform a DELETE operation.

The behavior of this keyword depends on other keywords:

- If OBJECTPREFIX is not specified, the request is to delete a container. The container must be empty for the operation to succeed.
- If OBJECTPREFIX is specified, the request is to delete the DFSMSdss dump data set objects that correspond to the specified object prefix.

Note:

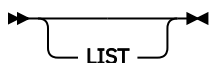
1. Do not specify LIST if you are specifying DELETE.
2. You must specify the CONTAINER keyword if you are specifying the DELETE keyword.
3. To specify DELETE, RACF authorization may be required.
4. To delete a non-empty container, see the FORCE keyword.

FORCE

Specifies that DELETE operation should delete any DFSMSdss dump data sets residing in the container prior to deleting the container itself.

Note:

1. Do not specify OBJECTPREFIX when specifying FORCE.
2. To specify FORCE, RACF authorization may be required.

LIST

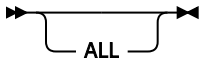
Specifies that the CLOUDUTILS command should perform a LIST operation.

The behavior of this keyword depends on other keywords:

- If CONTAINER is not specified, the request is to list all DFSMSdss-created containers. If a CDA provider file was identified, the container names may be printed in lowercase.
- If CONTAINER is specified and OBJECTPREFIX is not specified, the request is to list all DFSMSdss dump data sets within the specified container. The list will result in object prefixes of one or more backups as specified during their dump operation.
- If OBJECTPREFIX is additionally specified, the request is to list all DFSMSdss dump data sets within the specified container that begin with the specified object prefix. The list will result in object prefixes of one or more backups as specified during their dump operation.

Note:

1. Do not specify DELETE when specifying LIST.
2. If an object-level listing is wanted, use the ALL keyword.

ALL

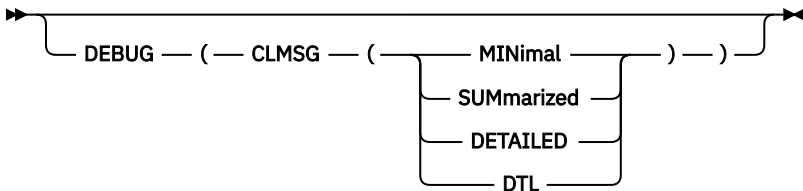
Specifies that the LIST operation should provide an object-level listing in addition to the dump data set listing.

OBJECTPREFIX**opx**

Specifies a prefix of the dump data sets created by DFSMSdss. This can be used as a filter to identify the requested backups to be listed or deleted.

Note:

1. The keyword can be up to 44 characters in length.
2. Single wildcards (*) are supported if they are specified as the last character.

DEBUG

You can use DEBUG as a diagnostic tool. When you specify the CLMSG subkeyword, DFSMSdss issues messages that provide details on the progress of a backup to an object storage cloud. When DEBUG(CLMSG) is not specified, MINimal is the default. Specify DEBUG(CLMSG) with one of the following sub-keywords:

CLMSG(MINimal)

Specifies that DFSMSdss is not to issue any messages that provide detail on the progress of a backup to an object storage cloud.

CLMSG(SUMmarized)

Specifies that DFSMSdss is to issue an informational message for each object that is stored in an object storage cloud.

CLMSG(DETAILED)

Specifies that DFSMSdss is to provide detailed information about each HTTP request that is made to an object storage cloud. If the CDAPROVIDERFILE keyword is used, debug information from Cloud Data Access (CDA) will be printed..

Examples of CLOUDUTILS operations

Examples of the CLOUDUTILS command are as follows:

List containers created by DFSMSdss

```
//LIST JOB accounting information,REGION=nnnnK
//QUERY EXEC PGM=ADDRDSSU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
CLOUDUTILS LIST -
CLOUD(cloudname) CDACREDSTORE
/*
```

List DFSMSdss dump data sets within a container

```
//LIST JOB accounting information,REGION=nnnnK
//QUERY EXEC PGM=ADDRDSSU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
CLOUDUTILS LIST -
CLOUD(cloudname) CDACREDSTORE CONTAINER(container)
/*
```

This operation might result in one or more dump data set object prefixes being listed.

You can reduce the scope of the dump data sets to list by specifying the OBJECTPREFIX keyword.

```
//LIST JOB accounting information,REGION=nnnnK
//QUERY EXEC PGM=ADDRDSSU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
CLOUDUTILS LIST -
CLOUD(cloudname) CDACREDSTORE CONTAINER(container) OBJPFX(objpfx)
/*
```

Delete a container created by DFSMSdss

```
//LIST JOB accounting information,REGION=nnnnK
//QUERY EXEC PGM=ADDRDSSU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
CLOUDUTILS DELETE CONTAINER(container) -
CLOUD(cloudname) CDACREDSTORE
/*
```

The container must be empty unless you also specify the FORCE keyword.

Delete a DFSMSdss dump data set

```
//LIST JOB accounting information,REGION=nnnnK
//QUERY EXEC PGM=ADDRDSSU
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
CLOUDUTILS DELETE CONTAINER(container) -
CLOUD(cloudname) CDACREDSTORE OBJECTPREFIX(objprefix)
/*
```

COMPRESS command for DFSMSdss

The COMPRESS command compresses partitioned data sets on a specified volume. Compressing (degassing) removes unused space between members in a partitioned data set. Depending on the filtering criteria that you specify, you can compress either all or some of the partitioned data sets. This command is useful for compressing system partitioned data sets before you apply maintenance (to avoid certain space-related abends).

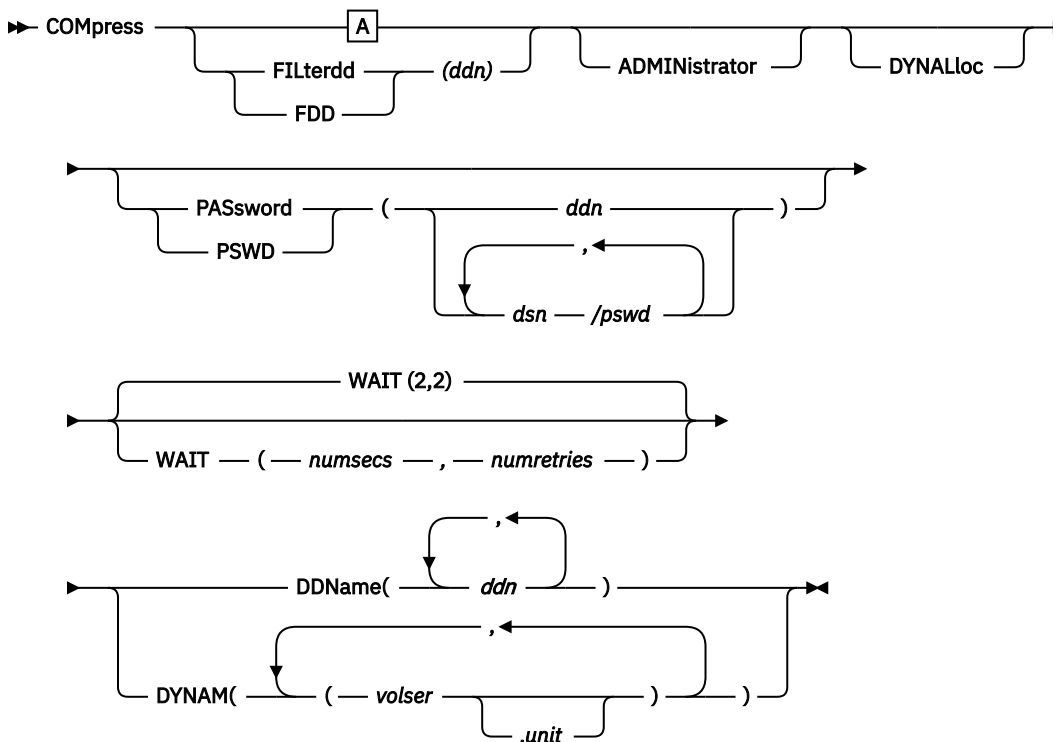
Restriction: You must not compress data sets that contain DFSMSdss or IEBCOPY executable code.

The actual PDS compression is done on the existing volume using the IEBCOPY utility. To prevent loss of data if the system or IEBCOPY abnormally ends during the processing, back up volumes or data sets that meet the filtering criteria before you use the COMPRESS command.

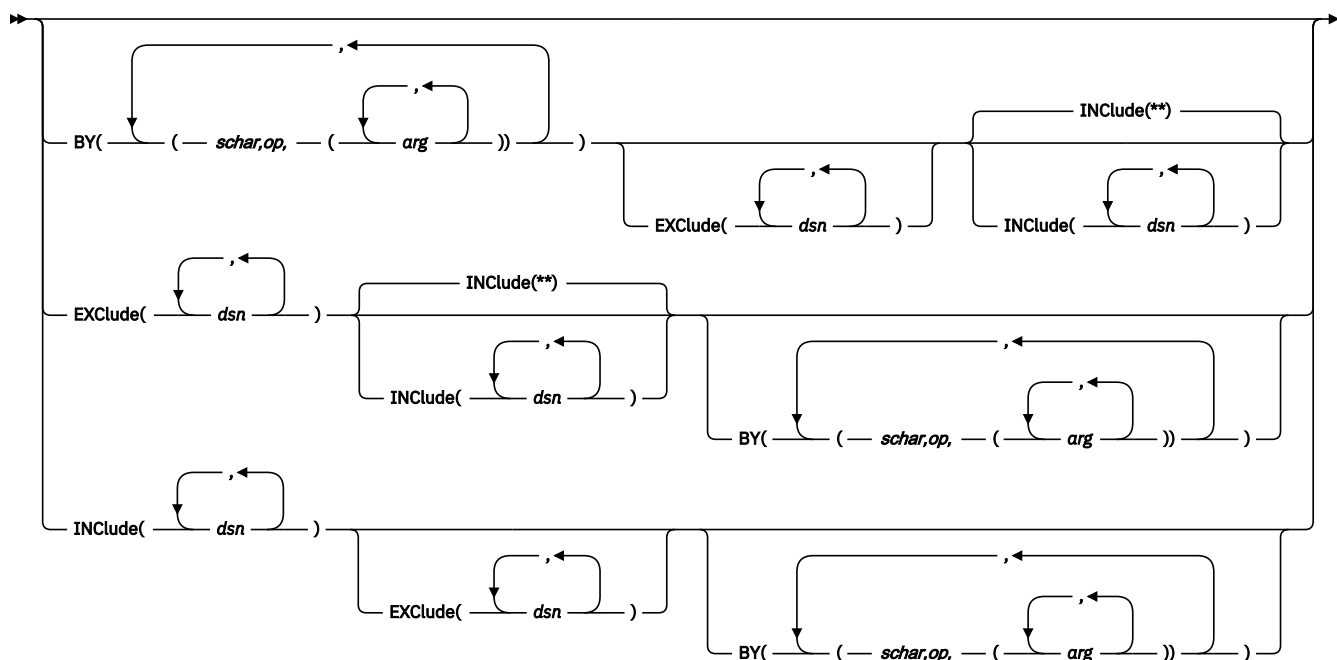
The COMPRESS command cannot process partitioned data sets that:

- Are unmovable
- Have no directory

COMPRESS syntax



A: Additional Keywords with the COMPRESS command



Explanation of COMPRESS command keywords

This section describes the keywords for the COMPRESS command.

ADMINISTRATOR



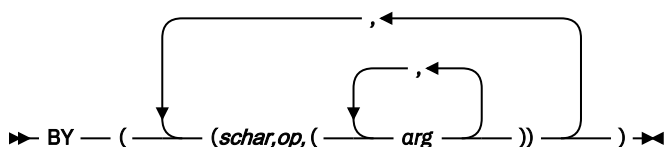
ADMINISTRATOR allows you to act as a DFSMSdss-authorized storage administrator for the COMPRESS command. DFSMSdss-initiated access checking to data sets and catalogs is bypassed. If you are not authorized to use the ADMINISTRATOR keyword, the command ends with an error message.

To use the ADMINISTRATOR keyword, all of the following conditions must be true:

- FACILITY class is active.
- The applicable FACILITY-class profile is defined.
- You have READ access to that profile.

For more details, see [“ADMINISTRATOR keyword” on page 542](#).

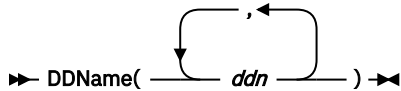
BY



BY specifies that the data sets selected up to this point, by the processing of the INCLUDE and EXCLUDE keywords, are to be further filtered. To select the data set, *all* BY criteria must be met. See [“Filtering by data set characteristics” on page 258](#) for a full discussion of *schar*, *op*, and *arg*. See the separate discussions of the INCLUDE and EXCLUDE keywords for information on how these keywords are specified.

Rule: You must use FILTERDD when you have more than 255 entries in the INCLUDE, EXCLUDE, or BY list keywords.

DDNAME



ddn

Specifies the name of the DD statement that identifies a volume whose partitioned data sets, if selected, are to be compressed. To assure correct processing, each of the DD statements corresponding to a DDNAME (*ddn*) must identify only one volume serial number.

For additional information about storage requirements when processing multiple volumes, see the [“Storage requirements” on page 19](#).

DYNALLOC

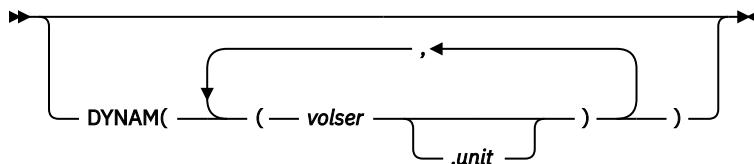


DYNALLOC specifies dynamic allocation, instead of enqueue, to serialize the use of selected partitioned data sets. This allows cross-system serialization in a JES3/MVS environment.

Consider:

- The serialization is of value only when the dynamic allocation/JES3 interface is not disabled.
- Run time increases when DYNALLOC is used to serialize data sets (as opposed to enqueue) because overhead is involved in dynamic allocation and serialization across multiple processors.

DYNAM



DYNAM specifies a dynamically allocated volume whose partitioned data sets, if selected, are to be compressed. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*). Consider using DYNAM instead of DD statements to allocate DASD volumes. This does not appreciably increase run time and permits easier coding of JCL and command input.

volser

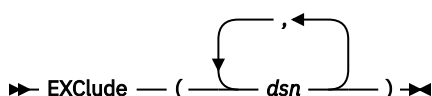
Specifies the volume serial number of a DASD volume to be processed.

unit

Specifies the device type of a DASD volume to be processed. This parameter is optional.

For additional information regarding storage requirements when processing multiple volumes, see the [“Storage requirements” on page 19](#).

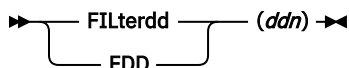
EXCLUDE



dsn

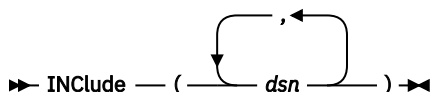
Specifies the name of a data set to be excluded from the data sets selected by the INCLUDE keyword. Either a fully or a partially qualified data set name can be used. See the separate discussions of the INCLUDE and BY keywords for information on how they are specified.

Rule: You must use FILTERDD when you have more than 255 entries in the INCLUDE, EXCLUDE, or BY list keywords.

FILTERDD**ddn**

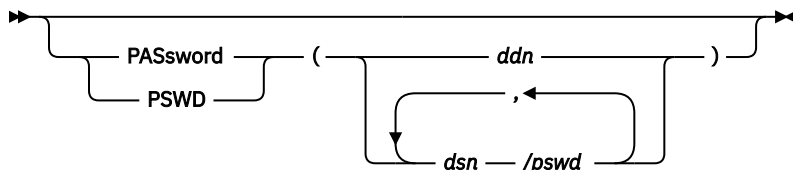
Specifies the name of the DD statement that identifies the sequential data set or member of a partitioned data set that contains the filtering criteria to use. This is in the form of card-image records, in DFSMSdss command syntax, that contain the INCLUDE, EXCLUDE, and BY keywords that complete the syntax of the COMPRESS command.

Rule: You must use FILTERDD when you have more than 255 entries in the INCLUDE, EXCLUDE, or BY list keywords.

INCLUDE**dsn**

Specifies the name of a data set eligible to be compressed. Either a fully or a partially qualified data set name can be used. See “Filtering by data set names” on page 256. If INCLUDE is omitted (but EXCLUDE or BY is specified) or if INCLUDE(**) is specified, *all* partitioned data sets are eligible to be selected for compressing. See the separate discussions of EXCLUDE or BY for information on how these keywords are specified.

Rule: You must use FILTERDD when you have more than 255 entries in the INCLUDE, EXCLUDE, or BY list keywords.

PASSWORD

PASSWORD specifies the passwords DFSMSdss uses for selected password-protected data sets. (Password checking is bypassed for data sets that are protected by the resource access control facility (RACF).) This must be specified only if:

- You do not have the required RACF DASDVOL or RACF DATASET access.
- The installation authorization exit does not bypass the checks.

Note: You should specify the passwords for all data sets that do not have RACF protection but do have password protection. During processing, a utility invoked by DFSMSdss may have to prompt the operator for a password. You can control authorization checking by using the installation authorization exit.

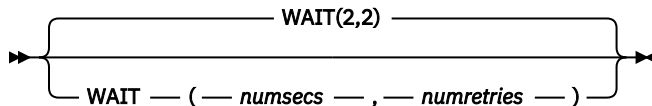
ddn

Specifies the name of the DD statement that identifies the sequential data set or member of a partitioned data set that contains data set names and their passwords. This data set must contain card-image records in DFSMSdss command syntax format.

dsn/pswd

dsn is a fully qualified data set name. *pswd* is its password. If no password follows the slash (/), *dsn* is treated as though it were *ddn*.

Printing of actual data set passwords specified in your input command stream is suppressed in the SYSPRINT output.

WAIT

WAIT specifies to DFSMSdss the length of a wait in seconds and the number of retries to obtain control of a data set.

numsecs

Is a decimal number (0–255) that specifies the interval, in seconds, between retries.

numretries

Is a decimal number (0–99) that specifies the number of times an attempt to gain control of a data set can be retried.

The default for *numsecs,numretries* is WAIT(2,2), which specifies two retries at 2-second intervals. If you do not want to wait for a data set, specify 0 for either numsecs or numretries.

Note: The WAIT keyword does not control wait/retry attempts for system resources (such as the VTOC and the VVDS). For system resources, the default wait time is 3 seconds and the default retry count is 30. This results in a total wait time of 90 seconds.

For information about controlling the wait/retry attempts for system resources, see the [“Controlling the wait/retry time for serialization of system resources \(PN11523\)”](#) on page 215.

Example of compress operations

The following example compresses a selected partitioned data set.

```
//JOB1      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADDRSSU
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
COMPRESS    -
  DYNAM(338000) /* DYNAM ALLOC VOL 338000          */ -
  EXCLUDE(SYS1.***) /* EXCL 'SYS1....' DATA SETS          */ -
                  /* IF THEY MEET THIS CRITERION    */ -
  BY((DSCHA EQ 0)) /* DATA SET WAS BACKED UP              */ -
/*
```

Compress partitioned data sets on volume 338000 if:

- They are not system data sets (EXCLUDE(SYS1.**)), and
- They have not been updated (DSCHA EQ 0) since the last time they were backed up (dumped). This ensures that the data set can be recovered if the system fails while the compress operation is running.

CONSOLIDATE command for DFSMSdss

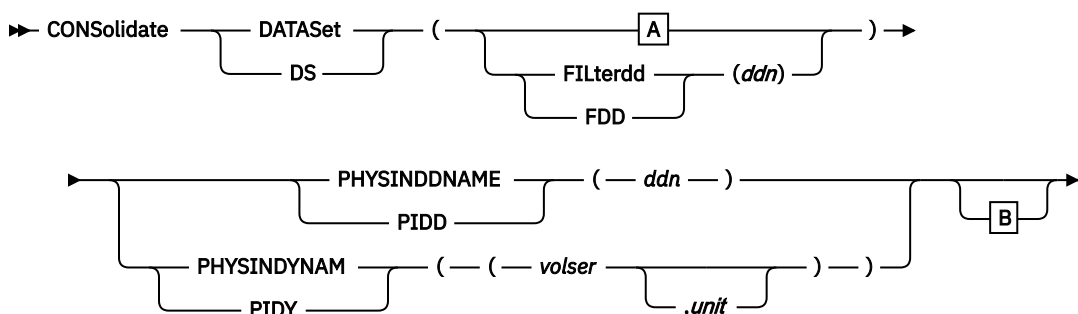
When you enter the CONSOLIDATE command, DFSMSdss performs extent reduction by combining multiple extents of a data set into as few extents as possible given the contiguous free space on a volume. You can specify which data sets are to be included and excluded from this processing.

The amount of time needed for a CONSOLIDATE operation to complete depends on the size of the volume and the number of multiple extent data sets to be processed. In general, larger volumes and data sets with many extents take longer to complete. You can use the MAXTIME keyword to control the amount of time that the CONSOLIDATE operation is allowed to run. For more information, see [“MAXTIME” on page 292](#).

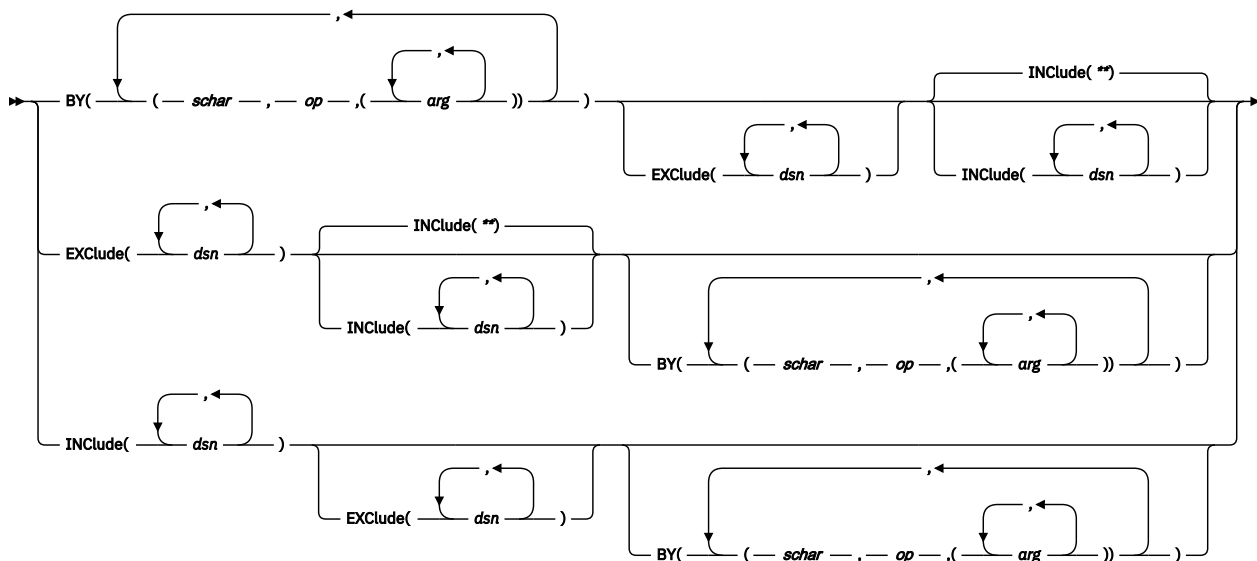
Attention: Canceling the CONSOLIDATE command is strongly discouraged, because doing so can damage data in numerous and unpredictable ways. Before entering this command, consider how long the CONSOLIDATE operation will take by evaluating the size of the volume and the number of data sets with multiple extents to be processed.

CONSOLIDATE command syntax

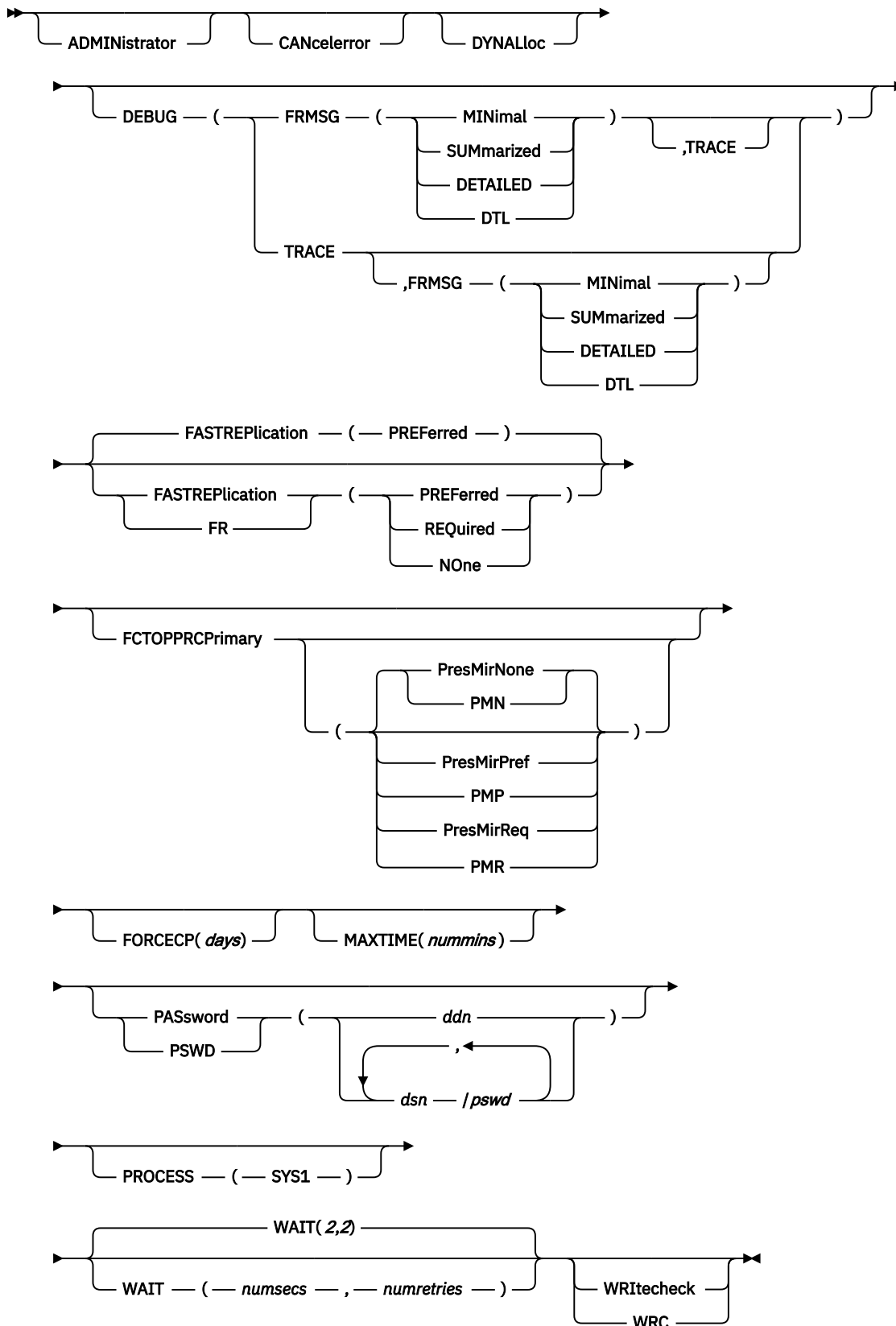
The syntax of the CONSOLIDATE command is:



A: Additional Keywords with CONSOLIDATE



B: Optional Keywords with CONSOLIDATE



Explanation of CONSOLIDATE command keywords

This section describes the keywords for the CONSOLIDATE command.

ADMINISTRATOR



The ADMINISTRATOR keyword allows you to act as a DFSMSdss authorized storage administrator for the CONSOLIDATE command. For administrators, DFSMSdss bypasses access checking for data sets and catalogs.

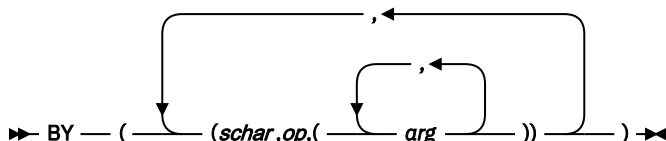
To use the ADMINISTRATOR keyword, all of the following must be true:

- The RACF FACILITY class is active
- The applicable FACILITY class profile is defined
- You have READ access to that profile.

If you are not authorized to use the ADMINISTRATOR keyword, the command ends with an error message.

For more information, see [“ADMINISTRATOR keyword” on page 542](#).

BY



The BY keyword specifies additional filtering criteria for the data sets specified on the INCLUDE and EXCLUDE keywords. To be selected, a data set must satisfy this criteria.

For information about BY filtering, see [“Filtering by data set characteristics” on page 258](#).

Note: You must use FILTERDD when you specify more than 255 entries on the INCLUDE, EXCLUDE, or BY keywords.

CANCELERROR



The CANCELERROR keyword specifies that the CONSOLIDATE operation is to be ended if any of the following errors occur:

- Permanent read error, such as a data check. Processing of the data set ends and the CONSOLIDATE operation is ended.
- Write error, such as an incorrect track format. Processing of the data set ends and the CONSOLIDATE operation continues with the next data set.

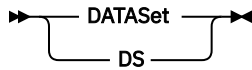
If you omit the CANCELERROR keyword, and CONSOLIDATE processing encounters a permanent read error, the track in error is not copied and processing continues with the next data set.

The CANCELERROR keyword has no effect with the following types of DASD volume errors:

- Equipment check
- Command reject
- Intervention required
- Busout parity.

For information about handling errors for incorrect tracks, see [Chapter 10, “Diagnosing problems in DFSMSdss operations,” on page 155](#).

DATASET

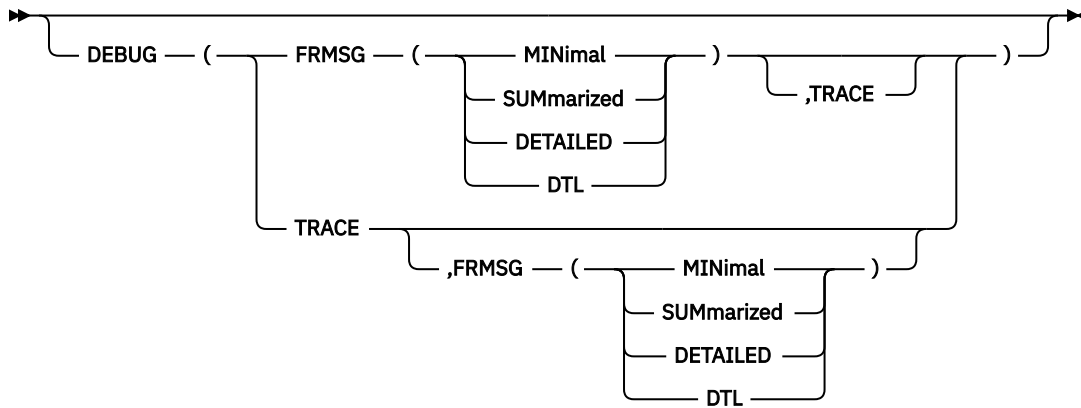


The DATASET keyword specifies the data sets to be consolidated.

For a description of the data set filtering process, see [Chapter 16, “DFSMSDss filtering—choosing the data sets you want processed,”](#) on page 255.

Note: Using the DATASET keyword requires that you also specify the FILTERDD, INCLUDE, EXCLUDE, or BY keywords.

DEBUG



You can use the DEBUG keyword as a diagnostic tool. Specify DEBUG with one of the following sub-keywords:

TRACE

Specifies that DFSMSDss is to print messages that identify the relocated extents.

FRMSG

Specifies that DFSMSDss is to issue messages that explain why you cannot use fast replication or Preserve Mirror during a CONSOLIDATE operation. For Preserve Mirror operations, the DEBUG(FRMSG) keyword might not have an effect if the FlashCopy target is not a PPRC Primary device. Specify DEBUG(FRMSG) with an additional sub-keyword, as follows:

FRMSG(MINIMAL)

Specifies that DFSMSDss is to issue a message with a minimal level of information.

FRMSG(SUMMARIZED)

Specifies that DFSMSDss is to issue a message with summarized information. When applicable, summarized information regarding ineligible volumes is provided in the message text.

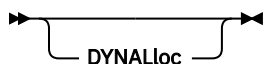
FRMSG(DETAILED)

Specifies that DFSMSDss is to issue a message with detailed information. When applicable, detailed information regarding ineligible volumes is provided in the message text.

Note:

1. If you specify FASTREPLICATION(REQUIRED) without specifying the DEBUG keyword, DFSMSDss issues an informational message whenever a fast replication method cannot be used.
2. The FRMSG sub-keyword overrides the DEBUG=FRMSG parameter specified on the JCL EXEC statement.

DYNALLOC

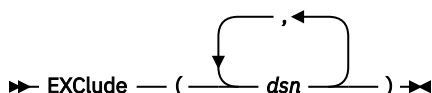


The DYNALLOC keyword requests that dynamic allocation, rather than enqueueing, be used as the serialization method for relocating data set extents. Use DYNALLOC when you require cross-system serialization in a JES3 environment.

Note:

1. Serialization is of value only for dynamic allocation or when the JES3 interface is enabled.
2. Using the DYNALLOC keyword to serialize data set access (as opposed to enqueueing) will increase run-time because of the additional processing involved in dynamic allocation and performing serialization across multiple processors.
3. If a data set passes INCLUDE/EXCLUDE filtering, and is migrated before BY filtering, and you specify the DYNALLOC keyword, dynamic allocation causes the data set to be recalled. DFSMSDss waits for the recall processing to complete. If the data set is recalled to a different volume, DFSMSDss issues a message to indicate that the VTOC entry was not found.
4. For an HFS source data set, CONSOLIDATE processing ignores the DYNALLOC keyword, and, instead, attempts to obtain a SYSZDSN enqueue for the data set. If the enqueue attempt fails, DFSMSDss attempts to quiesce the data set.

EXCLUDE

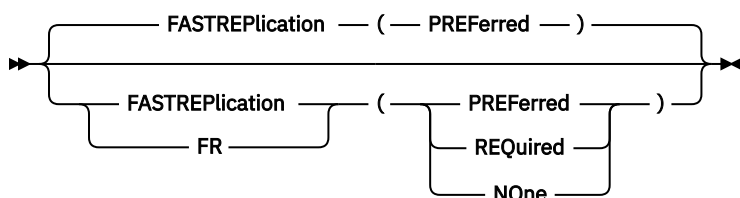


The EXCLUDE keyword specifies one or more data sets (*dsn*) to be excluded from processing by the INCLUDE keyword. You can specify fully or partially qualified data set names.

For information about the INCLUDE and BY keywords, see [“INCLUDE” on page 292](#) and [“BY” on page 288](#).

Note: You must use FILTERDD when you specify more than 255 entries on the INCLUDE, EXCLUDE, or BY keywords.

FASTREPLICATION



The FASTREPLICATION keyword specifies whether the use of fast replication is required, preferred, or not desired for the CONSOLIDATE operation. This keyword applies to fast replication methods, such as FlashCopy and SnapShot.

REQUIRED

Specifies that fast replication must be used. If fast replication cannot be used, DFSMSDss stops processing the current data set, and continues with subsequent data sets. If you do not specify the DEBUG keyword, DFSMSDss issues summarized information to indicate why you cannot use fast replication.

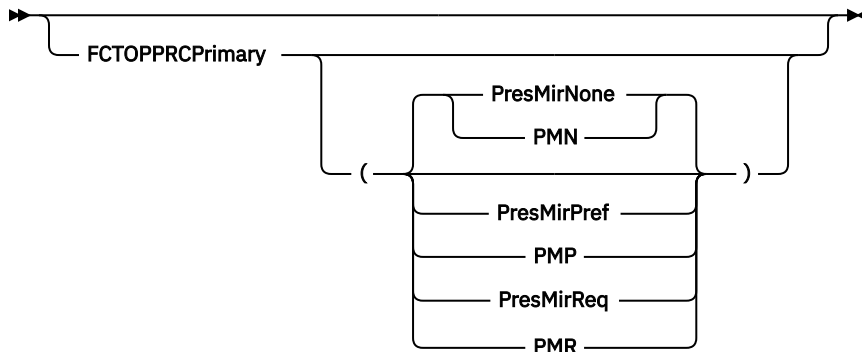
PREFERRED

This is the default. The PREFERRED keyword specifies that the use of fast replication is preferred. If fast replication cannot be used, DFSMSdss completes the CONSOLIDATE operation using traditional data movement methods.

NONE

Specifies that fast replication should not be used. DFSMSdss completes the CONSOLIDATE operation using traditional data movement methods.

FCTOPPRCPPRIMARY



The FCTOPPRCPPRIMARY keyword specifies that if FlashCopy is used to perform the CONSOLIDATE operation, a Peer-to-Peer Remote Copy (PPRC) primary volume can become a FlashCopy target volume. Use the following sub-keywords to specify whether the device pair is allowed to go to duplex pending state if the target volume of the FlashCopy operation is a metro mirror primary device:

PRESMIRREQ

specifies that if the target volume is a Metro Mirror primary device, the pair must not go into a duplex pending state as the result of a FlashCopy operations.

PRESMIRPREF

specifies that if the target volume is a Metro Mirror primary device, it would be preferable that the pair does not go into a duplex pending state as the result of a FlashCopy operation. However, if a Preserve Mirror operation cannot be accomplished, the FlashCopy operation is still to be performed.

The PRESMIRPREF option is not valid for DS888x and newer, and compatible, storage subsystems. If the PRESMIRPREF option is specified and the volumes involved in the operation reside on DS888x or newer storage subsystems, the command results in a warning or error message. This is also true of the corresponding option provided in byte 33 (X'21') in the ADRUFO data area.

PRESMIRNONE

specifies that Preserve Mirror operation is not to be done, even if all of the configuration requirements for a Preserve Mirror operation are met. If the target specified is a Metro Mirror primary device, the pair is to go into a duplex pending state while the secondary device is updated with the tracks to be copied. PRESMIRNONE is the default if you specify FCTOPPRCPPrimary without a subkeyword.

Attention: When you specify FCTOPPRCPPrimary or FCTOPPRCPPrimary(PRESMIRNONE), the FlashCopy operation causes a PPRC primary volume to become a FlashCopy target volume. A Metro Mirror pair currently in full duplex state, goes into a duplex pending state when the FlashCopy relationship is established. When Metro Mirror completes the copy operation, the Metro Mirror pair goes to full duplex state. To prevent Metro Mirror pairs from going to duplex pending state during FlashCopy operation, you must specify FCTOPPRCPPrimary(PRESMIRREQ).

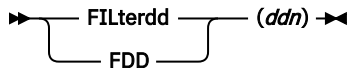
Note:

1. Using FCTOPPRCPPRIMARY might require RACF authorization.
2. When FlashCopy is not used to perform the CONSOLIDATE operation, the FCTOPPRCPPRIMARY keyword is ignored.

3. If you do not specify FCTOPPRCPRIMARY, or your storage subsystem does not support this capability, a PPRC primary volume cannot become a FlashCopy target volume.
4. When you use FCTOPPRCPRIMARY, the FlashCopy operation causes a PPRC primary volume to become a FlashCopy target volume. The PPRC-SYNC volume pair currently in full duplex state changes to a duplex pending state when the FlashCopy relationship is established. When PPRC completes the copy operation, the PPRC-SYNC volume pair changes to full duplex state.

For more information about IBM Remote Pair FlashCopy, Metro Mirror, also known as synchronous Peer-to-Peer Remote Copy (PPRC), and other copy services functions, see [*z/OS DFSMS Advanced Copy Services*](#).

FILTERRDD



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 (the INCLUDE, EXCLUDE, and BY keywords). This data set must contain card-image records in DFSMSdss command syntax format.

Note: You must use FILTERDD when you specify more than 255 entries on the INCLUDE, EXCLUDE, or BY keywords.

FORCECP

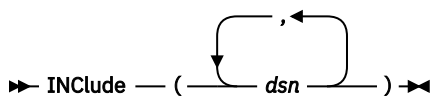


The FORCECP keyword specifies that checkpoint data sets on SMS-managed volumes can be processed. Checkpoint indicators are removed from the resulting consolidated data set.

days

Specifies the number of days (0-255) that must elapse since the last referenced date before the data set can be processed.

INCLUDE



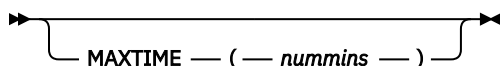
The INCLUDE keyword specifies one or more data sets that are eligible to be consolidated. You can use either fully or partially qualified data set names. If you specify INCLUDE(**) or omit INCLUDE, but specify EXCLUDE or BY, all data sets are eligible to be selected for processing.

Restrictions

- You must use **FILTERDD** when you have more than 255 entries in **INCLUDE**, **EXCLUDE**, or **BY** list keywords.
- **DFSMSdss** does not support **INCLUDE** filtering of non-VSAM data sets using an alias.

For more information, see “Filtering by data set names” on page 256.

MAXTIME



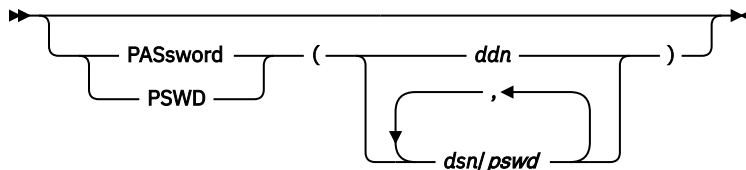
Specifies the maximum time, in minutes, for the CONSOLIDATE operation to complete. MAXTIME is checked after each data set is processed. When the MAXTIME value is reached, the CONSOLIDATE operation ends.

nummins

Specifies the maximum number of minutes (0-9999 in decimal) that a CONSOLIDATE operation can run. A value of 0 is ignored.

Note: The elapsed time of the CONSOLIDATE operation might be slightly longer than the MAXTIME value because the value is checked after each data set is processed.

PASSWORD



The PASSWORD keyword specifies the passwords that DFSMSdss is to use for password-protected data sets. (Password checking is bypassed for RACF-protected data sets.)

This keyword is required only when either of the following is true:

- You do not have the required RACF DASDVOL or RACF data set access
- The installation authorization exit does not bypass the checks.

ddn

Specifies the name of the DD statement that identifies the sequential data set, or member of a partitioned data set, that contains data set names and their passwords. This data set must contain card-image records in DFSMSdss command syntax format.

dsn/pswd

dsn is a fully qualified data set name, *pswd* is the data set password. If no password follows the slash (/), *dsn* is treated as if it were *ddn*.

Note:

1. Specify passwords for all data sets that do not have RACF protection, but do have password protection. During processing, a utility invoked by DFSMSdss might prompt the system operator to supply a password. You can control authorization checking through the installation authorization exit.
2. Do not request password prompting for VSAM data sets.
3. Catalog passwords are not supported. Instead, it is recommended that you use RACF or another access control facility to secure your catalogs.
4. The SYSPRINT output does not show the data set passwords that are specified in the input command stream.
5. When you use a system utility to perform the CONSOLIDATE operation, you must supply the password for each password-protected data set selected, or have the proper RACF data set access authority.

PHYSINDDNAME



The PHYSINDDNAME keyword specifies the name of the DD statement (*ddn*) that identifies the input volume to be processed. Specify only one volume per CONSOLIDATE operation.

PHYSINDYNAM

```

➔ PHYSINDYnam ( — volser — , — unit — ) ➔
    PIDY

```

The PHYSINDYNAM keyword specifies dynamic allocation for the volume to be processed.

volser

Specifies the volume serial number of a DASD volume to be processed.

unit

Specifies the device type of a DASD volume to be processed. This parameter is optional.

Note:

1. The volume must be mounted and online.
2. Do not specify a non-specific volume serial number, such as an asterisk (*).
3. Specify only one volume for CONSOLIDATE processing.
4. Consider using PHYSINDYNAM instead of PHYSINDDDNAME to allocate DASD volumes. Doing so does not appreciably increase run-time and might simplify your coding of JCL and command input.

PROCESS

```

➔ PROCESS — ( — SYS1 — ) ➔

```

The PROCESS keyword specifies that DFSMSdss is to allow data sets with a high-level qualifier of SYS1 to be consolidated.

Note:

1. SYS1.VVDS and SYS1.VTOCIX data sets are not processed.
2. To use PROCESS(SYS1), you might require RACF authorization.

WAIT

```

➔ WAIT( 2,2 ) ➔
    WAIT — ( — numsecs — , — numretries — ) —

```

The WAIT keyword specifies the maximum wait time, and the number of attempts permitted, for a CONSOLIDATE operation to obtain control of a data set.

numsecs

Specifies a decimal number (0-255) that designates the interval, in seconds, to wait before attempting another pass through the list of selected data sets.

numretries

Specifies a decimal number (0-99) that designates the number of attempts permitted to obtain control of a data set.

The default for *numsecs*, *numretries* is (2,2), which specifies two retries at a two-second intervals. To avoid waiting for a resource, specify zero (0) for either *numsecs* or *numretries*.

The WAIT keyword does not control wait or retry attempts for system resources (such as the VTOC and the VVDS). For system resources, the default wait time is three seconds and the default retry count is 30. This results in a total wait time of 90 seconds.

For information about controlling the wait or retry attempts for system resources, see [“Controlling the wait/retry time for serialization of system resources \(PN11523\)”](#) on page 215.

WRITECHECK



The WRITECHECK keyword specifies that the data set being processed is to be verified for successful completion.

Note:

1. The WRITECHECK keyword is not supported for extended-format sequential data sets.
2. This keyword increases the overall elapsed time of the CONSOLIDATE operation.

Example of a CONSOLIDATE operation

This example shows a CONSOLIDATE operation. All eligible data sets on the volume identified by the DASD DD statement are to be filtered using the specified INCLUDE and EXCLUDE criteria. Data sets that satisfy this filtering criteria are processed.

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD      DD       UNIT=3390,VOL=(PRIVATE,SER=111111),DISP=OLD
//SYSIN     DD       *

CONSOLIDATE DATASET(INCLUDE(**) -
EXCLUDE(USER2.**.LIST,*.LOAD)) -
PHYSINDDNAME(DASD)
/*
```

CONVERTV command for DFSMSdss

The CONVERTV command is used to convert existing volumes to and from SMS management without data movement. The CONVERTV command performs three functions:

- Locks volumes that are ready for conversion to prevent new data set allocations (PREPARE keyword).
- Examines volumes identified by SMS to determine if they can be converted to SMS management (TEST keyword). No conversion is actually performed, but DFSMSdss identifies any data sets that cannot be converted to SMS management and why they cannot be converted.
- Performs conversion of volumes into or out of SMS management. Any conditions that prevent conversion are identified.

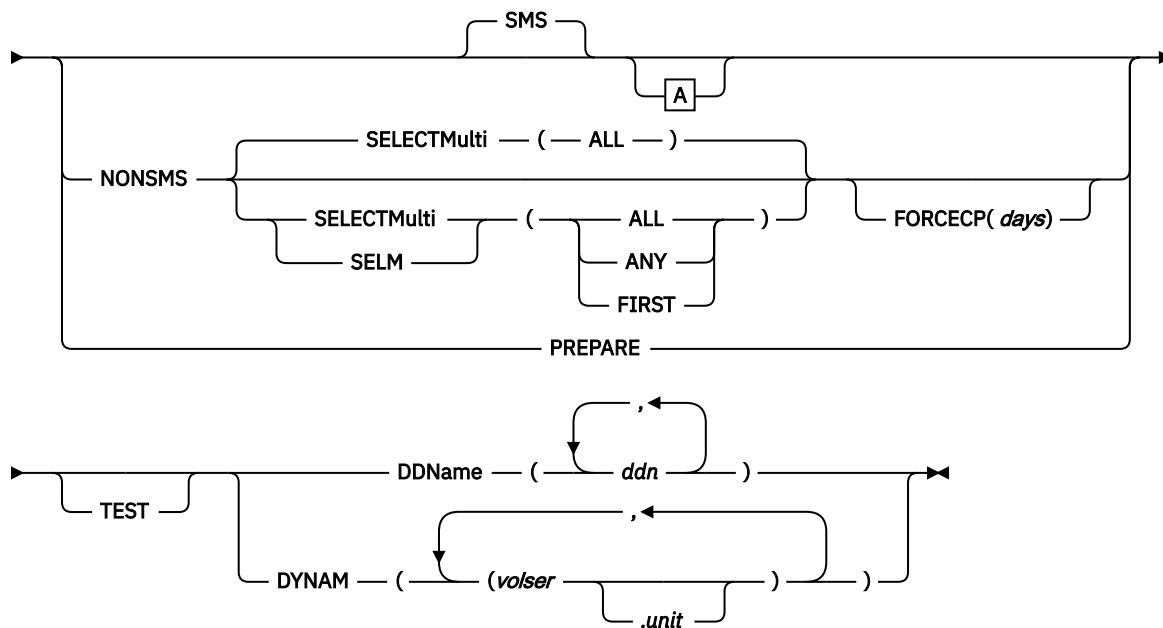
Guideline: Proper RACF security authorization might be required.

For additional information about RACF security authorization, see the [Chapter 5, “Protecting DFSMSdss functions,”](#) on page 35.

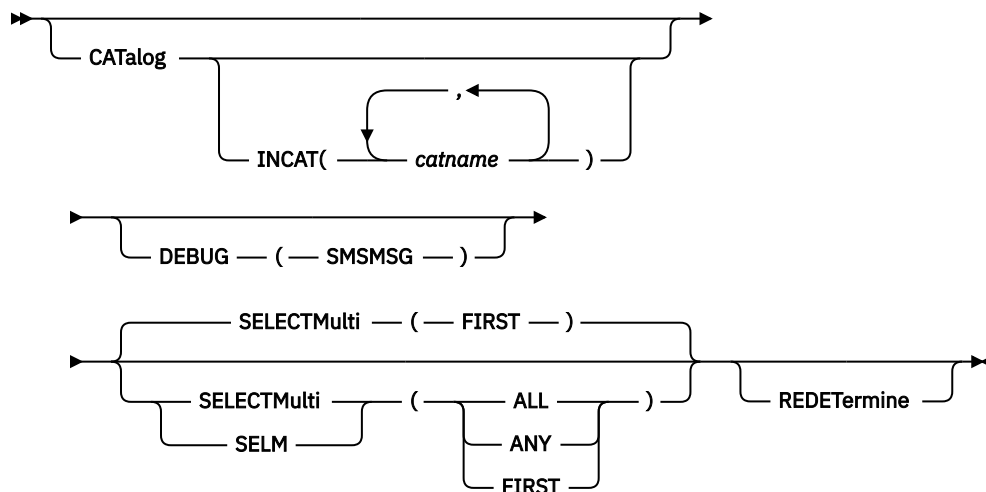
CONVERTV command syntax

The syntax of the CONVERTV command is:

➡ CONVERTV ➡



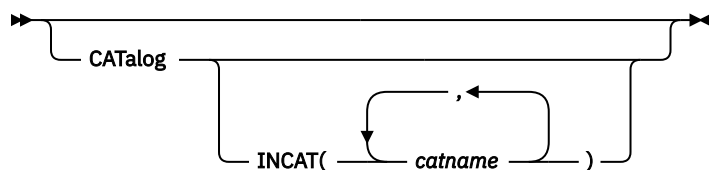
A: The syntax of optional keywords with CONVERTV SMS is:



Explanation of CONVERTV command keywords

This section describes the keywords for the CONVERTV command.

CATALOG



CATALOG specifies that if a data set's catalog entry is not found in the standard order of search, the data set is to be cataloged during the conversion.

If the CATALOG keyword has not been specified, and a data set's catalog entry is not found in the standard order of search, the data set is not converted.

INCAT

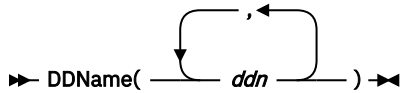
Specifies input catalogs that are not in the standard search order. This allows non-VSAM data sets cataloged outside the standard order of search to be processed.

catname

Specifies a fully qualified catalog name.

If CATALOG is specified without INCAT, a single volume, non-VSAM data set cataloged outside the standard order of search might be cataloged in more than one place.

DDNAME

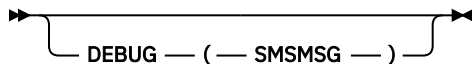


DDNAME specifies a volume that you want converted. Use this keyword to designate the list of volumes that need conversion. Use this keyword when you do not use the DYNAM keyword.

ddn

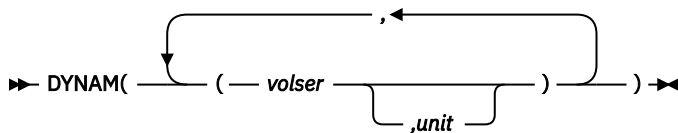
Specifies the name of the DD statement that identifies a volume to be processed. Up to 255 DDNAMEs can be specified.

DEBUG



DEBUG(SMSMSG) instructs DFSMSdss to display ACS WRITE statements to the job output.

DYNAM



DYNAM specifies the volume that you want to process must be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number by using an asterisk (*). Use this keyword to designate the list of volumes that you need to convert. Use the DYNAM keyword when you do not specify the DDNAME keyword.

Consider using DYNAM instead of DD statements to allocate DASD volumes. This does not noticeably increase run time and presents easier coding of JCL and command input.

volser

Specifies the volume serial number of a DASD volume to be processed. Up to 255 volumes can be specified.

unit

Specifies the device type of a DASD volume to be processed. This parameter is optional.

FORCECP



FORCECP specifies that checkpointed data sets resident on the SMS volume can be converted to non-SMS management. Checkpoint indications are removed from the data set during conversion.

days

Specifies the number of days that must have elapsed since the last referenced date before the data set can be converted. It is a one-to-three-digit number in the range of zero to 255.

INCAT

See “[CATALOG](#)” on page 296.

NONSMS

➤ NONSMS ➤

NONSMS specifies that a volume and all of the data sets on that volume be converted from SMS management to non-SMS management.

PREPARE

➤ PREPare ➤

PREPARE specifies that a volume is to be prepared for SMS without conversion of data sets. This prevents the volume from changing prior to performing the full SMS conversion. After the PREPARE is requested, the volume is placed in initial status and you cannot allocate new data sets. However, you can delete existing data sets.

The NONSMS keyword must be specified to return the volume to non-SMS management.

REDETERMINE

➤ REDETermine ➤

REDETERMINE specifies that the SMS class information is to be reset for data sets previously converted to SMS management whose SMS management class or SMS storage class do not match those returned by the current ACS routines. REDETERMINE allows management class and storage class to be reset, but does not update the data class.

If REDETERMINE is used with the TEST keyword, a report is produced specifying all data sets eligible for conversion, including those already converted.

SELECTMULTI

➤ SELECTMulti (— ALL —) ➤

SELECTMulti ({ ALL ANY })

SELM FIRST

SELECTMULTI specifies how cataloged multivolume data sets are to be selected during conversion to or from SMS management. The volume list is the list of volumes supplied by the DDNAME or DYNAM keyword.

ALL

Specifies that DFSMSdss *not process* a multivolume data set unless all of the volumes that contain a part of the non-VSAM data set or VSAM base cluster are in the volume list specified by DDNAME or DYNAM. ALL is the default for non-SMS processing.

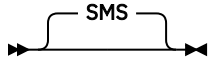
ANY

Specifies that DFSMSdss process a multivolume data set when any part of the non-VSAM data set or VSAM base cluster is on a volume in the volume list specified by DDNAME or DYNAM.

FIRST

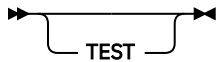
Specifies that DFSMSdss process a multivolume data set only when the DDNAME or DYNAM volume list includes the volume that contains the first part of the non-VSAM data set or the primary data component of the base cluster for a VSAM sphere. FIRST is the default for SMS processing.

SMS



SMS specifies that a volume and all of the data sets on that volume are to be converted to SMS management. SMS is the default when the SMS, NONSMS, or PREPARE keyword is not specified.

TEST



TEST specifies that DFSMSdss is to verify that a volume and its data sets are eligible for conversion or for preparation. The TEST keyword functions just as if TYPRUN=NORUN had been specified on the JCL EXEC PARM field. DFSMS must be active to use this function.

Note: It is also possible to use TEST to verify that the ACS algorithms would process correctly because the resulting report indicates the classes associated with the various data sets on the volume.

Examples of CONVERTV operations

The following are examples of the CONVERTV command.

Example 1: using the CONVERTV command to simulate conversion

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//SYSIN     DD       *

  CONVERTV SMS -
    DYNAM((VOL001,3380),(VOL002,3380),(VOL003)) -
    TEST
/*
```

The preceding example uses the TEST keyword to simulate conversion. The TEST keyword produces a report that indicates whether the three volumes (VOL001, VOL002, and VOL003) can be converted to SMS management.

Example 2: using the CONVERTV command to convert to SMS

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=*
//DVOL1     DD       UNIT=SYSDA,VOL=SER=338001,DISP=OLD
//DVOL2     DD       UNIT=SYSDA,VOL=SER=338002,DISP=OLD
//SYSIN     DD       *

      CONVERTV -
        DDNAME(DVOL1,DVOL2) -
        SMS -
        INCAT(SYS1.ICFCAT.V338002) -
        SELECTMULTI(FIRST) -
        CATALOG
/*
```

The non-SMS-managed volume 338002 and the SMS-managed volume (in INITIAL state) 338001 are converted to SMS. The volume 338001 has been placed in the initial state by the storage administrator. Regardless of where the data sets reside, all multivolume data sets whose first extent is on volume 338001 or 338002 are processed. In addition, there are some data sets on volume 338002 cataloged in the user catalog SYS1.ICFCAT.V338002. These data sets are uncataloged from the user catalog and cataloged in the standard order of search. The INCAT keyword provides access to the user catalog.

Example 3: using the CONVERTV command to convert from SMS

```
//JOB1      JOB      accounting information,nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=*
//SYSIN     DD       *

      CONVERTV -
        DYNAM(338003) -
        NONSMS
/*
```

This example converts a volume to non-SMS-managed.

COPY command for DFSMSdss

The DFSMSdss COPY command performs data set movement, volume movement, and track movement from one DASD volume to another.

You can copy data sets to another volume of either like or unlike device types. Like devices have the same track capacity (3390 Model 2 and 3390 Model 3), while unlike devices have different track capacities (3380 Model K and 3390 Model 3).

However, the DASD must be of *like* device type if you copy a full volume, range of tracks, or physically copy a data set. The user must specify the source volumes and the target volumes. DFSMSdss only allows one source volume and one target volume.

DFSMSdss offers two ways to process COPY commands as follows:

- *Logical processing* is data set-oriented, which means that it operates against data sets and volumes independently of physical device format.
- *Physical processing* can operate against data sets, volumes, and tracks, but is oriented toward moving data at the track-image level. The processing method is determined by the keywords specified on the command.

Integrated catalog facility catalogs should not have a high-level qualifier of SYSCTLG because this causes DFSMSdss to treat them as control volumes.

DFSMSdss COPY will always preserve data set encryption attributes of the source data set. Therefore, new allocations will be defined with the source encryption attribute. If a pre-allocated target data set is encountered, it must also be encrypted to be considered a usable target data set. The usable

pre-allocated target will be overwritten with the source encryption attributes (which includes the key label).

For more information about using the COPY command, see [“Backup with concurrent copy”](#) on page 45 and [“Moving data sets with concurrent copy”](#) on page 106.

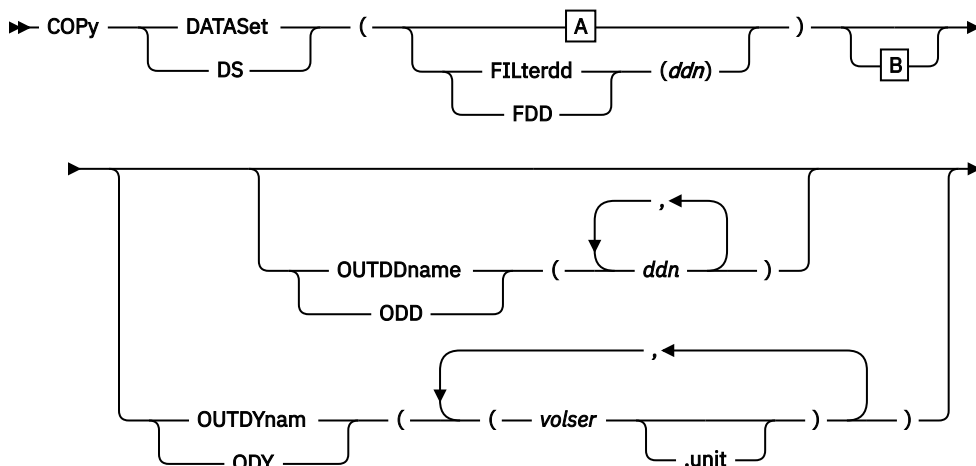
Special considerations for COPY

The following special considerations may apply when you perform a COPY operation:

- The logical and physical data set COPY function supports hierarchical file system (HFS) data sets and zFS data sets. There is no support for copying individual files within an HFS or zFS.
- The COPY function is not supported for SAM compressed extended-format data sets being copied to a non-SMS-managed target.
- The COPY FULL or COPY TRACK commands might invoke ICKDSF to rebuild the VTOC INDEX data set for a target volume. Therefore, users of these commands require the appropriate authority for ICKDSF.
- When you perform a logical or physical COPY operation of a VSAM compressed data set, the target data set allocation must be consistent with the source data set allocation as follows:
 - If the source is an extended-format VSAM KSDS, then the target must be an extended-format VSAM KSDS.
 - If the source is a compressed VSAM KSDS, then the target must be a compressed 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.
 - The target control interval size must be equal to the source.
- If you copy a data set that has an F8/F9 DSCB pair to a volume that does not support F8/F9 DSCBs, the attributes in the F9 DSCB are lost. To retain these extended attributes, the target volumes of the COPY, either SMS or nonSMS, must support F8/F9 DSCBs.

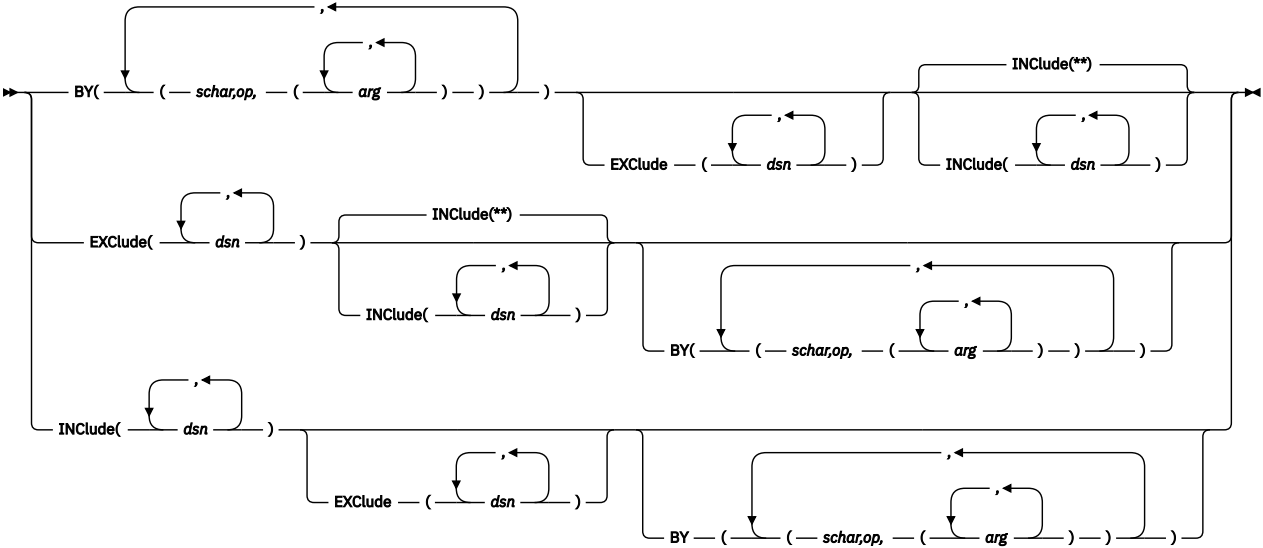
Target data set allocation differs between a physical data set and logical data set copy of non-VSAM data sets. Logical data set copy allocates target data sets according to the amount of used space in the source data set, thereby freeing unused space. Physical data set copy preserves the original size of the source data set. To force unused space to be kept during logical data set copy, the ALLDATA or ALLEXCP keyword must be specified.

COPY DATASET Command Syntax for Logical Data Set

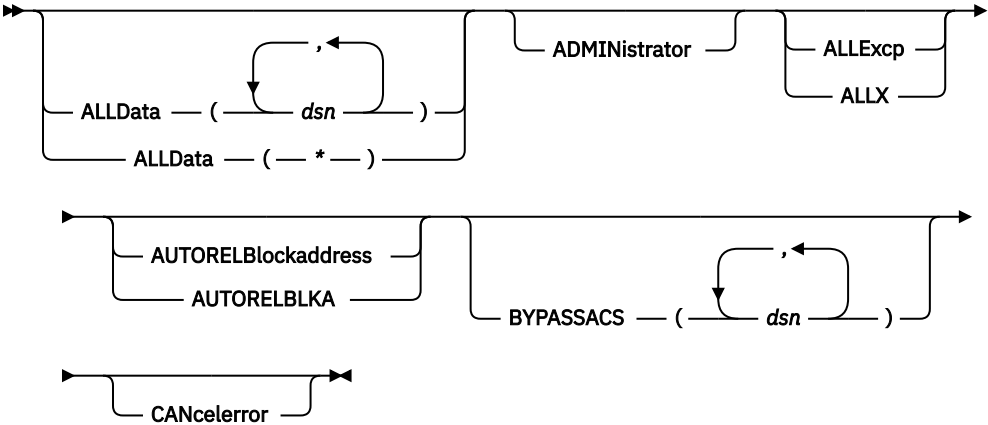


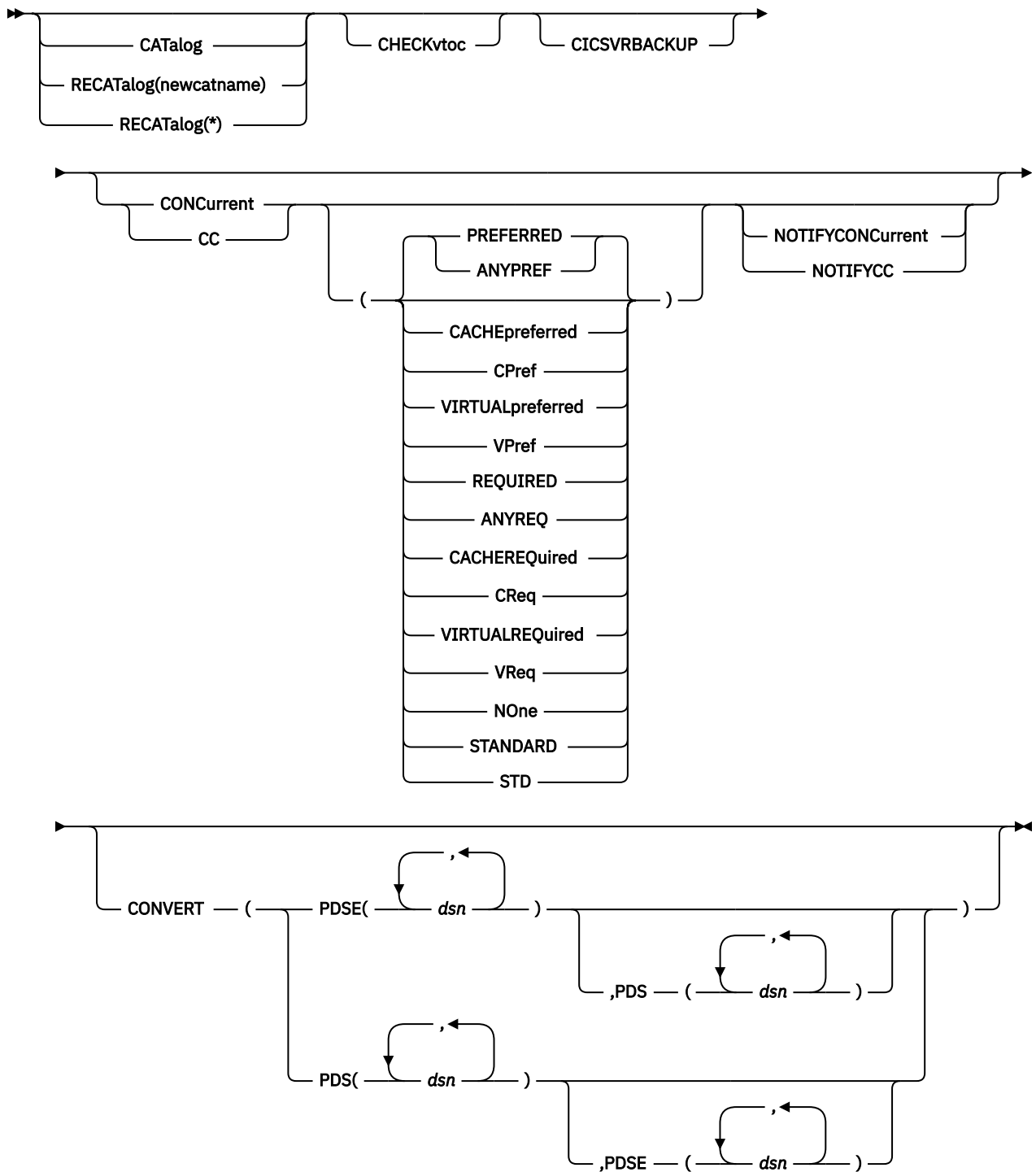
A: Additional Keywords Used for Logical Data Sets

COPY Command

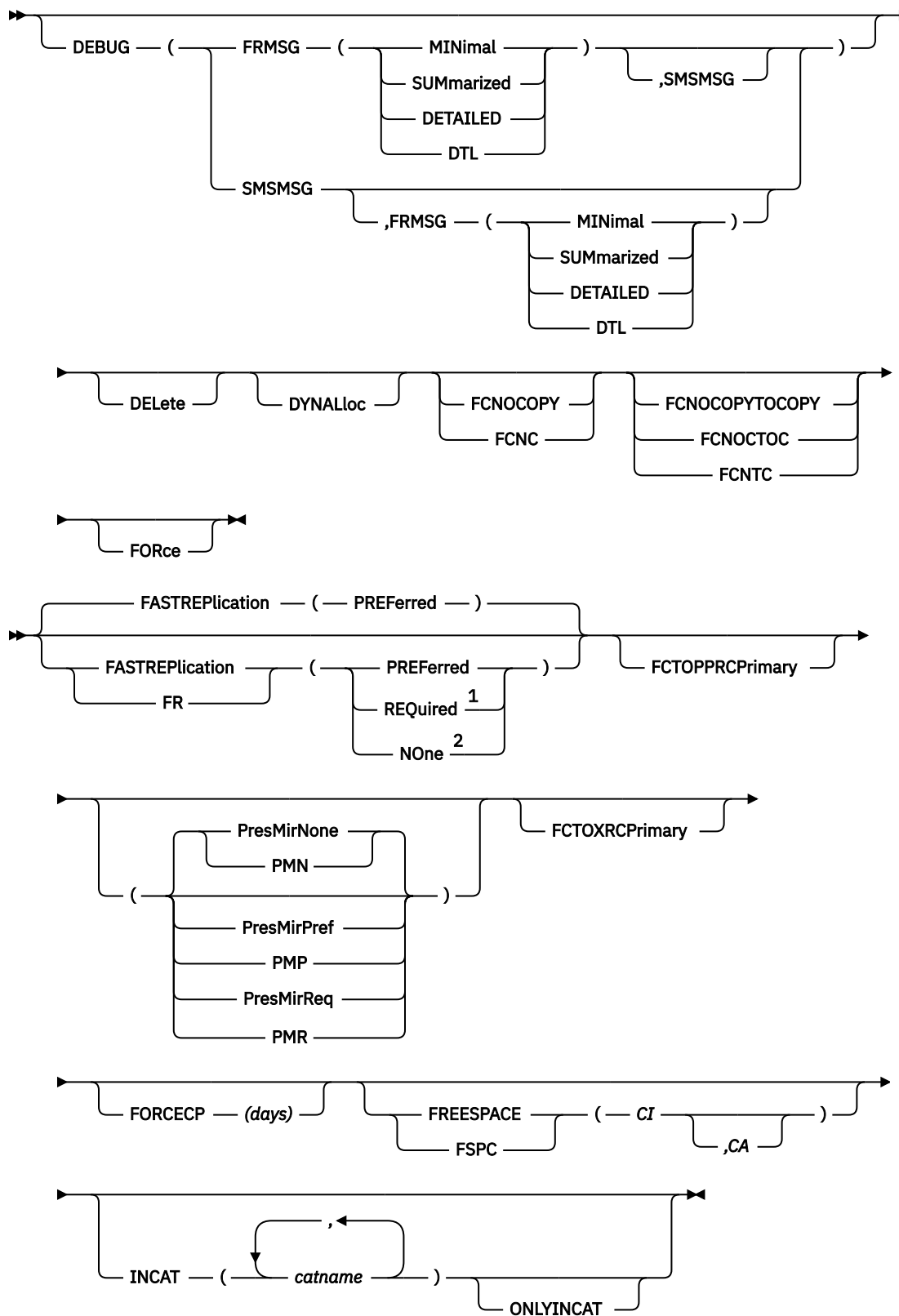


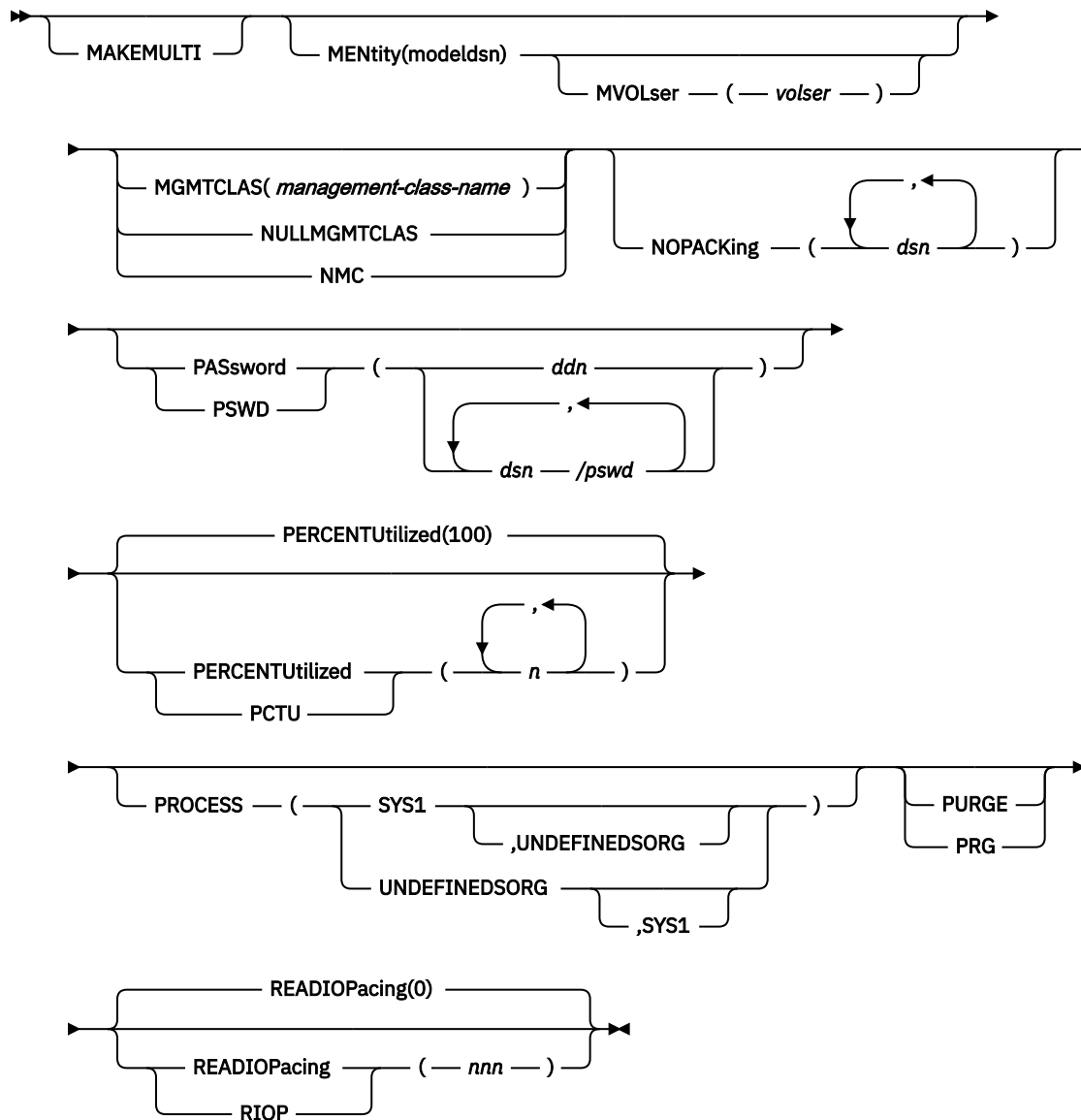
B: Optional Keywords Used for Logical Data Sets

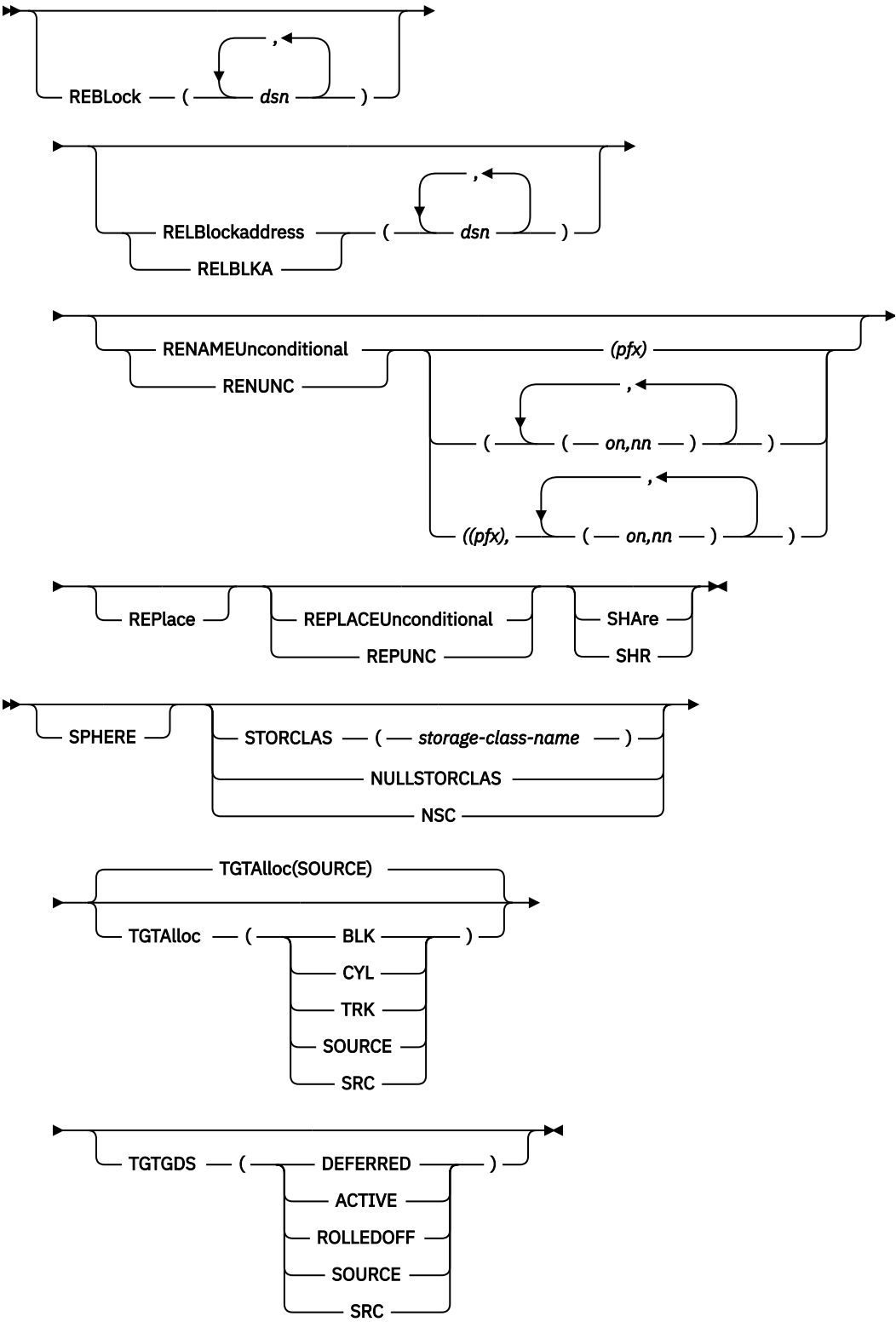


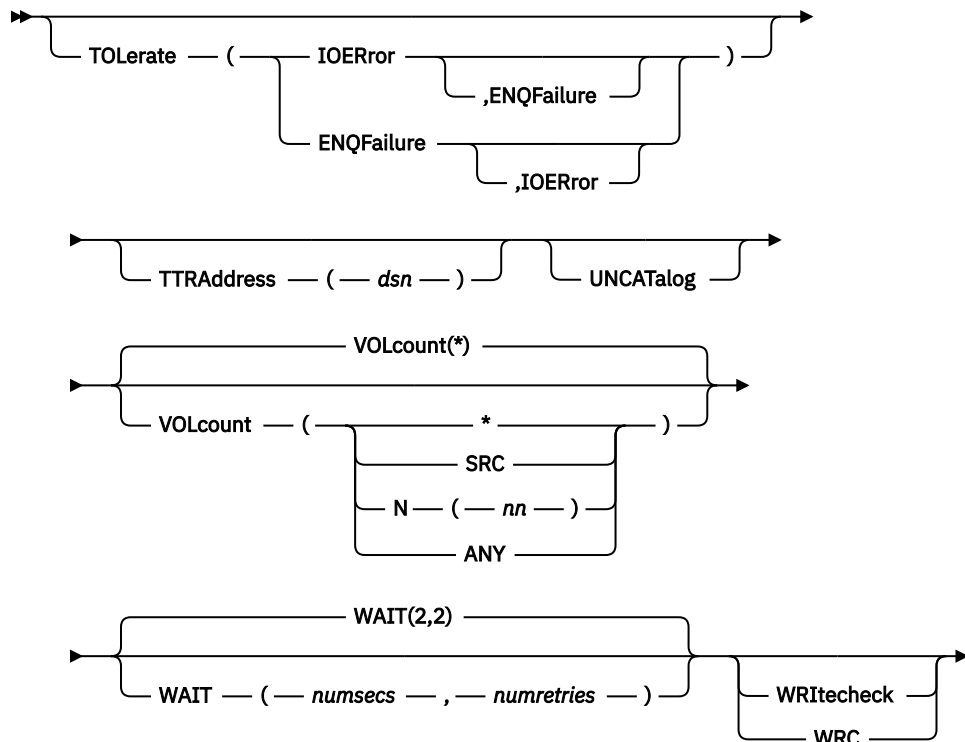


COPY Command







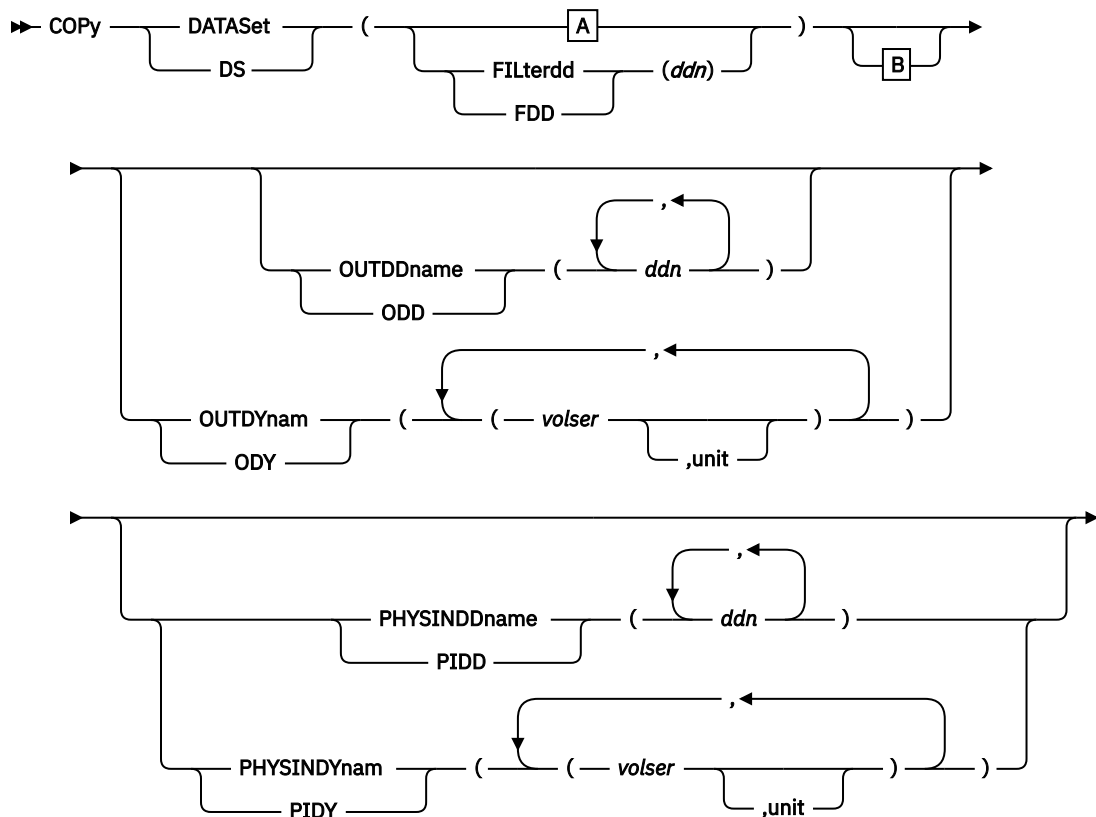


Notes:

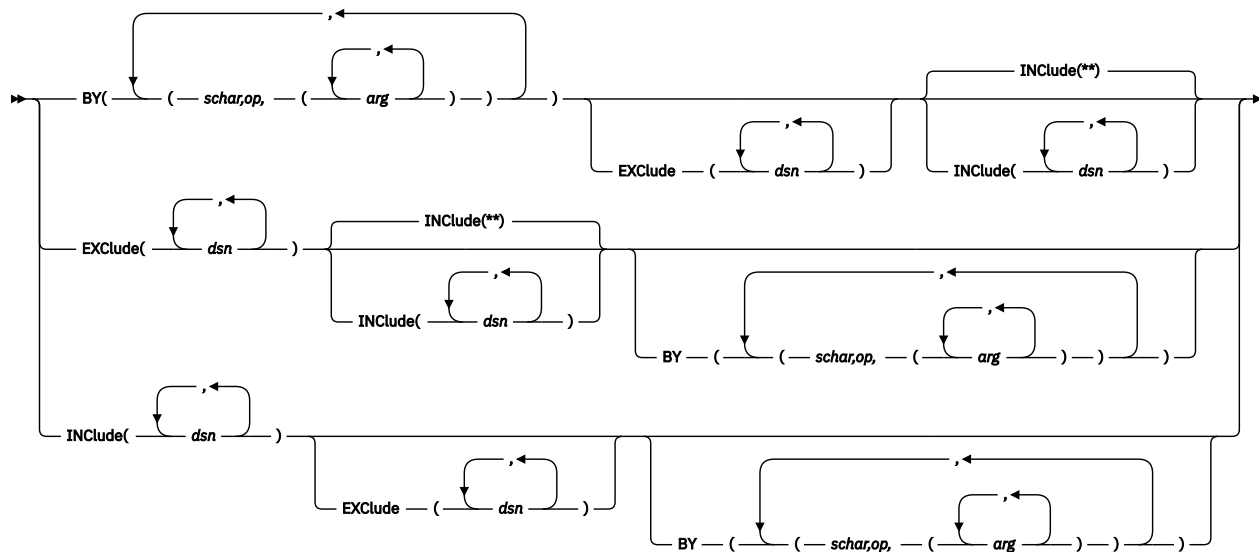
¹ Do not use the FASTREPLication (REQuired) keyword with the CONCURRENT(ANYPREF | ANYREQ | VIRTUALPREF | VIRTUALREQ | CACHEPREF | CACHEREQ) keyword.

² Do not use the FASTREPLication (NONE) keyword with the FCNOCOPY or FCTOPPRCPrimary keywords.

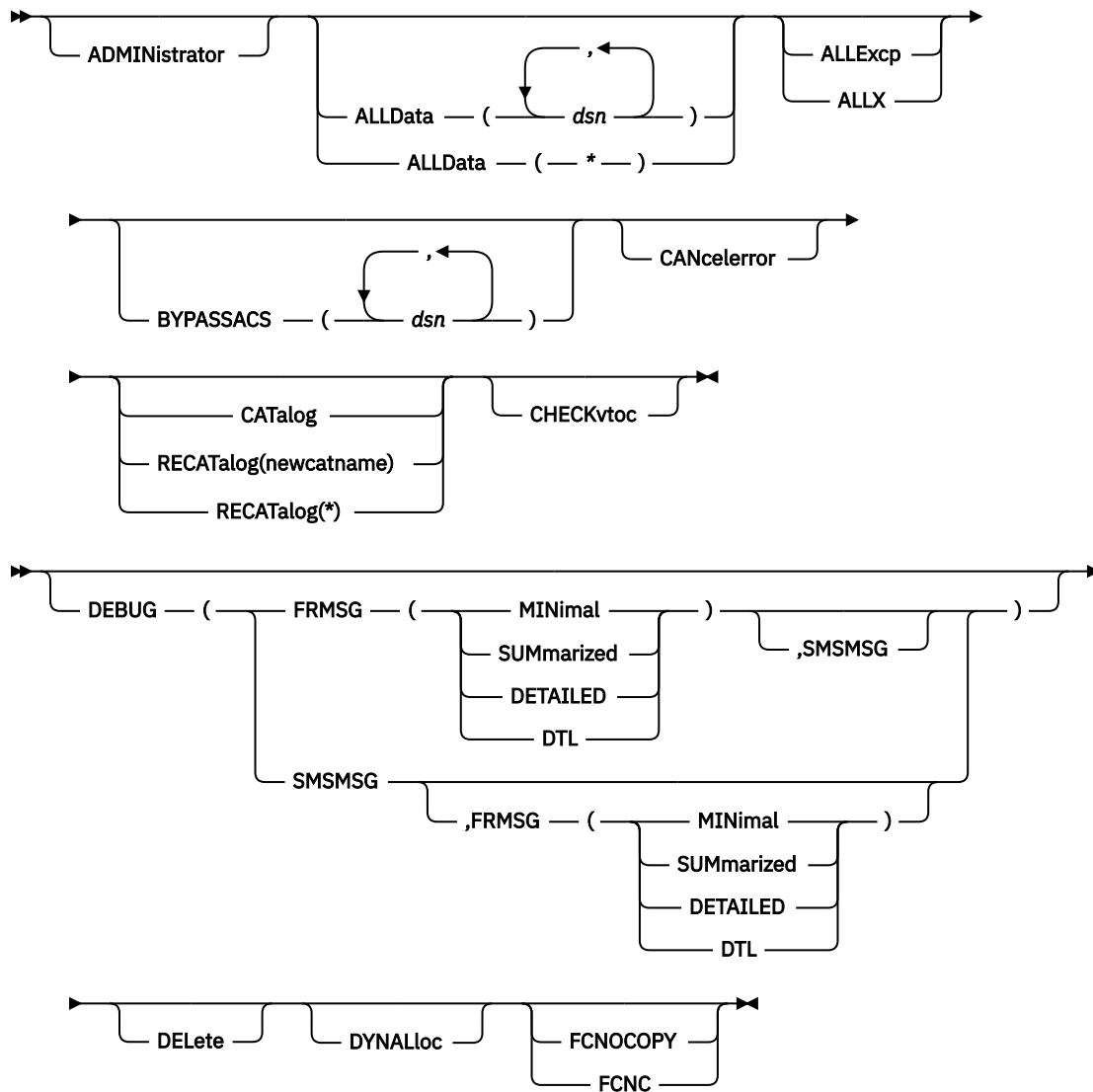
COPY DATASET Command Syntax for Physical Data Set



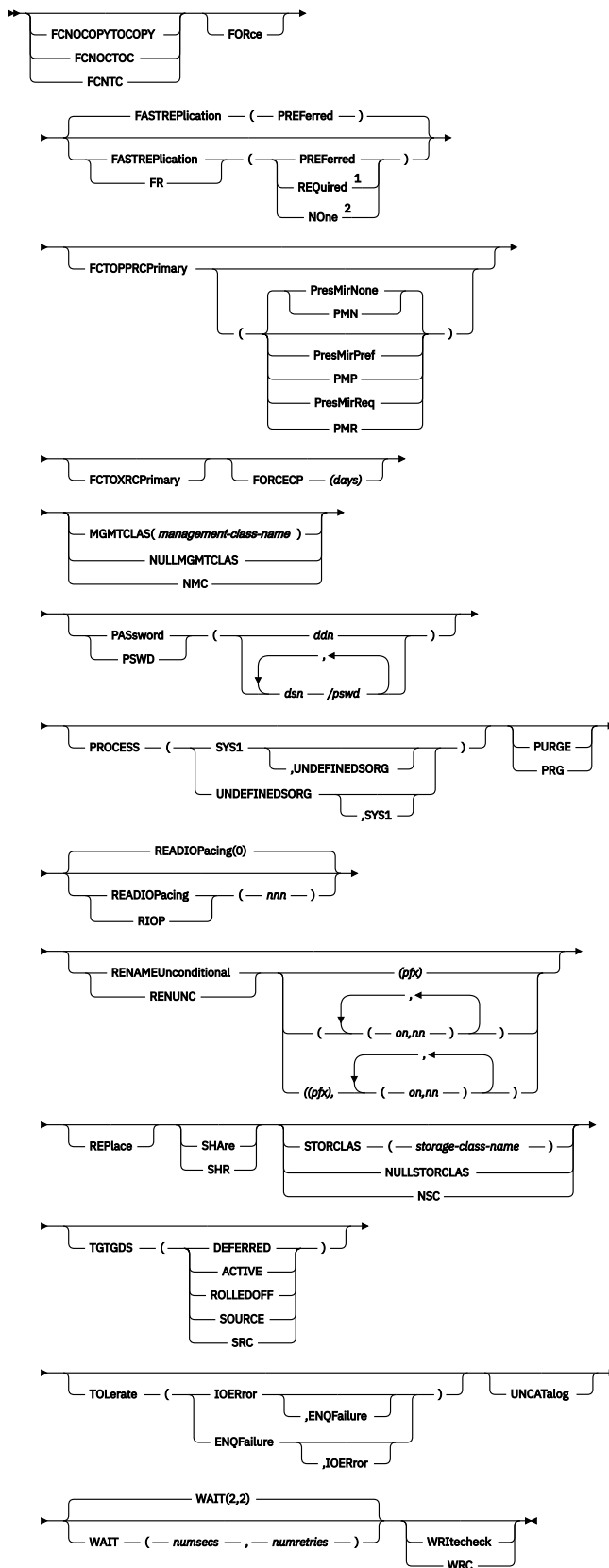
A: Additional Keywords Used for Physical Data Sets



B: Optional Keywords Used for Physical Data Sets



COPY Command



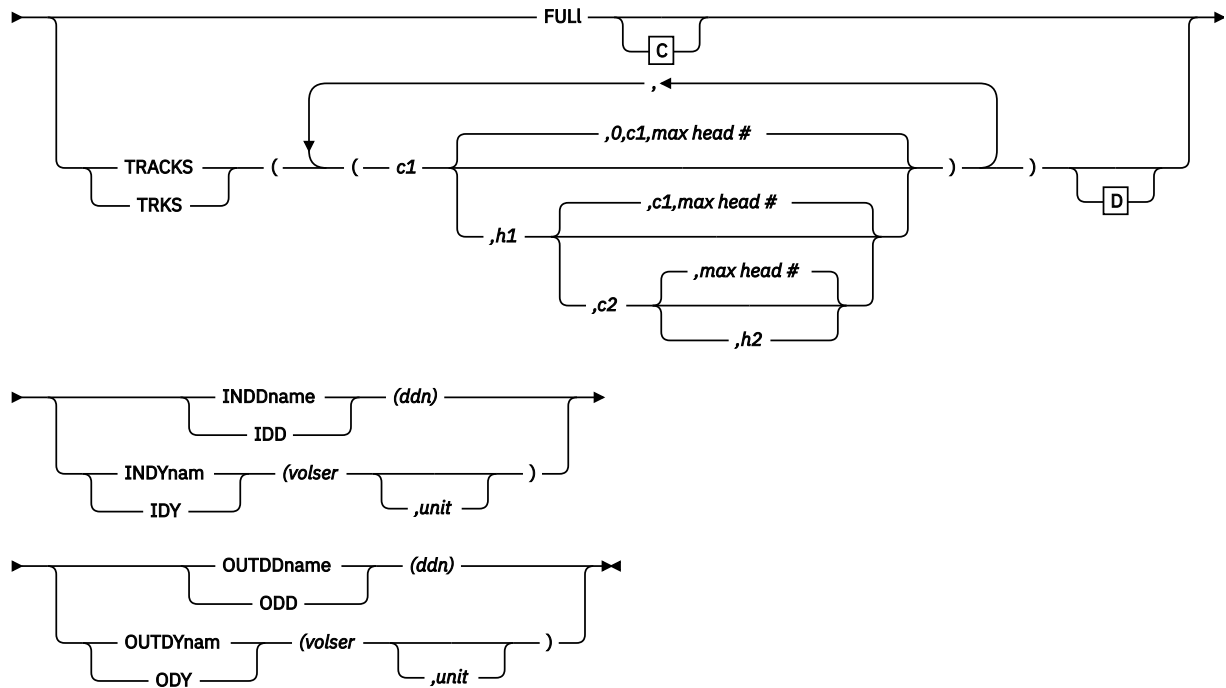
Notes:

¹ Do not use the FASTREplication (REQuired) keyword with the CONCURRENT(ANYPREF | ANYREQ | VIRTUALPREF | VIRTUALREQ | CACHEPREF | CACHEREQ) keyword.

² Do not use the FASTREPLICATION (NONE) keyword with the FCNOCOPY or FCTOPPRCPPrimary keywords.

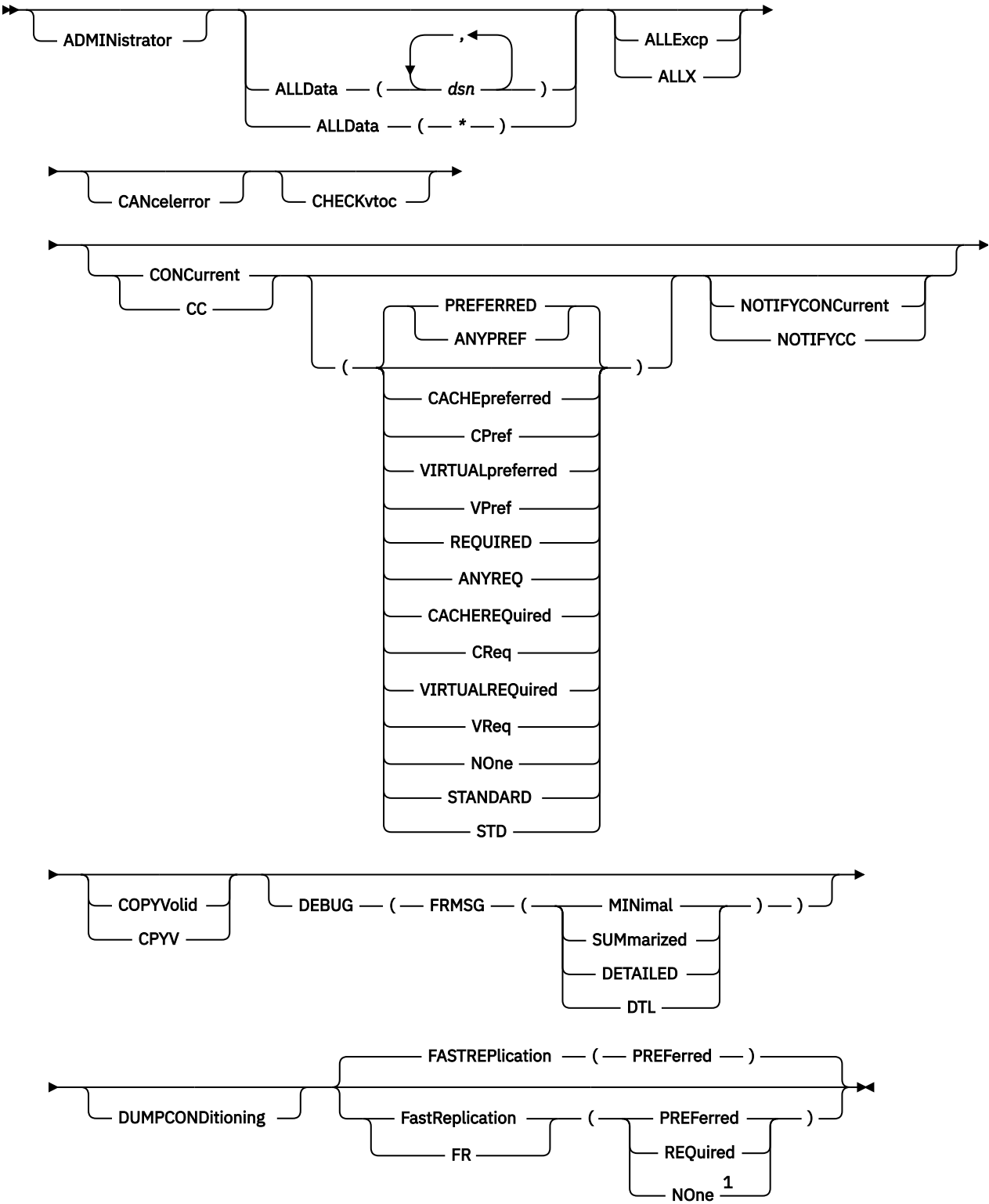
COPY FULL and COPY TRACKS syntax

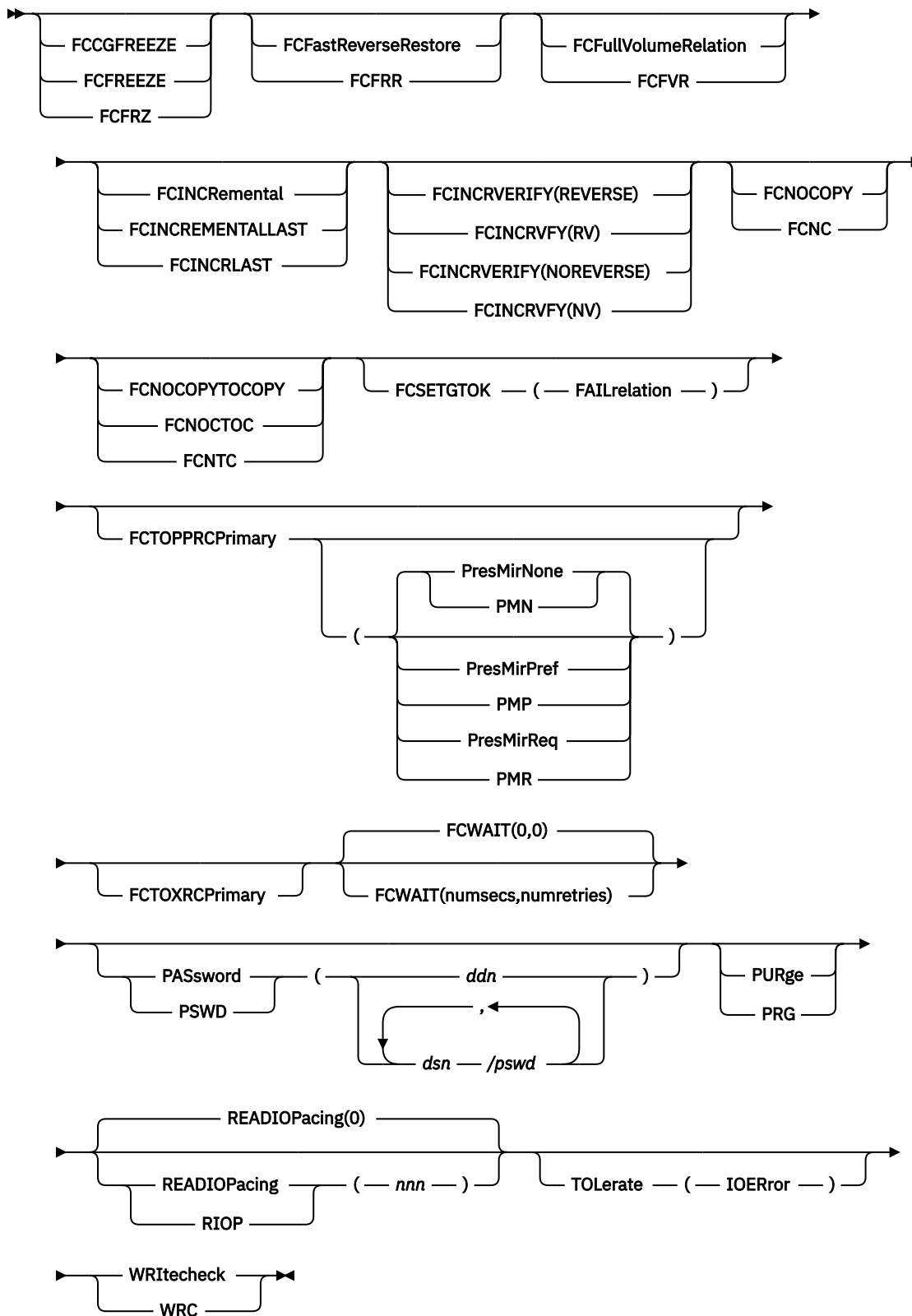
►► COPY ►



C: Optional Keywords with COPY FULL

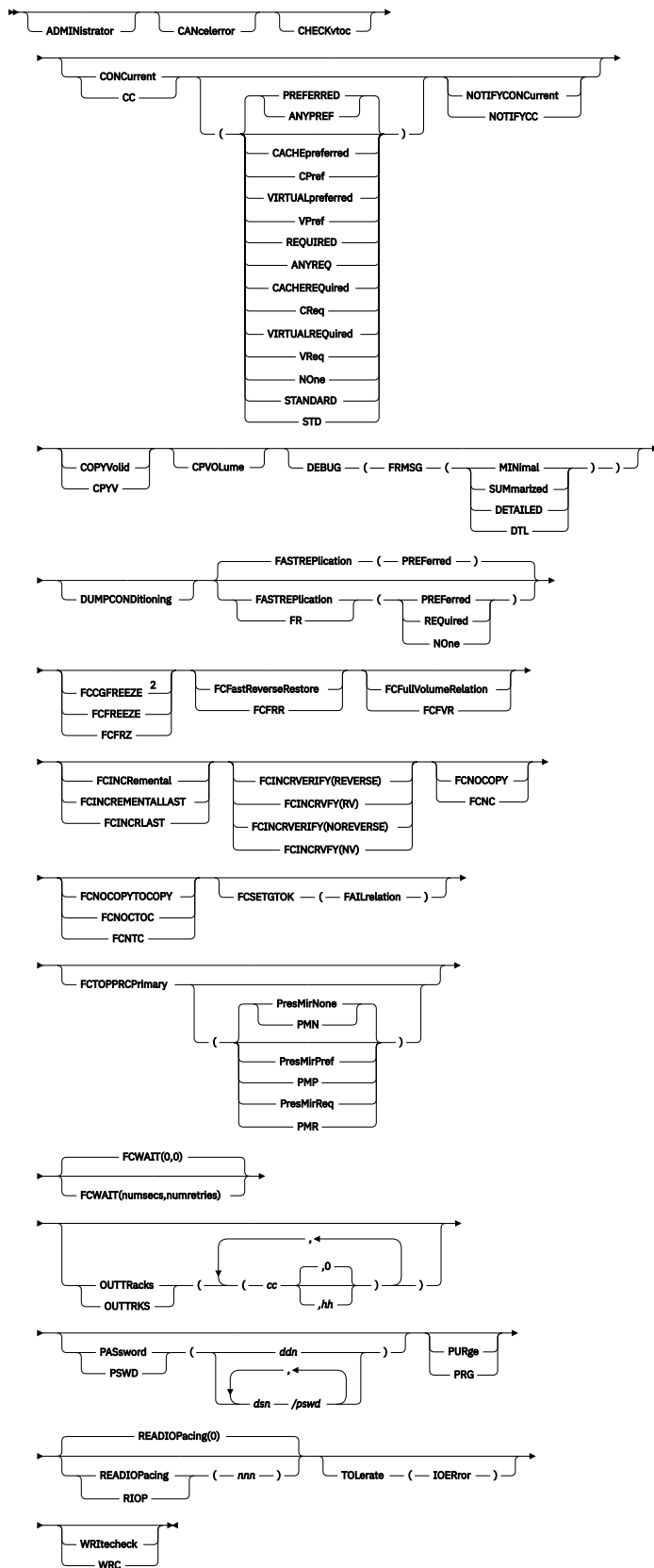
COPY Command





D: Optional Keywords with COPY TRACKS

COPY Command



Notes:

¹ Do not use the FASTREPLICATION (NONE) keyword with the FCFULLVOLUMERELATION, FCNOCOPY, FCSETGTOK, or FCTOPPRCPPRIMARY keywords.

² For COPY TRACKS operations, the FCCGFREEZE, FCINCREMENTAL, and FCINCREMENTALLAST keywords require that the CPVOLUME keyword be specified, too. For more information, see the keyword descriptions.

Explanation of COPY Command Keywords

This section describes the keywords for the COPY command.

ADMINISTRATOR



The ADMINISTRATOR keyword allows you to act as a DFSMSdss authorized storage administrator for the COPY command. For administrators, DFSMSdss bypasses access checking for data sets and catalogs.

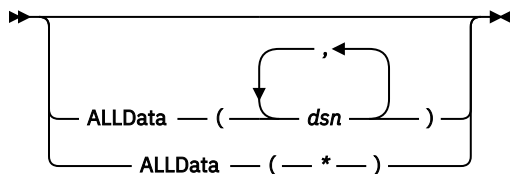
To use the ADMINISTRATOR keyword, all of the following conditions must be true:

- FACILITY class is active.
- Applicable FACILITY-class profile is defined.
- You have READ access to that profile.

If you are not authorized to use the ADMINISTRATOR keyword, the command ends with an error message.

For more details, see [“ADMINISTRATOR keyword” on page 542](#).

ALLDATA



ALLDATA applies to full, logical and physical data set copy operations.

dsn

Specifies the fully qualified name of a data set whose data set organization is physical sequential (PS), physical sequential undefined (PSU), partitioned organization (PO), partitioned organization undefined (POU), or null.

Specify ALLDATA(dsn) or ALLDATA(*) if the data set is not empty, or ALLEXCP if the data set is empty, for the following conditions (this applies to *like* targets only):

- The data set has data beyond the last-used block pointer in the data set's VTOC entry.
- The data set has a null data set organization.
- The data set is the first or intermediate volume of a multivolume data set and has a null data set organization.

JES2/JES3 data sets can have the characteristics specified above, as can CICS journal data sets.

The data set is processed as follows:

- For a full-volume copy, all of the allocated space for the source data set is copied to the target volume.
- For a physical data set copy, all of the allocated space for the part of a data set that resides on the input volume will be copied to the target volume.

- For a data set copy, the function of ALLDATA is dependent upon certain data set characteristics, device characteristics, and other DFSMSdss keywords specified. See [Table 24 on page 366](#) and [Table 25 on page 368](#) for more information.

*** (asterisk)**

Specifies all data sets whose data set organization is PS, PSU, PO, POU, or null and are not empty (the last used block pointer in the data set's VTOC entry is not zero). The data sets are processed as follows:

- For a full-volume copy, all of the allocated space for the source data set is copied to the target volume.
- For a physical data set copy, the allocated space for the piece of a data set that resides on the input volume will be copied to the target volume.
- For a data set copy, the function of this parameter is dependent upon certain data set characteristics, device characteristics, and other DFSMSdss keywords specified. See [Table 24 on page 366](#) and [Table 25 on page 368](#) for more information.

Note:

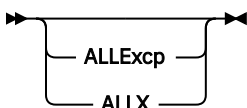
1. When you specify ALLDATA or ALLEXCP for a sequential extended format data set during a logical copy operation DFSMSdss does not retain data beyond the last used block pointer. Also, DFSMSdss allocates the same amount of space for the target data set as the source data set.
2. When you specify ALLDATA for a PDSE data set during a logical copy operation, DFSMSdss does not retain the data that resides in the allocated but unused space. DFSMSdss allocates the same amount of space for the target data set as the source data set.

DFSMSdss determines the amount of space allocated or used for the data set by counting how many tracks have been allocated or used by the data set. For this reason, the allocated space for the target data set may occupy more tracks than the source when going to a different device type.



Attention: Because the unused portion of the data set may or may not be copied, care should be used when specifying the ALLDATA keyword with DELETE. For example, if a data set contains records past the last used block pointer in the data set's VTOC entry that you wish to preserve and you perform a data set copy with ALLDATA and DELETE to an unlike device, these records are not copied to the target, but the source will be deleted upon successful completion of the copy.

ALLEXCP



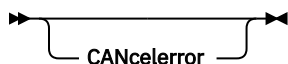
ALLEXCP specifies all data sets whose data set organization is PS, PSU, PO, POU, or null and are empty (the last used block pointer in the data set's VTOC entry is zero). The data sets are processed as follows:

- For a full-volume copy, all of the allocated space for the source data set are copied to the target volume.
- For a physical data set copy operation, the allocated space for the piece of an empty data set on the input volume will be copied to the target volume. If there is no allocated space, but there is an entry on the VTOC, no tracks will be processed and an entry for the data set will be created in the VTOC of the target volume.
- For a data set copy, the function of this keyword is dependent upon certain data set characteristics, device characteristics, and other DFSMSdss keywords specified. See [Table 24 on page 366](#) and [Table 25 on page 368](#) for more information.



Attention: Because all of the allocated space may or may not be copied, use care in specifying the ALLEXCP keyword with DELETE. For example, if a data set contains records that you wish to preserve, but the last block pointer in the data set's VTOC entry is zero and you perform a data set copy with ALLEXCP and DELETE to an unlike device, these records are not copied to the target, but the source will be deleted upon successful completion of the copy.

CANCELERROR



CANCELERROR specifies that the copy task be ended for a permanent read error, or that the copy of a data set is ended for a write error.

- Permanent read error, such as a data check:

If CANCELERROR is specified, the copy task is ended. If this keyword is not specified, the track in error is not copied and the copy continues. Only the data set receiving the error is ended, and the DFSMSdss copy function continues to process any subsequent data sets.

- Write error, such as an invalid track format:

For data set copy, processing of the data set ends and the target data set is deleted. The copy operation continues with the next data set. For full volume and tracks copy, processing for the volume ends. Subsequent tracks are not processed.

DFSMSdss allows you to change this default operation. A patch byte is provided to allow you to change the default handling of invalid tracks created during COPY processing.

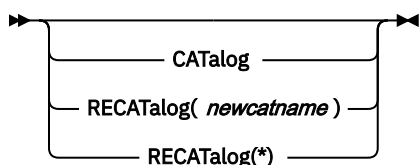
During copy operations in which a utility performs the copy, DFSMSdss ignores this keyword. CANCELERROR has no effect on the following types of errors on a DASD volume:

- Equipment check
- Command reject
- Intervention required
- Busout parity

This keyword may be used in conjunction with CHECKVTOC to specify whether or not the operation is to continue in the event of terminating VTOC errors found during VTOC checking. Refer to CHECKVTOC keyword.

For more information about the handling of invalid tracks, see [Chapter 10, “Diagnosing problems in DFSMSdss operations,”](#) on page 155.

CATALOG



CATALOG specifies that on a data set copy operation, DFSMSdss is to catalog data sets that it allocates. For a *logical* copy operation, CATALOG instructs DFSMSdss to catalog data sets that it allocates. For a *physical* copy operation, CATALOG instructs DFSMSdss to catalog the non-VSAM single volume data sets that it allocates. An IDCAMS DEFINE RECATALOG must be used to catalog the VSAM data sets after the physical copy. If the CATALOG keyword is not specified, single volume non-VSAM target data sets will be uncataloged as well.

CATALOG

catalogs the target data set as determined by the standard catalog search order. This is the default for VSAM, multivolume data sets, and SMS-managed data sets.

Note:

1. If the CATALOG keyword was specified, but the RENAMEUNCONDITIONAL or UNCATALOG keywords were not specified, an ADR385E message will be issued because two data sets with the same name cannot be cataloged in the standard order of search at the same time.

2. The CATALOG keyword is ignored for preallocated target data sets.

RECATALOG(*newcatname*)

catalogs the target data set in the *newcatname* catalog. If you do not specify the *RECATALOG(newcatname)* keyword, single volume non-VSAM target data sets remain uncataloged as well.

RECATALOG(*)

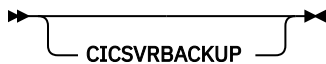
catalogs the target data set in the same catalog that points to the source data set. If the source data set was not cataloged, the new data set is not cataloged either. After DFSMSdss determines the catalog status of the data set and is changed by other means outside of DFSMSdss, the original catalog status is used. If you do not specify the *RECATALOG(*)* keyword, single volume non-VSAM target data sets remain uncataloged as well.

Note:

1. Be careful when using the *RECATALOG(newcatname)* keyword because the target data set may be cataloged outside of the standard order of search.
2. The CATALOG or RECATALOG operation fails if the target data set is already cataloged in the same catalog and DELETE, RENAMEU, or UNCATALOG is not specified. The RECATALOG keyword is ignored for SMS-managed targets.
3. An alternate index (AIX) is always cataloged in the same catalog as its associated base cluster. If the base cluster is recataloged, the AIX is recataloged.
4. If you omit the CATALOG or RECATALOG keyword for a single volume, non-VSAM, non-SMS-managed data set, the target data set is uncataloged.
5. The CATALOG and RECATALOG keywords are ignored for preallocated data sets.
6. If the *RECATALOG(newcatname)* or *RECATALOG(*)* keyword is specified, but RENAMEUNCONDITIONAL or UNCATALOG is not specified, message ADR385E will be issued if the *newcatname* catalog is in the standard order of search since two data sets with the same name cannot be cataloged in the standard order of search.

CHECKVTOC

CHECKVTOC specifies that a VTOC analysis of the source volume be performed during copy processing. In the event of terminating VTOC errors found during analysis, operation continues unless the CANCELerror keyword is specified. CHECKVTOC is ignored if CPVOLUME is also specified.

CICSVRBACKUP

CICSVRBACKUP specifies that DFSMSdss create backups for use by CICSVR for a data set copy operation. DFSMSdss notifies the CICSVR server address space when a CICSVR backup is made for a VSAM base cluster. This notification enables CICSVR to manage backups that are made by DFSMSdss.

CICSVR provides DFSMSdss with a new name for each VSAM base cluster that is to be copied when CICSVRBACKUP is specified. DFSMSdss uses the CICSVR-generated new name instead of the one that is specified in the RENAMEUNCONDITIONAL keyword.

Note:

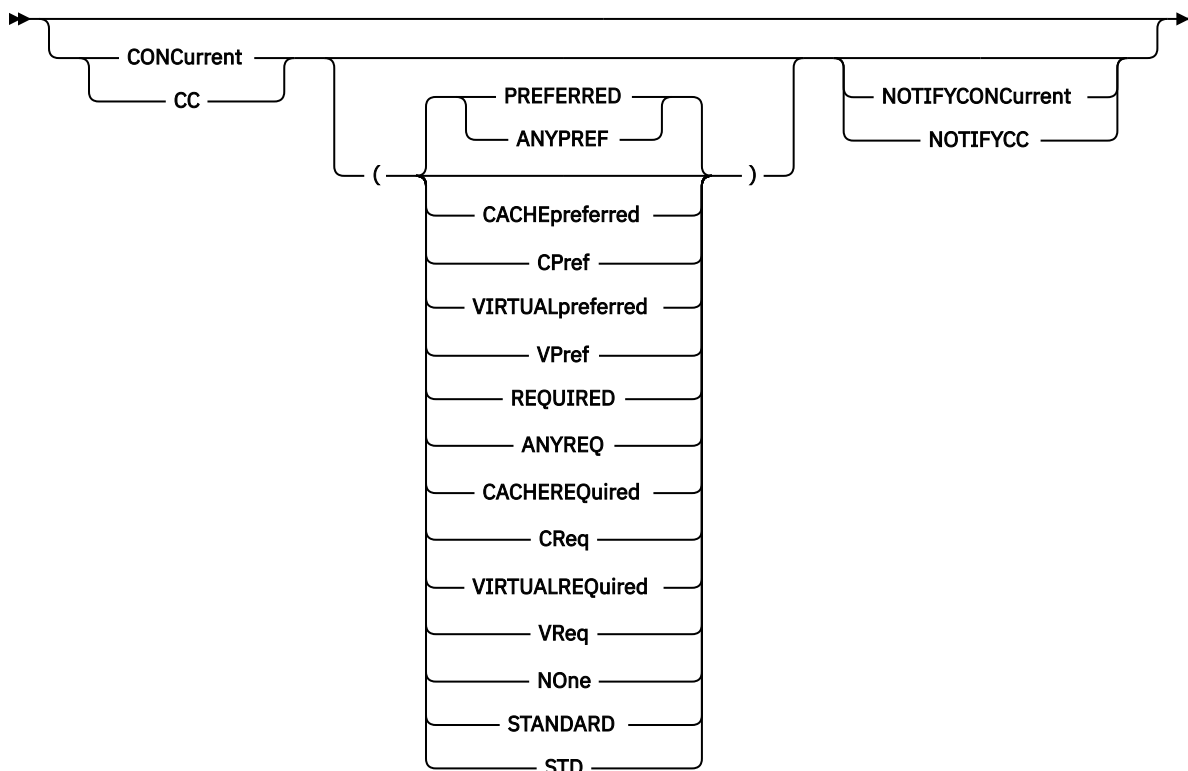
1. CICSVRBACKUP is intended to be used with CICSVR. The minimum required CICSVR release is z/OS 3.1. To use CICSVRBACKUP, the CICSVR server address space must be active.
2. CICSVRBACKUP applies to COPY DATASET for logical data set processing only and does not apply to RLS catalogs.

3. CICSVR manages VSAM base clusters and RLS user catalogs that are backed up using the DFSMSdss COPY command. DFSMSdss COPY fails the processing of alternate indexes when you specify the CICSVRBACKUP keyword. Because CICSVR removes reusable alternate indexes (AIX) from the upgrade set before recovery and rebuilds the reusable AIXs after recovery, you need not copy the alternate indexes. When DFSMSdss copies non-VSAM data sets, the CICSVRBACKUP keyword is ignored.
4. CICSVRBACKUP cannot be specified with the SPHERE or DELETE keyword.
5. You must specify the RENAMEUNCONDITIONAL keyword when you specify the CICSVRBACKUP keyword. The use of RENAMEU must follow the DFSMSdss syntax rules. However, be aware that DFSMSdss uses the CICSVR-generated new name instead of the name that you specify.

Recommendation: To avoid confusion or frustration, you can specify the RENAMEU keyword as RENAMEU(**,CICSVR.**).

For more information about CICSVR-generated new name, its naming convention, and required RENAMEU specifications, see *CICSVR Implementation Guide*.

CONCURRENT



The CONCURRENT keyword specifies that the data is to be processed with concurrent copy except when CONCURRENT(STANDARD | NO) is specified. You can specify one of the following optional sub-keywords to indicate the type of concurrent copy to be used and whether DFSMSdss can use standard I/O when concurrent copy could not be used or has failed.

ANYPREFERRED or PREFERRED

Specifies that data is to be processed with concurrent copy. Virtual concurrent copy is attempted first, if the storage subsystem on which the data resides is capable of it and working-space data sets have been defined. Otherwise, cache-based concurrent copy is attempted if the storage subsystem is capable of it. If neither type of concurrent copy is possible or both fail, the data is processed with standard I/O. PREFERRED is the default if you specify the CONCURRENT keyword without a sub-keyword.

ANYREQUIRED or REQUIRED

Specifies that data is to be processed with concurrent copy. Virtual concurrent copy is attempted first, if the storage subsystem on which the data resides is capable of it and working-space data sets have been defined. Otherwise, cache-based concurrent copy is attempted, if the storage subsystem is capable of it. If neither type of concurrent copy is possible or both fail, the data is not processed.

CACHEPREFERRED

Specifies that data is to be processed with cache-based concurrent copy. If cache-based concurrent copy cannot be used or fails, the data is processed with standard I/O. DFSMSdss does not attempt to use virtual concurrent copy.

CACHEREQUIRED

Specifies that data is to be processed with cache-based concurrent copy. If cache-based concurrent copy cannot be used or fails, the data is not processed. DFSMSdss does not attempt to use virtual concurrent copy or standard I/O.

STANDARD or NONE

Specifies that data is to be processed with standard I/O as if the CONCURRENT keyword was not specified.

VIRTUALPREFERRED

Specifies that data is to be processed with virtual concurrent copy. If virtual concurrent copy cannot be used or fails, the data is processed with standard I/O. DFSMSdss does not attempt to use cache-based concurrent copy.

VIRTUALREQUIRED

Specifies that data is to be processed with virtual concurrent copy. If virtual concurrent copy cannot be used or fails, the data is not processed. DFSMSdss does not attempt to use cache-based concurrent copy or standard I/O.

For a logical data set copy operation, you can also specify the NOTIFYCONCURRENT keyword, as follows:

NOTIFYCONCURRENT

Specifies that DFSMSdss is to issue an informational message for every data set that is successfully included in the concurrent copy operation. If you do not specify NOTIFYCONCURRENT, DFSMSdss issues messages only for data sets that are not successfully included in the concurrent copy operation.

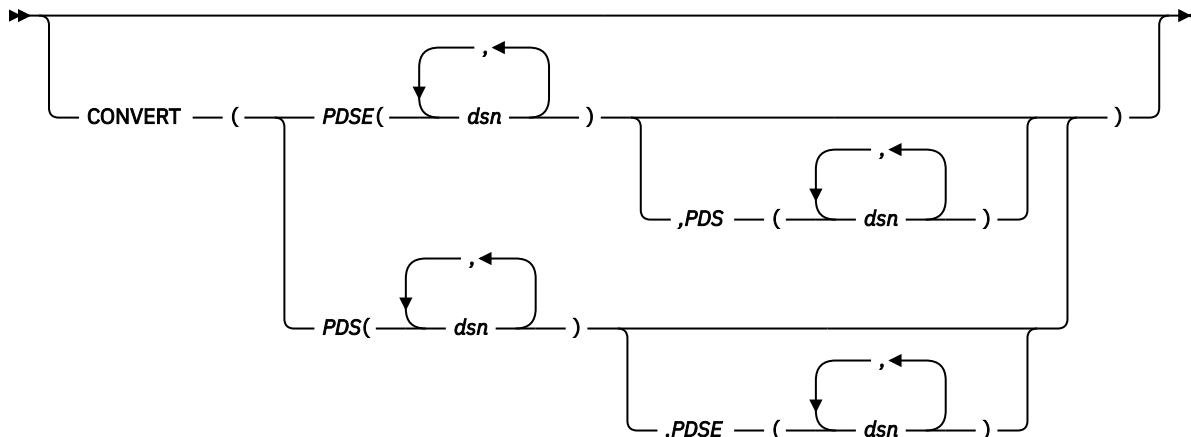
Note:

1. Do not specify NOTIFYCONCURRENT with CONCURRENT(STANDARD | NONE).
2. You cannot use the CONCURRENT keyword with the DELETE, UNCATALOG, because after the concurrent copy operation starts, the original data might still be updated.
3. You cannot use the concurrent copy option with FASTREPLICATION(REQUIRED) keyword.
4. The use of concurrent copy and virtual concurrent copy with the DFSMSdss COPY command is controlled by the RACF FACILITY class profile, STGADMIN.ADR.COPY.CNCURRNT.
5. Cache-based and virtual concurrent copy operations are not affected if you specify the FASTREPLICATION(PREFERRED) keyword for the COPY command. If you specify both FASTREPLICATION(PREFERRED) and CONCURRENT keywords, DFSMSdss attempts to use fastreplication first.
6. Cache-based and virtual concurrent copy operations are not affected by RACF FACILITY class profile STGADMIN.ADR.COPY.FLASHCPY.

For help with determining concurrent copy storage requirements, see [“Concurrent copy storage requirements” on page 60](#).

For more information about virtual concurrent copy and working space data sets, see [“Performance considerations” on page 59](#) and [z/OS DFSMS Advanced Copy Services](#).

CONVERT

**CONVERT(PDSE(dsn))**

Specifies that the PDSs that are listed in the *dsn* be converted to PDSE

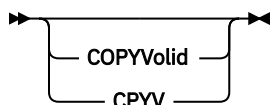
CONVERT(*PDS(dsn)*)

Specifies that the PDSEs that are listed in the *dsn* be converted to PDS

Note:

1. When a PDSE version 2 is converted to a PDS all PDSE version attributes are lost, including member generations.

COPYVOLID



COPYVOLID specifies that the volume serial number (VOLID) from the input DASD volume is to be copied to the output DASD volume. This applies to full copy operations and to tracks copy operations if track 0 (zero) is copied.

When the volume serial number on a DASD volume is changed, the operator is notified. The operating system then initiates either:

- A demount if there is another volume with the same serial number, or
- A mount to get the volume with the new serial number mounted.

This might change the mount attributes of the volume. You should exercise operating precautions if there are two or more processors sharing the same DASD volume.

When the volume serial number is changed by using a COPYVOLID keyword or when both the dumped volume and the restored volume have different serial numbers, profiles are not built for the RACF-protected data sets on the restored volume or for the RACF DASDVOL for RACF-protected DASD volumes.

Note:

1. DFSMSdss requires the COPYVOLID keyword for a full-volume copy operation of an SMS-managed input volume—unless you specify the DUMPCONDITIONING keyword.
2. When the volume serial number is changed by using a COPYVOLID keyword, profiles are not built for the RACF-protected data sets on the target volume or for the RACF DASDVOL for the RACF-protected DASD volume. When the volume serial number on a DASD volume is changed, the operator is notified. The operating system then initiates a deformat of the volume.

3. Exercise caution using COPYVOLID in a multiple task job step when two or more of the tasks are using the same output volume. If the output volume is made unavailable by the first task, all succeeding tasks that use the same output volume fail.
4. COPYVOLID cannot be performed if there are permanent I/O errors or if CANCELERROR is specified. If TOLERATE(IOERROR) is honored, however, COPYVOLID is performed.
5. You cannot use the COPYVOLID keyword with the DUMPCONDITIONING keyword.

CPVOLUME



CPVOLUME specifies that the input and output volumes are VM-format volumes and that the OS-compatible VTOCs must begin on track zero, record five. You must specify the track range to be copied with the TRACKS keyword, as the OS-compatible VTOCs do not describe the extents of any data on the volume. You must also specify the ADMINISTRATOR keyword with CPVOLUME because DFSMSdss cannot check access authorization for VM data.

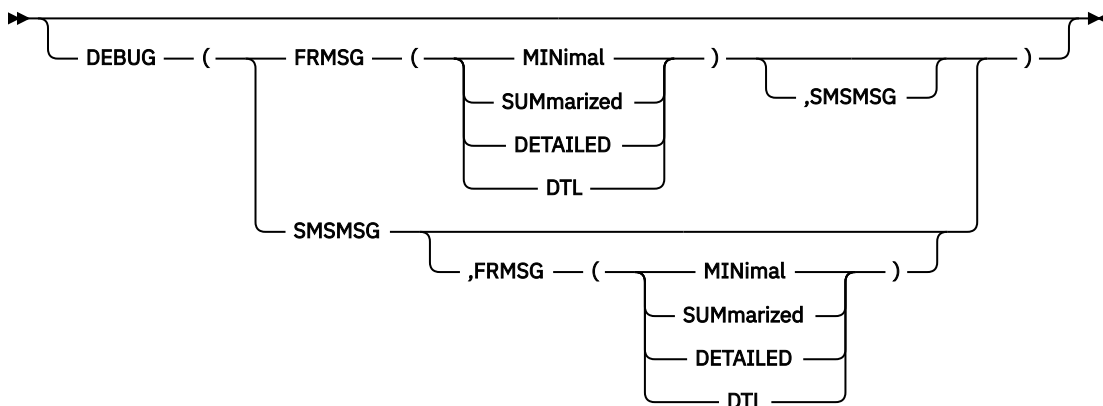
DATASET



DATASET specifies a data set copy operation using filtering. See [Chapter 16, “DFSMSdss filtering—choosing the data sets you want processed,”](#) on page 255 for an explanation of the filtering process used. Unless ALLDATA or ALLEXCP is specified, only *used tracks* are copied for sequential and partitioned data sets and for data sets with a data set organization that is null (for example, JES2/JES3 data sets). If the free space map in the VTOC is invalid, all tracks for the data set are copied.

Note: Either the FILTERDD, INCLUDE, EXCLUDE, or BY keyword must be specified when data set is selected.

DEBUG



You can use DEBUG as a diagnostic tool. When you specify the FRMSG subkeyword, DFSMSdss issues messages that explain why you cannot use fast replication or Preserve Mirror operation during COPY processing. The DEBUG(FRMSG) keyword overrides the DEBUG=FRMSG parameter that is specified in the JCL EXEC statement. For Preserve Mirror operations, the DEBUG(FRMSG) keyword might not have an effect if the FlashCopy target is not a PPRC Primary device.

Specify DEBUG(SMSMSG) to instruct DFSMSdss to display ACS WRITE statements and SMS-specific allocation messages to the job output.

Specify DEBUG(FRMSG) with one of the following sub-keywords:

FRMSG(MINIMAL)

Specifies that DFSMSdss is to issue a message with a minimal level of information. The following are examples of messages that are issued when you use this keyword:

Example 1: Data set copy

```
ADR948I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
DATA SET TEST.SRC.KSDS1 BECAUSE THE TARGET DEVICES DO NOT PROVIDE
COMPATIBLE DATA SET FAST REPLICATION FUNCTIONS
```

Example 2: Data set copy

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED
FOR DATA SET TEST.SRC.KSDS1, RETURN CODE 3
```

Return code 3 indicates that one or more source devices are not eligible for fast replication at this time.

Example 3: Data set copy

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED
FOR DATA SET TEST.SRC.KSDS1, RETURN CODE 15
```

Return code 15 indicates that for the SMS allocation, target volumes that would allow fast replication to be used could not be selected.

Example 4: Full volume or tracks copy

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED
FOR VOLUME SRCV01, RETURN CODE 1
```

Return code 1 indicates that the source device is not capable of fast replication.

Guideline: DFSMSdss suppresses SMS allocation messages regarding fast replication during a data set copy operation when you specify the DEBUG(FRMSG(MINIMAL)) keyword.

FRMSG(SUMMARIZED)

Specifies that DFSMSdss is to issue an informational message with summary information. When applicable, summary information regarding ineligible volumes is provided in the message text. The following examples show the messages that are issued when you use this keyword:

Example 1: Data set copy

```
ADR948I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED
FOR DATA SET TEST.SRC.KSDS1 BECAUSE THE TARGET DEVICES DO NOT PROVIDE
COMPATIBLE DATA SET FAST REPLICATION FUNCTIONS
    2 VOLUMES SUPPORT DATA SET FLASHCOPY
    1 VOLUME SUPPORTS SNAPSHOT
    3 VOLUMES DO NOT SUPPORT ANY TYPE OF DATA SET FAST REPLICATION
```

Example 2: Data set copy

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR DATA SET
TEST.SRC.KSDS1, RETURN CODE 3
    1 VOLUME WAS REJECTED FOR QFRVOLS REASON CODE 7 - VERSION 1 FC RELATION
      EXISTS
    2 VOLUMES WERE REJECTED FOR QFRVOLS REASON CODE 8 - MAX ESS FC
      RELATIONS
    1 VOLUME WAS REJECTED FOR QFRVOLS VOLUME REASON CODE CA - BOUNDARY
      EXCEPTION
```

Example 3: Data set copy, target data set is non-SMS-managed


```

ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
DATA SET TEST.SRC.KSDS1, RETURN CODE 14
  1 VOLUME WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 7 - VERSION 1 FC
    RELATION EXISTS
  2 VOLUMES WERE REJECTED FOR QFRVOLS VOLUME REASON CODE 8 - MAX ESS
    FC RELATIONS
  2 VOLUMES WERE REJECTED FOR QFRVOLS VOLUME REASON CODE C9 - FLASHCOPY
    NOT SUPPORTED
  1 VOLUME WAS REJECTED FOR DFSMSDSS REASON CODE 1 - INSUFFICIENT SPACE
  1 VOLUME WAS REJECTED FOR DFSMSDSS REASON CODE 2 - NO FREE DSCB
    IN THE VTOC
  2 VOLUMES WERE REJECTED FOR DFSMSDSS REASON CODE 3 - VOLUME IS SMS
    MANAGED
  1 VOLUME WAS REJECTED FOR DFSMSDSS REASON CODE 4 - LSPACE MACRO FAILED
    WHILE CALCULATING FREE SPACE
  1 VOLUME WAS REJECTED FOR DFSMSDSS REASON CODE 8 - DADSM FAILURE
    OCCURRED WHILE ALLOCATING THE DATA SET ON THE VOLUME

```

Example 4: Data set copy, target data set is SMS-managed Data set copy, target data set is non-SMS-managed

```

ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
DATA SET TEST.SRC.KSDS1, RETURN CODE 15
IGD17330I DATA SET TEST2.TGT.KSDS1 WAS ALLOCATED ON VOLUME(S) WHICH ARE
    NOT ELIGIBLE FOR FAST REPLICATION. PREFERRED FAST REPLICATION
    WAS SPECIFIED BY CALLER
IGD17290I THERE WERE 3 CANDIDATE STORAGE GROUPS OF WHICH THE FIRST 3
    WERE ELIGIBLE FOR VOLUME SELECTION. THE CANDIDATE STORAGE
    GROUPS WERE: SG1, SG2, SG3
IGD17267I THE FOLLOWING 1 CANDIDATE STORAGE GROUPS WERE INELIGIBLE FOR
    PREFERRED FAST REPLICATION BECAUSE THEY DID NOT HAVE A SUFFICIENT
    NUMBER (2) OF ELIGIBLE FAST REPLICATION VOLUMES: SG3
IGD17268I 2 VOLUMES WERE NOT USED FOR FAST REPLICATION BECAUSE THE SMS VOLUME
    STATUS WAS DISABLED
IGD17268I 2 VOLUMES WERE NOT USED FOR FAST REPLICATION BECAUSE THEY WERE
    NOT ONLINE
IGD17268I 6 VOLUMES WERE NOT USED FOR FAST REPLICATION BECAUSE OF FLASHCOPY
    NOT SUPPORTED - ANTRQST QFRVOLS VOLUME RSN(201)
IGD17268I 2 VOLUMES WERE NOT USED FOR FAST REPLICATION BECAUSE OF BOUNDARY
    EXCEPTION - ANTRQST QFRVOLS VOLUME RSN(202)
IGD17268I 2 VOLUMES WERE NOT USED FOR FAST REPLICATION BECAUSE OF XRC SRC
    CURRENTLY ACTIVE - ANTRQST QFRVOLS VOLUME RSN(5)
IGD17268I 1 FR-ELIGIBLE VOLUMES WERE NOT USED FOR FAST REPLICATION BECAUSE
    STORAGE GROUP HAS INSUFFICIENT FAST REPLICATION VOLUMES
IGD17268I 4 FR-ELIGIBLE VOLUMES WERE NOT USED FOR FAST REPLICATION BECAUSE
    THEY DID NOT HAVE SUFFICIENT SPACE
IGD17269I 2 NON-FR VOLUMES WERE REJECTED BECAUSE THE SMS VOLUME STATUS WAS
    DISABLED
IGD17269I 2 VOLUMES WERE REJECTED BECAUSE THEY WERE NOT ONLINE
IGD17269I 1 VOLUMES WERE REJECTED BECAUSE OF A DADSM FAILURE
IGD17269I 5 VOLUMES WERE REJECTED BECAUSE THEY DID NOT HAVE SUFFICIENT
    SPACE

```

Example 5: Full volume or tracks copy

```

ADR947I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR VOLUME SRCV01 BECAUSE
THE SOURCE AND TARGET DEVICES DO NOT PROVIDE COMPATIBLE FAST REPLICATION FUNCTIONS
    VOLUME SRCV01 SUPPORTS DATA SET FLASHCOPY
    VOLUME TGTV01 SUPPORTS SNAPSHOT

```

In examples 4 and 5, DEBUG(FRMSG(SUM)) and DEBUG(FRMSG(DTL)) result in the same level of informational message being issued.

Note:

1. DFSMSdss supplies fast replication ineligible reasons at the summarized level when summary information is applicable.
2. Specifying SUMMARIZED causes DFSMSdss to issue SMS allocation messages regarding fast replication in a data set copy operation when the target data set is SMS-managed.
3. When the FASTREPLICATION(REQUIRED) keyword is specified and the DEBUG(FRMSG(MIN | SUM | DTL)) keyword is not specified, DFSMSdss still issues an informational message when a fast replication

method cannot be used. It is as though the DEBUG(FRMSG(SUMMARIZED)) keyword had been specified.

FRMSG(DETAILED)

Specifies that DFSMSdss is to issue a message with detailed information. When applicable, detailed information regarding ineligible volumes is provided in the message text. The following examples show the messages that are issued when you specify this keyword:

Example 1: Data set copy

```
ADR948I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
DATA SET TEST.SRC.KSDS1 BECAUSE THE SOURCE DEVICES DO NOT PROVIDE COMPATIBLE
DATA SET FAST REPLICATION FUNCTIONS
  VOLUME SRCV01 SUPPORTS DATA SET FLASHCOPY
  VOLUME SRCV02 SUPPORTS DATA SET FLASHCOPY
  VOLUME SRCV03 DOES NOT SUPPORT ANY TYPE OF DATA SET FAST REPLICATION
  VOLUME SRCV14 SUPPORTS SNAPSHOT
  VOLUME SRCV25 DOES NOT SUPPORT ANY TYPE OF DATA SET FAST REPLICATION
  VOLUME SRCV26 DOES NOT SUPPORT ANY TYPE OF DATA SET FAST REPLICATION
```

Example 2: Data set copy

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
DATA SET TEST.SRC.KSDS1, RETURN CODE 3
  VOLUME SRCV01 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 7 - VERSION
    1 FC RELATION EXISTS
  VOLUME SRCV02 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 8 - MAX ESS
    FC RELATIONS
  VOLUME SRCV03 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 8 - MAX ESS
    FC RELATIONS
  VOLUME SRCV04 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE CA - BOUNDARY
    EXCEPTION
```

Example 3: Data set copy, target data set is non-SMS-managed

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
DATA SET TEST.SRC.KSDK1, RETURN CODE 14
  VOLUME TGTV01 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 7 - VERSION 1
    FC RELATION EXISTS
  VOLUME TGTV02 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 8 - MAX ESS
    FC RELATIONS
  VOLUME TGTV03 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 8 - MAX ESS
    FC RELATIONS
  VOLUME TGTV04 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE C9 - FLASHCOPY
    NOT SUPPORTED
  VOLUME TGTV05 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE C9 - FLASHCOPY
    NOT SUPPORTED
  VOLUME TGTV21 WAS REJECTED FOR DFSMSDSS REASON CODE 1 - INSUFFICIENT
    SPACE
  VOLUME TGTV22 WAS REJECTED FOR DFSMSDSS REASON CODE 2 - NO FREE DSCB IN THE VTOC
  VOLUME TGTS01 WAS REJECTED FOR DFSMSDSS REASON CODE 3 - VOLUME IS
    SMS MANAGED
  VOLUME TGTS02 WAS REJECTED FOR DFSMSDSS REASON CODE 3 - VOLUME IS
    SMS MANAGED
  VOLUME TGTS23 WAS REJECTED FOR DFSMSDSS REASON CODE 4 - LSPACE MACRO
    FAILED WHILE CALCULATING FREE SPACE
  VOLUME TGTS24 WAS REJECTED FOR DFSMSDSS REASON CODE 8 - DADSM FAILURE
    OCCURRED WHILE ALLOCATING THE DATA SET ON THE VOLUME
```

Example 4: Full volume or tracks copy

```
ADR947I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
VOLUME SRCV01 BECAUSE THE SOURCE AND TARGET DEVICES DO NOT PROVIDE
COMPATIBLE FAST REPLICATION FUNCTIONS
  VOLUME SRCV01 SUPPORTS DATA SET FLASHCOPY
  VOLUME TGTV01 SUPPORTS SNAPSHOT
```

Note:

1. DFSMSdss supplies fast replication ineligible reasons at the individual volume level when detailed information is applicable.

2. Specifying DETAILED causes DFSMSdss to issue SMS allocation messages regarding fast replication in a data set copy operation when the target data set is SMS-managed. For an SMS allocation, DFSMSdss supplies the same level of information as if you had specified DEBUG(FRMSG(SUMmarized)).
3. For a non-SMS allocation during a data set copy operation, DFSMSdss supplies fast replication ineligible reasons at the individual volume level.

DELETE



DELETE specifies that for a data set copy DFSMSdss deletes VSAM and non-VSAM data sets from the source volume after a successful copy. This moves, in effect, a data set from one volume to another. The data sets are scratched and uncataloged.

Note:

1. Specify DELETE when you are copying cataloged data sets. If you do not specify DELETE when you are copying cataloged data sets, the target data set must either be cataloged in a different catalog (using the RECATALOG keyword) or renamed (using the RENAMEU keyword).
2. If you copy a data set with DFM attributes to a non-SMS-managed target, the new data set will not have the DFM attributes.
3. *Unexpired* source data sets are deleted only if you also specify PURGE.
4. Even if PROCESS (SYS1) is specified, SYS1.VVDS and SYS1.VTOCIX data sets cannot be copied and deleted.
5. Do not specify SHARE if you specify DELETE.
6. If DFSMSdss encounters a damaged PDS during logical data set copy, it displays messages indicating the nature and relative location of the problem. In order to maintain complete data integrity, DFSMSdss does not delete the source data set. The copy of the data set fails, and the target is deleted. In order to copy and delete a damaged PDS, use the NOPACKING keyword.
7. Do not specify DELETE with CONCURRENT, because after the concurrent copy operation has begun, the original data can still be updated.
8. Do not specify DELETE with CICSVRBACKUP.
9. Specify DELETE to preserve aliases that are associated with non-VSAM data sets. The following criteria must be met for this to work:
 - RENAMEU cannot be specified at the same time.
 - The data set must be SMS-managed and remain SMS-managed during the copy operation.
10. For physical data set copy processing, only a single-volume non-VSAM data set may be deleted from a volume that is not a dump conditioned volume. If the DELETE keyword is specified and the input volume specified on the PHYSINDD keyword is a dump conditioned volume, then the DELETE keyword will be ignored while processing all data sets from that volume. If the DELETE keyword is specified and the target data set is single volume and the target volume is SMS managed and the target is not preallocated, the target data set will be cataloged.
11. If the data set being processed is a generation data set (GDS), an exclusive enqueue on the GDG BASE is required in addition to the exclusive enqueue on the GDS.

For more information about copying non-VSAM data sets that have aliases, see [“Moving data sets with special requirements”](#) on page 111.

DUMPCONDITIONING



DUMPCONDITIONING specifies that you want to create a copy of the source volume for backup purposes rather than for the applications to use the target volume.

When you specify DUMPCONDITIONING, the volume serial number of the target volume does not change, and the target volume remains online after the copy. The VVDS and VTOC index names on the target volume will not change to match the target volume serial number. They will continue to match the source volume serial number. This volume is a "dump conditioned volume."

Note:

1. Do not use the DUMPCONDITIONING keyword with the COPYVOLID keyword.
2. DUMPCONDITIONING applies to TRACKS COPY operations only if the tracks selected for copying include the VTOC. Otherwise, DFSMSdss ignores DUMPCONDITIONING.
3. You may not be able to access data on the target volume after a DUMPCONDITIONING COPY operation. This is because the VVDS and VTOC index names do not match the target *volser*. Use the resulting target volume for either of the following operations: as the source volume for a FULL volume DUMP operation or the source of another FULL volume DUMPCONDITIONING COPY operation.
4. You must specify the DUMPCONDITIONING keyword to perform a FULL volume COPY operation of a dump conditioned volume.
5. If a conditioned volume is copied back using DUMPCONDITIONING, conditioning is not performed on the original source volume. Instead, DFSMSdss recognizes that it is copying from the target of a previous conditioned-backup and recovers the original source volume.

DYNALLOC

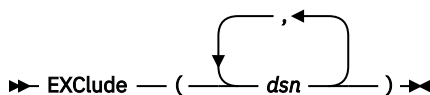


DYNALLOC specifies dynamic allocation, instead of enqueue, to serialize the use of data sets. The data sets whose extents are to be relocated are serialized throughout the copy operation. This allows cross-system serialization in a JES3/MVS environment.

Note:

1. Serialization is of value only when you use the dynamic allocation or the JES3 interface is not disabled.
2. Run time increases when you use the DYNALLOC keyword to serialize data sets (as opposed to enqueue) because overhead is involved in dynamic allocation and serialization across multiple processors.
3. If a data set passes INCLUDE/EXCLUDE filtering and is migrated before BY filtering and the DYNALLOC keyword is used, the dynamic allocation causes the data set to be recalled. DFSMSdss waits for the recall processing to complete. If the data set is recalled to a different volume, a message indicates that the VTOC entry was not found.
4. For an HFS source data set, DFSMSdss ignores DYNALLOC and attempts to get a SYSZDSN enqueue. If the enqueue attempt fails, DFSMSdss attempts to quiesce the HFS data set.
5. For a physical data set copy operation, the DYNALLOC keyword will be ignored for the source data set when that data set resides on a dump conditioned volume.

EXCLUDE

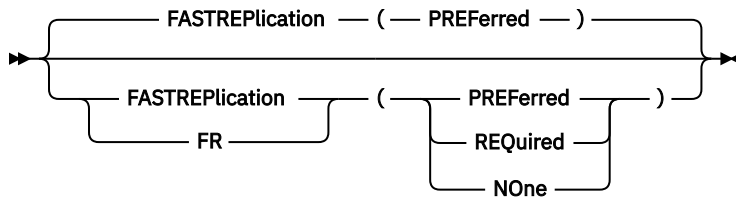


dsn

Specifies the name of a data set to be excluded from the data sets selected by the INCLUDE keyword. Either a fully or a partially qualified data set name can be used. See the separate discussions of INCLUDE and BY for information about how these keywords are specified.

Note: You must use FILTERDD when you have more than 255 entries in INCLUDE, EXCLUDE, or BY list keywords.

FASTREPLICATION



FASTREPLICATION specifies whether the use of fast replication is preferred, required, or not desired. This keyword applies to fast replication methods such as FlashCopy and SnapShot. It does not affect concurrent copy or virtual concurrent copy processing.

PREFERRED specifies that you want to use a fast replication method, if possible. If fast replication cannot be used, DFSMSDss completes the operation using traditional data movement methods. PREFERRED is the default (unless changed to NONE by the installation).

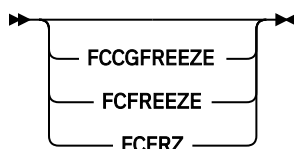
REQUIRED specifies that fast replication must be used. For full volume or tracks COPY operations, DFSMSDss fails the operation if fast replication cannot be used. For a data set COPY operation, DFSMSDss stops processing the current data set if fast replication cannot be used. However, DFSMSDss continues processing the rest of the data sets using fast replication. When the DEBUG(FRMSG(MIN|SUM|DTL)) keyword is not specified, DFSMSDss still issues summarized information regarding why a fast replication method cannot be used as though DEBUG(FRMSG(SUMMARIZED)) had been specified. The DEBUG(FRMSG(MIN | SUM | DTL)) keyword determines the amount of information provided for why you cannot use a fast replication method.

NONE specifies that fast replication should not be used. DFSMSDss does not attempt to use fast replication and completes the operation using traditional data movement methods.

Note:

1. Do not use the FASTREPLICATION (REQUIRED) keyword with the CONCURRENT(ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ) keyword.
2. Do not use the FASTREPLICATION (NONE) keyword with the FCNOCOPY keyword.
3. Use VOLCOUNT(ANY) together with FASTREPLICATION (REQUIRED) or FASTREPLICATION (PREFERRED) when copying VSAM data sets and the following conditions are all true:
 - The data set is single volume
 - The smallest allocation quantity is less than 1 cylinder
 - The primary allocation is less than the secondary allocation.

FCCGFREEZE



FCCGFREEZE specifies that the source volume is to be part of a FlashCopy Consistency Group. Subsequent I/O activity to the FlashCopy source volume will be held until the CGCREATED (thaw) command is processed on the logical subsystem (LSS) where the volume resides or when the FlashCopy Consistency Group timer expires.

Note:

1. Do not specify FCCGFREEZE with any of the following keywords:

- CONCURRENT(ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ)
 - DATASET
 - FASTREPLICATION(PREFERRED | NONE)
 - FCNOCOPYTOCOPY
2. DFSMSDss supports FlashCopy Consistency Group at a volume level. When you specify FCCGFREEZE with TRACKS, you must also specify the CPVOLUME keyword. FCCGFREEZE is supported with the TRACKS keyword only for full volume copy of VM-format volumes with OS-compatible VTOCs.
 3. When FCCGFREEZE is specified, it indicates FlashCopy Version 2 must be used to copy the data. If FlashCopy V2 cannot be used, the copy operation will fail.
 4. The FCCGFREEZE option requires the specified devices support the FlashCopy Consistency Group function.
 5. Because I/O activity is held on source volumes that are frozen, it is recommended *not* to include system volumes that are required to run the current COPY command (or subsequent COPY commands) needed to form a FlashCopy Consistency Group. Examples of such system volumes include spool, page, and volumes containing checkpoint data sets, catalogs, and RACF databases.

FCFASTREVERSERESTORE



FCFASTREVERSERESTORE specifies that the use of fast reverse restore is required. Fast reverse restore gives the option to restore a FlashCopy source from its FlashCopy target without having to wait for completion of the background copy operation.

A FlashCopy relationship must exist between the source and the target and must be a single FlashCopy relationship that covers the entire volume (from track 0 through the last track on the volume). The relationship can be an incremental FlashCopy relationship.

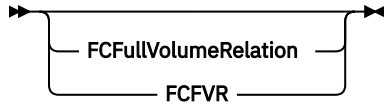
The existence of a FlashCopy relationship will be verified. If a relationship does not exist between the source and the target, the fast reverse restore request will fail. Verification can be bypassed using ADRUFO, however, it will be difficult to determine the cause of a request failure.

The contents of the source volume, which is the original target of a FlashCopy operation, are unpredictable after the fast reverse restore operation is complete and should not be used.

Note:

1. Do not specify FCFASTREVERSERESTORE with any of the following keywords:
 - FCTOPPRCPPRIMARY (PRESMIRPREF | PRESMIRREQ)
 - FASTREPLICATION (PREFERRED | NONE)
 - CONCURRENT, FCFREEZE, FCINCREMENTAL, FCINCREMENTALLAST, FCINCRVERIFY, FCNOCOPYTOCOPY, FCWAIT
 - FCTOXRCPPRIMARY
2. The source and target device capacity must be the same.
3. If the target volume specified is the source volume of other FlashCopy relationships, and the storage subsystem does not support Cascaded FlashCopy relationships, then those copies must be withdrawn prior to using FCFASTREVERSERESTORE. Otherwise the request will fail.
4. ADRUFO can be configured to retry a failed FlashCopy recovery without the use of fast reverse restore. If the subsequent attempt is successful, a new FlashCopy relationship between the specified source and target volumes will be created.

FCFULLVOLUMERELATION

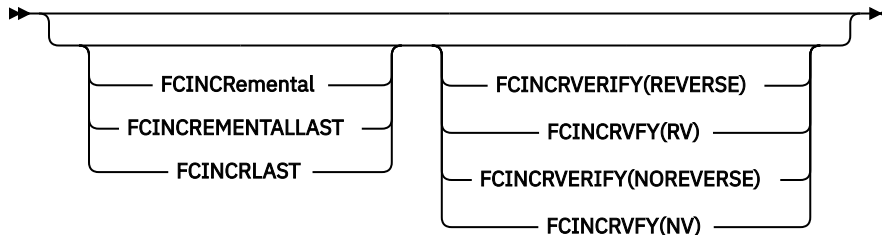


FCFULLVOLUMERELATION specifies that a single FlashCopy relationship will be created that covers the entire volume (from track 0 through the last track on the volume). This means the entire volume will be copied, including free space. If FCFULLVOLUMERELATION is not specified, free space is not copied.

Note:

1. Do not specify FCFULLVOLUMERELATION with any of the following keywords:
 - FASTREPLICATION (NONE)
2. If FlashCopy cannot be used to move the data, the FCFULLVOLUMERELATION keyword will be ignored.
3. If the FCFASTREVERSERESTORE keyword is specified, the FCFULLVOLUMERELATION keyword does not need to be specified.

FCINCREMENTAL



FCINCREMENTAL specifies that DFSMSdss establishes a full volume Incremental FlashCopy relationship from the specified source volume (in the INDD/INDYNAM keyword) to the specified target volume (in the OUTDD/OUTDYNAM keyword). The full volume FlashCopy relationship remains in effect after the initial copy has completed and subsequent changes to the source and target volumes are tracked so that a future FlashCopy operation is performed to only copy incremental changes.

When FCINCREMENTAL is specified and no Incremental FlashCopy relationship currently exists between the volume pair, the storage subsystem initiates background copy of the entire source volume to the target volume.

When FCINCREMENTAL is specified and an Incremental FlashCopy relationship already exists between the volume pair, the storage subsystem only copies the changed data in the specified direction which can be the same as the existing (original) or the reverse of the existing FlashCopy direction. When no updates have been made to the existing target since the last Incremental FlashCopy, the reverse of FlashCopy direction can be used to restore the original source volume back to the previous point-in-time copy state. The new source volume is designated by the INDD/INDYNAM keyword and the new target is designated by the OUTDD/OUTDYNAM keyword on the copy command.

Attention: DFSMSdss does not inhibit writes to the FlashCopy source or target volumes. If you plan to use the Incremental FlashCopy target volume as a backup, you must ensure that the target volume is not updated inadvertently.

Note:

1. Do not specify the FCINCREMENTAL keyword with any of the following keywords:
 - DATASET
 - FCINCRMENTALLAST
 - CONCURRENT(ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ)
 - FASTREPLICATION(PREFERRED | NONE)

- FCNOCOPY
 - FCNOCOPYTOCOPY
2. Incremental FlashCopy is supported at a volume level. It is not supported for data set FlashCopy.
 3. Incremental FlashCopy Version 2 supports up to the maximum number of full volume flashcopies allowed for a source, which is 12. The initial Incremental FlashCopy support, known as Incremental FlashCopy Version 1, allows only 1 full volume FlashCopy to be incremental. A Version 1 Incremental FlashCopy relationship can coexist with Version 2 Incremental FlashCopy relationships.
 4. When you specify FCINCREMENTAL with TRACKS, you must also specify the CPVOLUME keyword. FCINCREMENTAL is supported with the TRACKS keyword only for full volume copy of VM-format volumes with OS-compatible VTOCs.
 5. The benefit of Incremental FlashCopy is to minimize the amount of data transfer needed when a FlashCopy pair is refreshed. There is no saving in data transfer when the no-background copy option is chosen. Therefore, DFSMSdss does not allow FCINCREMENTAL and FCNOCOPY to be specified together.
 6. 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.
 7. When you specify COPYVOLID with FCINCREMENTAL, the volume serial number of the target volume is changed to match the source's and the target volume is varied offline. Prior to performing a subsequent incremental copy using DFSMSdss, the offline volume's volume serial number must be changed by using a utility such as ICKDSF and the volume must be varied online. Although you can specify FCINCREMENTAL with COPYVOLID, IBM recommends you use FCINCREMENTAL with DUMPCONDITIONING.
 8. The PURGE keyword may be required on subsequent incremental copies.
 9. When FCINCREMENTAL is specified, it requires that the storage facility has the Change Recording feature enabled. The Incremental FlashCopy request fails if the storage facility does not support Change Recording.
 10. The Incremental FlashCopy direction can only be reversed when the previous physical background copy has completed. If the background copy is still in progress, the new Incremental FlashCopy attempt fails. You can instruct DFSMSdss to wait for background copy to complete by specifying the FCWAIT keyword. See the FCWAIT keyword description for more information.
 11. FCINCREMENTAL specifies that the full volume FlashCopy relationship remains in effect (persists) after the background copy has completed and subsequent changes to the source and target volumes are tracked. You can specify the INCREMENTALLAST keyword instead of FCINCREMENTAL. If you want DFSMSdss to initiate FlashCopy of the final increment, stop tracking the changes, and have the relationship ended when background copy has completed.

FCINCREMENTALLAST

See "FCINCREMENTAL" in ["FCINCREMENTAL"](#) on page 331 for syntax diagram.

FCINCREMENTALLAST specifies that DFSMSdss establishes FlashCopy of the final increment from the specified source volume (in the INDD/INDYNAM keyword) to the specified target volume (in the OUTDD/OUTDYNAM keyword), stops change recording, and have the FlashCopy relationship ended when the background copy has completed.

When FCINCREMENTALLAST is specified and no Incremental FlashCopy relationship currently exists between the volume pair, DFSMSdss will establish a non-incremental FlashCopy of the entire source volume to the target volume. The FlashCopy relationship will end when the background copy has completed.

When FCINCREMENTALLAST is specified and an Incremental FlashCopy relationship already exists between the volume pair, the storage subsystem will only copy the changed data in the specified direction which can be the same as the existing (original) or the reverse of the existing FlashCopy direction. When no updates were made to the existing target since the previous Incremental FlashCopy, the reverse of FlashCopy direction can be used to restore the original source back to the previous point-in-time

copy state. The new source volume is designated by the INDD/INDYNAM keyword and the new target is designated by the OUTDD/OUTDYNAM keyword on the copy command.

See the INCREMENTAL keyword description for more information about Incremental FlashCopy.

Note:

1. Do not specify the FCINCREMENTALLAST keyword with any of the following keywords:
 - DATASET
 - FCINCRMENTAL
 - CONCURRENT (ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ)
 - FASTREPLICATION(PREFERRED | NONE)
 - FCNOCOPY
 - FCNOCOPYTOCOPY
2. When you specify FCINCREMENTALLAST with TRACKS, you must also specify the CPVOLUME keyword. FCINCREMENTALLAST is supported with the TRACKS keyword only for full volume copy of VM-format volumes with OS-compatible VTOCs.
3. When FCINCREMENTALLAST is specified, it requires that the storage facility has the Change Recording feature enabled. The Incremental FlashCopy request will fail if the storage facility does not support Change Recording.
4. The Incremental FlashCopy direction can only be reversed when the previous physical background copy has completed. If the background copy is still in progress, the new Incremental FlashCopy attempt will fail. You can instruct DFSMSdss to wait for background copy to complete by specifying the FCWAIT keyword. See the FCWAIT keyword description for more information.
5. If you want the full volume Incremental FlashCopy relationship to remain in effect (persists) after the copy has completed and subsequent changes to the source and target volumes to be tracked, specify the FCINCREMENTAL keyword.

FCINCRVERIFY

See [“FCINCREMENTAL”](#) on page 331 for Syntax diagram.

NOREVERSE

Specifies that the new FlashCopy direction is the same as the existing (original) FlashCopy direction.

REVERSE

Specifies that the new FlashCopy direction is the reverse of the existing (original) FlashCopy direction.

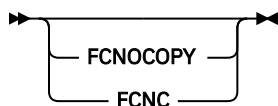
FCINCRVERIFY specifies that DFSMSdss should verify the new and the existing Incremental FlashCopy direction before copying incremental changes. The new source volume is designated by the INDD/INDYNAM keyword and the new target is designated by the OUTDD/OUTDYNAM keyword on the copy command. If the new and the existing FlashCopy directions match what the user expected -- the same or reversed -- DFSMSdss will proceed to copy the incremental changes in the specified (new) direction. If the new and the existing FlashCopy directions do not match what the user expected, DFSMSdss will fail the copy task without copying the new increment.

When FCINCRVERIFY is specified and no Incremental FlashCopy relationship currently exists, DFSMSdss will fail the copy task without establishing a new FlashCopy relationship.

When FCINCRVERIFY is specified and an Incremental FlashCopy relationship exists between the specified volume pair, DFSMSdss will verify the FlashCopy directions before proceeding.

Note: When FCINCRVERIFY is specified, either the FCINCREMENTAL or the FCINCREMENTALLAST keyword must also be specified.

FCNOCOPY



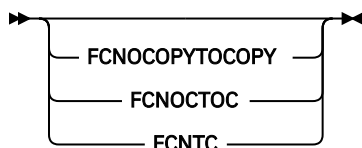
FCNOCOPY specifies that if FlashCopy is used to perform the copy operation, then the ESS subsystem does not perform a physical copy of the data. If FCNOCOPY is not specified and FlashCopy is used to perform the operation, then the ESS subsystem performs a physical copy of the data in order to release the subsystem resources that are used to maintain the FlashCopy relationship (a virtual copy of the data).

When FlashCopy is not used to perform the copy operation, the FCNOCOPY keyword is ignored.

Note:

1. Do not specify the FCNOCOPY keyword with the FASTREPLICATION(NONE) keyword.
2. If FCNOCOPY is not specified and the ESS subsystem performs the physical copy, the DFSMSdss copy operation will not be delayed. However, performing the physical copy uses subsystem resources, which can impact the performance of other I/O operations that are issued to the ESS.
3. Be aware that if you use the FCNOCOPY keyword, you must withdraw the FlashCopy relationship in which the copy is no longer needed in order to free up the subsystem resources that maintain the FlashCopy relationship. You can withdraw the FlashCopy relationship by performing one of the following options:
 - Initiate a dump of the target of the copy and specify the FCWITHDRAW keyword on the DUMP command.
 - Initiate the TSO FCWITHDR command.

FCNOCOPYTOCOPY



FCNOCOPYTOCOPY specifies that DFSMSdss initiate background copy between the specified source and any target regardless of whether there is an existing FlashCopy no-background copy (NOCOPY) relationship associated with that source. As a result, the remaining unchanged source tracks will be written to the target. When the physical background copy completes, the FlashCopy relationship will end unless the relationship is persistent.

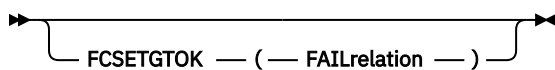
If there are not any existing FlashCopy no-background copy relationships associated with the specified source, no operations will be performed as a result of this COPY command. No messages will be issued if there are not any existing relationships to be converted.

Note:

1. Do not specify the FCNOCOPYTOCOPY keyword with any of the following keywords:
 - CPVOLUME
 - DELETE
 - FASTREPLICATION(REQUIRED | PREFERRED | NONE)
 - FCCGFREEZE
 - FCINCREMENTAL | FCINCREMENTALLAST
 - FCNOCOPY
2. FCNOCOPYTOCOPY operation does not create a new copy of the source data.

3. FCNOCOPYTOCOPY operation initiates background copy of any NOCOPY FlashCopy relationships in which the specified source is participating. If output data sets, tracks (OUTTRACKS), or volumes (OUTDDNAME/OUTDYNAM) are specified, they are ignored.
4. The source extent ranges specified or determined by DFSMSdss in the FCNOCOPYTOCOPY conversion request might not match the existing FlashCopy relationships. Any existing no-background copy FlashCopy relationships with extent ranges intersecting the source tracks specified in the conversion request will have the entire relationship converted. For example, if FlashCopy no-background copy relationships were established for 2 extents: tracks 1 through 50 and tracks 70 through 100, a subsequent COPY FCNOCOPYTOCOPY issued for an extent range of tracks 30 through 90 would result in background copy being initiated for tracks 1 through 50 and 70 through 100. Tracks 51 through 69 would be ignored.
5. The existing FlashCopy relationships may have been established with the no-background copy option by a previous DFSMSdss copy job with the FCNOCOPY option, TSO FCESTABL command with MODE(NOCOPY), or other programs. The FCNOCOPYTOCOPY option will convert any existing FlashCopy no-background copy relationships regardless of which program established the NOCOPY relationships.

FCSETGTOK



FCSETGTOK specifies that when FlashCopy is used to perform a full volume copy operation, the target volume can be a track space efficient volume. This type of FlashCopy relationship is called space efficient FlashCopy to a track space efficient target. You can use space efficient FlashCopy to a track space efficient target for a full-volume copy operation only; that is, a COPY FULL operation or a COPY TRACKS command that specifies a full volume.

Observe the following considerations:

- If you specify FCSETGTOK with COPY FULL, and the target is a track space efficient volume, DFSMSdss attempts to establish a full-volume FlashCopy relationship without excluding free space, which results in one FlashCopy relationship for the entire volume. If FlashCopy cannot be used, DFSMSdss issues an error message and the copy operation fails. DFSMSdss does not attempt to use another method of data movement.
- To use FCSETGTOK with COPY TRACKS, your command must specify one track range (an extent) that includes the entire volume (tracks 0 through *n*). Otherwise, the FCSETGTOK keyword has no effect on the copy operation.

When you specify the FCSETGTOK keyword, you must also specify the following sub-keyword to indicate what action DFSMSdss is to take if space on the repository volume is exhausted during the FlashCopy relationship:

FAILRELATION

This sub-keyword, abbreviated as FAIL, specifies that if space on the repository volume is exhausted during the FlashCopy relationship, DFSMSdss is to place the relationship in a failed state and mark the target copy as not valid. During the failed state, DFSMSdss continues to allow read and write operations to the source volume, but does not copy the updated tracks to the target volume. DFSMSdss fails any read or write requests for the target volume.

To clear this condition for the target volume, you must initialize the target volume through the ICKDSF INIT command. Doing so causes the FlashCopy relationship to be withdrawn and the space on the space efficient volume to be released.

Using FCSETGTOK might require RACF authorization. If your installation has defined the RACF FACILITY class profile, STGADMIN.ADR.COPY.FCSETGT, your user ID requires READ access to the profile. For more information, see [“Protecting the usage of DFSMSdss” on page 535](#).

DFSMSdss ignores the FCSETGTOK keyword for COPY operations in which:

- FlashCopy is not used to perform the copy operation
- The target volume is not a track space efficient volume
- Less than a full volume is to be copied, for example, a COPY DATASET operation.

Example: In the following example, a full volume copy and dump is requested. Based on the FCSETGTOK setting, if space on the repository volume is exhausted during the FlashCopy relationship, DFSMSdss continues to allow read and write operations to the source volume, VOL00A, but does not copy the updated tracks to the target volume VOL00B:

```
COPY FULL INDYNAM(VOL00A) OUTDYNAM(VOL00B) DUMPCONDITIONING -
  ADMIN PURGE FCNOCOPY FCSETGTOK(FAIL)

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

Note:

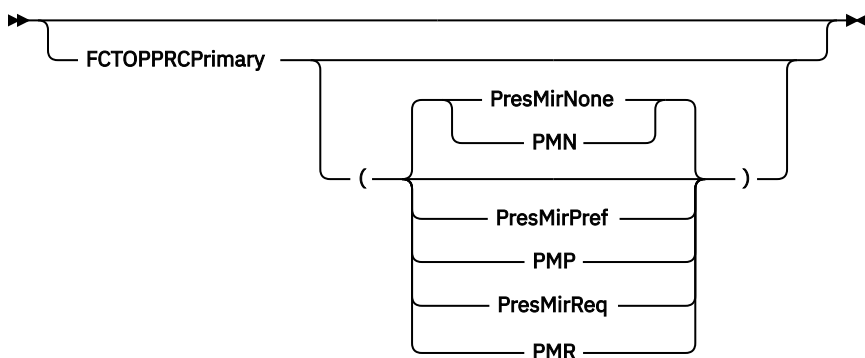
1. Space efficient FlashCopy to a track space efficient volume is intended for full volume copies that are short term in nature, such as those that are to be backed up to tape. Space efficient FlashCopy to a track space efficient volume might also be appropriate for longer term copies, if the source and target volumes are not frequently updated. Extent space efficient (ESE) volumes do not have these restrictions.
2. For a FlashCopy request with an intended target of a track space efficient volume, you must also specify the FCNOCOPY keyword. Otherwise, your request will fail. This restriction does not apply to extent space efficient (ESE) volumes.
3. Do not specify FCSETGTOK with the FASTREPLICATION(NONE) keyword.

For more information about space efficient FlashCopy, see [z/OS DFSMS Advanced Copy Services](#).

For information about RACF protection for DFSMSdss functions and keywords, see [Chapter 5, “Protecting DFSMSdss functions,”](#) on page 35.

For more information about RACF FACILITY class profiles, see [“Protecting DFSMSdss functions with RACF FACILITY class profiles”](#) on page 37.

FCTOPPRCPPRIMARY



FCTOPPRCPPrimary specifies that if FlashCopy is used to perform the copy operation, a Peer-toPeer Remote Copy (PPRC) primary volume is allowed to become a FlashCopy target volume. Use the following sub-keywords to specify whether the device pair is allowed to go to duplex pending state if the target volume of the FlashCopy operation is a metro mirror primary device:

PRESMIRREQ

specifies that if the target volume is a Metro Mirror primary device, the pair must not go into a duplex pending state as the result of a FlashCopy operations.

PRESMIRPREF

specifies that if the target volume is a Metro Mirror primary device, it would be preferable that the pair does not go into a duplex pending state as the result of a FlashCopy operation. However, if a Preserve Mirror operation cannot be accomplished, the FlashCopy operation is still to be performed.

The PRESMIRPREF option is not valid for DS888x and newer, and compatible, storage subsystems. If the PRESMIRPREF option is specified and the volumes involved in the operation reside on DS888x or newer storage subsystems, the command results in a warning or error message. This is also true of the corresponding option provided in byte 33 (X'21') in the ADRUFO data area.

PRESMIRNONE

specifies that Preserve Mirror operation is not to be done, even if all of the configuration requirements for a Preserve Mirror operation are met. If the target specified is a Metro Mirror primary device, the pair is to go into a duplex pending state while the secondary device is updated with the tracks to be copied. PRESMIRNONE is the default if you specify FCTOPPRCPPrimary without a subkeyword.

Attention: When you specify FCTOPPRCPPrimary or FCTOPPRCPPrimary(PRESMIRNONE), the FlashCopy operation causes a PPRC primary volume to become a FlashCopy target volume. A Metro Mirror pair currently in full duplex state, goes into a duplex pending state when the FlashCopy relationship is established. When Metro Mirror completes the copy operation, the Metro Mirror pair goes to full duplex state. To prevent Metro Mirror pairs from going to duplex pending state during FlashCopy operation, you must specify FCTOPPRCPPrimary(PRESMIRREQ).

Note:

1. Using FCTOPPRCPPrimary might require RACF authorization.
2. Do not specify the FCTOPPRCPPrimary keyword with the PRESMIRPREF and PRESMIRREQ subkeywords, if you specify the FCSETGTOK keyword with FAILRELATION.
3. Do not specify the FCTOPPRCPPrimary keyword with the FASTREPLICATION(NONE) keyword.
4. When FlashCopy is not used to perform the copy operation, the FCTOPPRCPPrimary keyword is ignored.
5. When FCTOPPRCPPrimary is not specified or if the capability is not supported by the ESS, a PPRC primary volume is not eligible to become a FlashCopy target volume.

For more information about RACF authorization, see [z/OS DFSMSdss Storage Administration](#).

For more information about RACF FACILITY class profiles, see [z/OS Security Server RACF Security Administrator's Guide](#).

For more information about PPRC, Metro Mirror, Global Copy, and Global Mirror, see [z/OS DFSMS Advanced Copy Services](#).

FCTOXRCPPrimary

FCTOXRCPPrimary indicates that if the specified target volume is an XRC primary device, it is allowed to become the target of a FlashCopy operation during any type of COPY command.

Note:

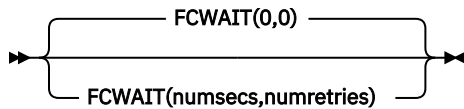
1. To specify FCTOXRCPPrimary, RACF authorization may be required.
2. Do not use the FastTReplication(NONE), COPYVolid, or FCFastReverseRestore keywords with the FCTOXRCPPrimary keyword.
3. The DEBUG(FRMSG(SUMMARIZED)) keyword is set as the default if no other DEBUG FRMSG option is specified.

For more information about RACF authorization, see [z/OS DFSMSdss Storage Administration](#).

For more information about RACF FACILITY class profiles, see [z/OS Security Server RACF Security Administrator's Guide](#).

For more information about Remote Pair FlashCopy for XRC, see [z/OS DFSMS Advanced Copy Services](#).

FCWAIT



numsecs

Specifies a decimal number (0-255) that designates the time, in seconds, to wait before checking for physical background copy completion.

numretries

Specifies a decimal number (0-99) that designates the maximum number of additional queries to make on physical background copy completion.

FCWAIT specifies to DFSMSdss the length of the wait in seconds, and the number of additional queries on FlashCopy background copy completion. The combination of retry interval and maximum number of retries (numsecs times numretries) designated in the FCWAIT keyword specifies the maximum length of time for DFSMSdss to wait for an existing physical background copy to complete before either initiating FlashCopy Establish or failing the COPY FULL or COPY TRACKS operation.

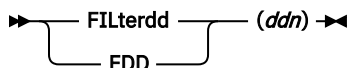
When FCWAIT(numsecs,numretries) is specified, DFSMSdss will check for existing background copy completion before initiating a FlashCopy attempt. If no background copy is currently in progress from the COPY source to the target, DFSMSdss will establish FlashCopy immediately. If background copy is in progress, DFSMSdss will recheck at the specified interval until the designated number of retries has been reached. If background copy remains in progress when the maximum wait time has been reached, DFSMSdss will fail the COPY FULL or COPY TRACKS operation.

The default for numsecs,numretries is (0,0). In other words, when FCWAIT is not specified, or when either numsecs or numretries is 0, DFSMSdss will attempt to establish FlashCopy without waiting for active background copy to end.

Note:

1. The FCWAIT keyword is ignored when neither the FCINCREMENTAL nor the FCINCREMENTALLAST keyword is also specified.
2. The FCWAIT keyword is ignored when Incremental FlashCopy direction is not being reversed.
3. The FCWAIT keyword is ignored if FlashCopy cannot be attempted.

FILTERDD

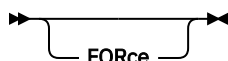


ddn

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 use. This is in the form of card-image records, in DFSMSdss command syntax, that contain the INCLUDE, EXCLUDE, and BY keywords that complete the COPY command syntax.

Note: You must use FILTERDD when you have more than 255 entries in the INCLUDE, EXCLUDE, or BY list of subkeywords.

FORCE



FORCE specifies that DFSMSDss copy one or more unmovable data sets to a like or unlike device type. Unmovable data sets are those allocated as absolute track (ABSTR) or as unmovable (PSU, POU, DAU, or ISU). The allocation attribute, unmovable or ABSTR, is carried over to the output volume.

When copying to like devices, DFSMSDss copies the data sets to the same track locations on the target volume. In this case, FORCE is not required if the target volume uses an indexed VTOC, and the space where the unmovable data set is to reside is available. If any of these conditions is not true, DFSMSDss does not copy any unmovable data sets unless FORCE is specified. In this case, DFSMSDss places the unmovable data sets in any available location.

You must specify FORCE when copying unmovable data sets to unlike devices. DFSMSDss places the unmovable data sets in any available location.

Restriction: Use the EXCLUDE keyword (with the FORCE keyword) to designate data sets that have CCHHR (cylinder, cylinder, head, head, record) location-dependent data. This prevents DFSMSDss from moving the location-dependent data sets.

FORCECP

➤ **FORCECP** (— *days* —) ➤

FORCECP specifies that checkpoint data sets resident on the SMS volume or volumes can be copied. Checkpoint indicators are removed from the target data set.

days

Specifies a decimal number in the range of zero to 255, and specifies the number of days that must have elapsed since the last referenced date before the data set can be copied.

FREESPACE

➤ **FREESPACE** (— *CI* — , *CA* —) ➤
 FSPC

FREESPACE specifies free space values for DFSMSDss-allocated target VSAM data sets. If this keyword is omitted, the control interval and control area free space are the same as the source data set.

CI

Specifies the percentage of free space to be kept in each control interval during allocation of the data set.

CA

Specifies the percentage of free space to be kept in each control area during allocation of the data set. When omitted, the control area free space is the same as the source data set.

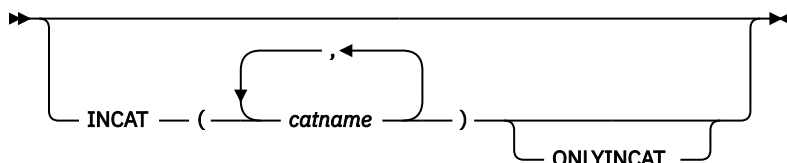
FULL

➤ **FULL** ➤

FULL specifies that an entire DASD volume is to be copied. This is the default. Unallocated tracks are not copied. Unless specified by ALLDATA or ALLEXCP, only the used (rather than allocated) tracks are copied for sequential data sets, partitioned data sets, and for data sets with unknown data set organization (for example, JES2/JES3 data sets with a data set organization that is null). If the VTOC has errors, all tracks are copied. Used tracks consist of the tracks from the beginning of the data set to the last-used track (as indicated by the last used block pointer in the data set's VTOC entry).

Note: You cannot specify the SHARE or TOL(ENQF) keywords for FULL operations.

INCAT



INCAT(*catname*) specifies that DFSMSdss search the user catalogs specified by the INCAT(*catname*) keyword, then follow the standard search order to locate data sets. INCAT(*catname*) allows you to identify specific source catalogs. You might need RACF authorization to use the INCAT keyword.

catname

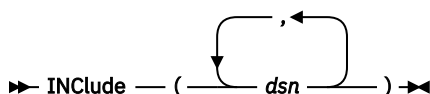
Specifies a fully qualified catalog name.

ONLYINCAT

Specifies that DFSMSdss only searches catalogs that are specified in the INCAT catalog name list.

DFSMSdss does not process an SMS-managed data set that is cataloged outside the standard order of search, even if it is cataloged in one of the catalogs that is specified with the INCAT keyword. Ensure that the SMS-managed data sets are cataloged under standard catalog search order.

INCLUDE



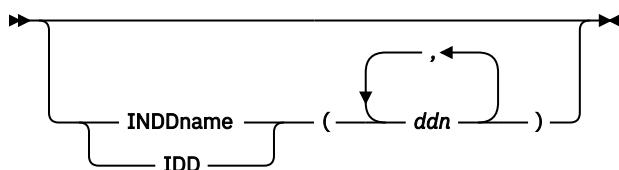
dsn

Specifies the name of a data set eligible to be copied. Either a fully or a partially qualified data set name can be used. See [“Filtering by data set names” on page 256](#). If INCLUDE is omitted (but EXCLUDE or BY is specified) or INCLUDE(**) is specified, *all* data sets are eligible to be selected for copying.

You must use FILTERDD when you have more than 255 entries in INCLUDE, EXCLUDE, or BY list keywords.

DFSMSdss does not support INCLUDE filtering of non-VSAM data sets using an alias.

INDDNAME



ddn

Specifies the name of the DD statement that identifies a volume processed during FULL or TRACKS copy. For a data set copy operation, you can specify multiple names (that is, multiple volumes), separated by commas. For single-volume data sets, each of the DD statements corresponding to a DDNAME (*ddn*) must identify only one volume serial number.

Note: If no input volumes are specified for a data set copy operation, DFSMSdss selects from all data sets cataloged in the catalogs accessible through the standard search order. If either INDDNAME or INDYNAM is specified, DFSMSdss still uses the standard catalog search order, but it selects data sets only from the specified volumes. For multivolume data sets, use LOGINDDNAME or LOGINDDYNAM with SELECTMULTI.

When DFSMSdss invokes IEHMOVE to copy multivolume non-VSAM data sets, IEHMOVE requires that the first DD statement in the job stream must identify all input volumes. DFSMSdss requires separate

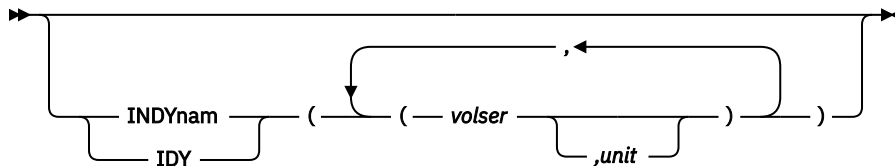
DD statements for the input volumes. To accommodate both requirements, you must code your JCL as follows:

```
//INDD1 DD UNIT=(SYSDA,2),VOL=SER=(VOL1,VOL2),DISP=SHR
//INDD2 DD UNIT=SYSDA,VOL=SER=VOL2,DISP=SHR
```

and code your DFSMSdss control statement:

```
COPY LOGINDD(INDD1,INDD2) ...
```

INDYNAM



The INDYNAM keyword specifies the input volumes that are to be dynamically allocated and copied. The volume must be both mounted and online. For an SMS-managed volume to be dynamically allocated it must be in a status other than DISALL or NOTCON. You cannot specify a nonspecific volume serial number by using an asterisk (*). Only one volume is allowed for a FULL or tracks COPY. Data set copy operations allow multiple volumes, up to 511. To ease JCL coding and command input without appreciably increasing run time, use the INDYNAM keyword instead of data definition statements to allocate DASD volumes.

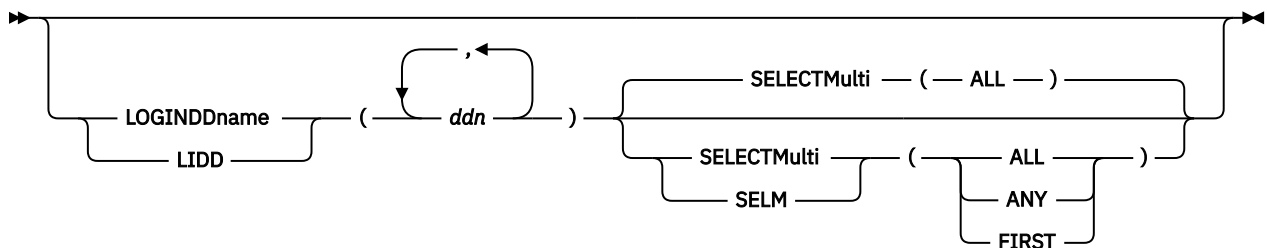
volser

Specifies the volume serial number of a DASD volume that will be copied.

unit

Specifies the device type of a DASD volume that will be copied. This parameter is optional.

LOGINDDNAME



LOGINDDNAME specifies that data sets be selected from the specified volume or volumes (which you can also do by specifying INDDNAME or INDYNAM) and also allows you to specify SELECTMULTI (which cannot be done with INDDNAME or INDYNAM).

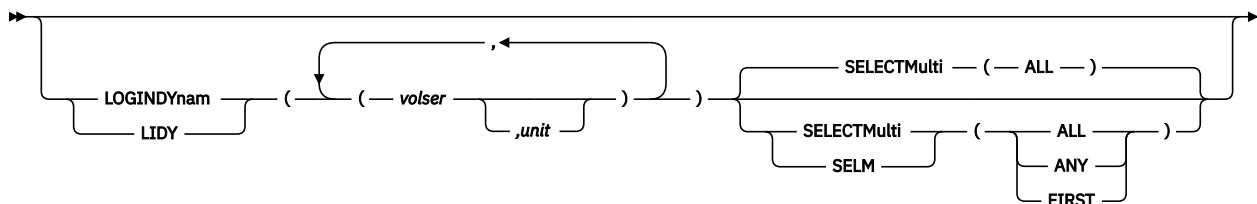
ddn

Specifies the name of the DD statement that identifies a volume that contains the data sets to be copied. For single-volume data sets, each of the DD statements corresponding to a DDNAME (*ddn*) must identify only one volume serial number.

For more information, refer to [“Notes for LOGINDDNAME, LOGINDYNAM and STORGRP keywords”](#) on page 343.

Refer to the description of SELECTMULTI in [“LOGINDYNAM”](#) on page 342.

LOGINDYNAM



LOGINDYNAM specifies that the volumes that contain the data sets to be copied are dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*). You can specify up to 511 volumes.

volser

Specifies the volume serial number of a DASD volume to be copied.

unit

Specifies the device type of a DASD volume to be copied. This parameter is optional.

SELECTMULTI

Specifies the method for determining how cataloged multivolume data sets are to be selected during a logical data set copy operation. SELECTMULTI is accepted only when logical volume filtering is specified with the following keywords:

- LOGINDDNAME
- LOGINDYNAM
- STORGRP

If logical volume filtering is not used, the specification of SELECTMULTI is not accepted.

ALL

Specifies that DFSMSdss *not copy* a multivolume data set unless the following criteria is met:

- The volume list created by the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword lists all the volumes that contain a part of the data set.
- The volume list created by the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword lists all the volumes that contain a part of the VSAM cluster.

ALL is the default.

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

- Specify SPHERE and you only need to 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.

ANY

Specifies that DFSMSdss copy a multivolume data set when any part of the data set or VSAM cluster is on a volume in the volume list created by the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword.

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

- Specify SPHERE and you only need to 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.

FIRST

Specifies that DFSMSdss copy a multivolume data set only when the volume list includes the volume that contains the first part of the data set. The volume list is created by the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword.

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

- Specify SPHERE and you only need to list the volume containing the first extent of the data component in the volume list.
- Do not specify SPHERE and you must list the following in the volume list:
 - The volume containing the first extent of the data component for the base cluster.
 - The volume containing the first extent of the data component for the associated alternate indexes

Notes for **LOGINDDNAME**, **LOGINDYNAM** and **STORGRP** keywords

Note:

1. If the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword is not specified, DFSMSdss selects from all data sets that are cataloged in the catalogs that are accessible through the standard search order.
2. If the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword is specified, DFSMSdss still uses the standard catalog search order, but it selects data sets only from the specified volumes.
3. You must specify the SELECTMULTI keyword to copy a multivolume data set that has extents on volumes which are not identified with the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword.

MAKEMULTI



MAKEMULTI allows DFSMSdss to convert single volume data sets into multivolume data sets. The default is not to convert single volume data sets into multivolume data sets.

This keyword applies only to SMS-managed target data sets. Only single volume, non-VSAM data sets are eligible to be changed into multivolume data sets.

SMS-managed target data sets are given a volume count (VOLCOUNT) that is either:

- The number of SMS output volumes specified in the COPY command, if output volumes are specified through OUTDDNAME or OUTDYNAM
- The number of volumes in the target storage group or 59, whichever is less.

A data set's volume count is the maximum number of volumes to which the data set may extend. At any one time, there may be a mixture of primary volumes (volumes on which space is allocated for the data set) and candidate volumes (volumes on which space may be allocated at a future time). The total sum of the primary volumes and candidate volumes is the data set's volume count.

Note: When MAKEMULTI is specified and VOLCOUNT is also specified with an option other than VOLCOUNT(*), the VOLCOUNT option overrides MAKEMULTI.

MENTITY



MENTITY specifies, for RACF-protected data sets, an entity (*modeldsn*) and, optionally, the serial number of the volume containing that entity (*volser*). These keywords are used to define the data sets to RACF. Specification of MVOLSER is optional for one of the following:

- When the model entity (MENTITY) is cataloged in an integrated catalog facility catalog.
- When a non-VSAM data set is cataloged in the standard catalog search order.

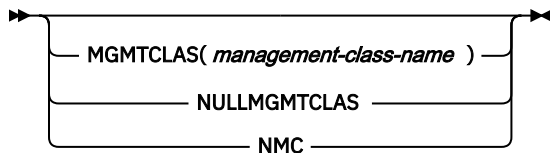
When MVOLSER is specified for a VSAM model entity, *volser* must be the volume serial number of the catalog in which the model entity is cataloged. If these keywords are not specified, DFSMSdss defines the data set to RACF by modeling the target profile after the source data set, if the source is

discretely protected. Target data sets (preallocated or nonpreallocated) are RACF-protected only if the corresponding source data sets were so protected. If a source data set is protected with a generic profile, RACF generic profile checking must be activated prior to invoking the copy function.

Restriction: You cannot specify the MVOLSER(*volser*) keyword by itself. It can only be specified in conjunction with the MENTITY(*modeldsn*) keyword.

For more information about data security and data set profile considerations, see [Chapter 21, “Data security and authorization checking,”](#) on page 533.

MGMTCLAS



MGMTCLAS specifies the user-desired management class that replaces the source management class as input to the ACS routines. You must have the proper RACF authority for the management class specified. The keyword itself does not require RACF authorization.

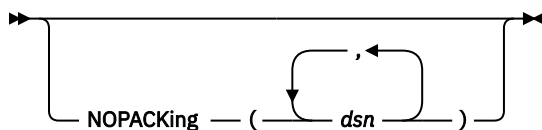
NULLMGMTCLAS/NMC specifies that the input to the ACS routines is a null management class rather than the source data set's management class.

MGMTCLAS and NULLMGMTCLAS are mutually exclusive; you cannot specify these keywords together.

Note:

1. All SMS-managed data sets specified in the BYPASSACS keyword are assigned the specified management class because the ACS routines are not invoked. Non-SMS-managed data sets do not have a management class.
2. See [“Assignment of class names by using the RESTORE and COPY commands”](#) on page 496 for information about the assignment of class names using the copy function.
3. If DFSMSdss physical data set copy is used to copy the parts of a multivolume data set, the BYPASSACS(*dsn*) keyword is specified, and a different management class is specified on the MGMTCLAS keyword on each invocation of DFSMSdss, the user will be unable to recatalog the multivolume data set because of the mismatching management classes of the pieces. DFSMSdss is unable to determine at the time of copying a part of the data set, whether or not the management class specified conflicts with one specified on a previous invocation of DFSMSdss. If the BYPASSACS(*dsn*) keyword is not specified, the correct management class should be chosen by the ACS routines.

NOPACKING



NOPACKING specifies that DFSMSdss is to allocate the target data set only to the same or like device types as the source data set is allocated on and that DFSMSdss is to use track level I/O to perform data movement. This results in an exact track-for-track image of the source data set on the target volume.

dsn

Specifies the fully or partially qualified names of a PDS to be processed.

NOPACKing is only valid with a PDS. If REBLOCK is specified, REBLOCK is ignored for the data set. If the data set is specified with CONVERT(PDSE()), NOPACKing is ignored for the data set.

A PDS copied or restored by using NOPACKing is not be compressed during data movement.

NOPACKing can be used for a damaged PDS that is currently usable by an application but would be made unusable by compression or other rearrangement of the physical layout of the data.

NOTIFYCONCURRENT

See [“CONCURRENT”](#) on page 404.

NULLMGMTCLAS

See [“MGMTCLAS”](#) on page 344.

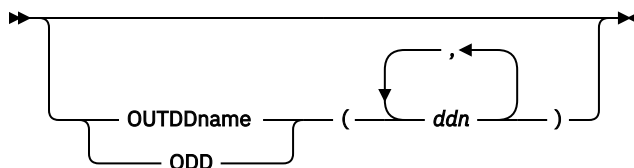
NULLSTORCLAS

See [“STORCLAS”](#) on page 353.

ONLYINCAT

See [“INCAT”](#) on page 340.

OUTDDNAME

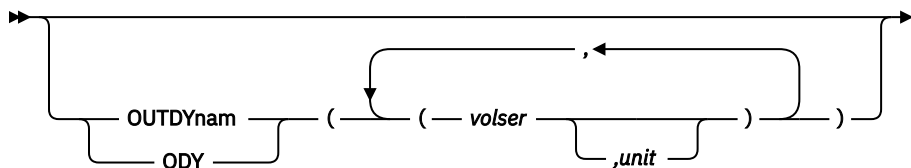


ddn

Specifies the name of the DD statement that identifies the output DASD volume. To assure correct processing, each of the DD statements corresponding to a DDNAME (*ddn*) must identify only one volume serial number. The volume serial number specified must be a valid DASD device; DD DUMMY is not supported. Only one volume is allowed for a full, tracks, or physical data set copy operations; one or more volumes are allowed for a logical data set copy operation. Multiple names in a data set copy must be separated by commas.

See the Note under OUTDYNAM for additional information.

OUTDYNAM



OUTDYNAM specifies that the output DASD volume is to be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*). Only one volume is allowed for a full, tracks or physical data set copy operation; one or more volumes are allowed for logical data set copy.

volser

Specifies the volume serial number of the volume.

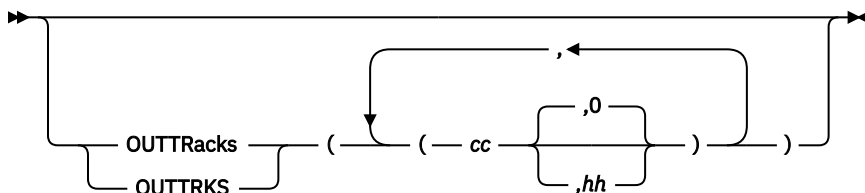
unit

Specifies the device type of the volume. This parameter is optional.

Notes for OUTDDNAME and OUTDYNAM Keywords

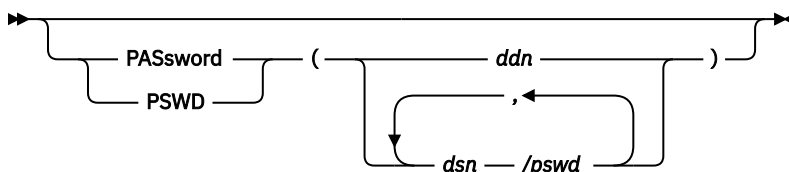
1. DFSMSdss now distinguishes between non-SMS and SMS volumes specified in the OUTDDNAME or OUTDYNAM keywords. For non-SMS allocations, only the volumes that are non-SMS are considered for allocation. Similarly, only SMS volumes are considered for SMS allocations.
2. The above distinction is also used 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 being done are counted. If there are no volumes that match the type of allocation (SMS volumes for SMS allocation, or non-SMS volumes for non-SMS allocation), processing proceeds with a null volume list.
3. OUTDDNAME and OUTDYNAM is always required for a COPY of a data set that is to be non-SMS managed even if a usable pre-allocated non-SMS target data set exists and is cataloged.

OUTTRACKS



OUTTRACKS specifies, for a tracks copy operation, the cylinder (cc) and head number (hh) on the output volume to which the tracks from the input volume are to be copied. If you do not specify OUTTRACKS operation, the tracks are copied to the same place on the output volume where they were on the input volume. The number of (cc, hh) combinations specified in the OUTTRACKS keyword must be the same as the number of (c1, h1, c2, h2) combinations specified in the TRACKS keyword.

PASSWORD



PASSWORD specifies the passwords that DFSMSdss is to use for password-protected data sets. (Password checking is bypassed for RACF-protected data sets.) This keyword is required only when:

- You do not have the required RACF DASDVOL or RACF data set access.
- The installation authorization exit does not bypass the checks.
- You do not want to be prompted for the password for VSAM data sets.

Note: You should specify the passwords for all data sets that do not have RACF protection but do have password protection. During processing, a utility invoked by DFSMSdss may have to prompt the operator for a password. You can control authorization checking by using the installation authorization exit.

Catalog passwords are not supported to facilitate disaster recovery operations, application data transfers, and data set migration. Catalog protection via an access control facility, such as RACF, is the preferred method of protection.

Passwords for a tracks copy operation are required only for the data sets on which the requested ranges fall.

ddn

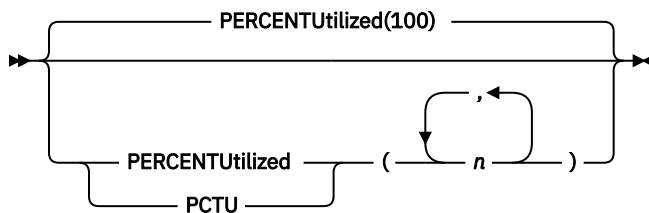
Specifies the name of the DD statement that identifies the sequential data set, or member of a partitioned data set, that contains data set names and their passwords. This data set must contain card-image records in DFSMSdss command syntax format.

dsn/pswd

dsn is a fully qualified data set name. *pswd* is its password. If no password follows the slash (/), *dsn* is treated as though it were *ddn*.

Printing of actual data set passwords specified in the input command stream is suppressed in the SYSPRINT output.

When a system utility is being used to perform the DFSMSDss copy operation, the user must supply the password for each password-protected data set selected or have the proper RACF data set access authority.

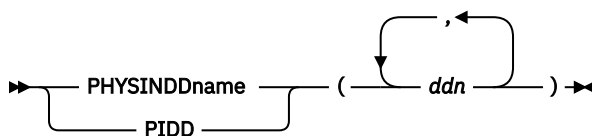
PERCENTUTILIZED

PERCENTUTILIZED specifies that DFSMSDss must stop allocating data sets to the target volumes when the allocated space reaches *n* percent of the total space on the target volume. The default value is 100. Specify more than one *n* if you have more than one target volume (for instance, a volume for overflow). If there are more target volumes than you have values in this keyword, the last value is used for the remaining target volumes. This keyword is used as a guide only and might not be precise for all situations.

If the output volume is an extended address volume and DFSMSDss is allocating non-VSAM or unsupported VSAM data sets, PERCENTUTILIZED specifies that DFSMSDss must stop allocating data sets to the target volumes when the allocated space reaches *n* percent of the track-managed space on the target volume.

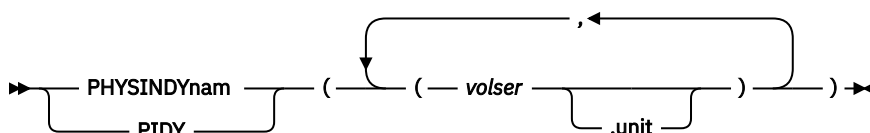
Note:

1. PERCENTUTILIZED is ignored when the target data set is preallocated or if you do not specify an output volume.
2. PERCENTUTILIZED is not supported in an SMS environment.

PHYSINDD

PHYSINDD may be specified to request that DFSMSDss perform a physical copy. It specifies a ddname that describes an input volume to be used for the copy operation. Only one ddname may be specified. The device described by the ddname must be the same type as the output device specified on the OUTDD or OUTDYNAM keyword.

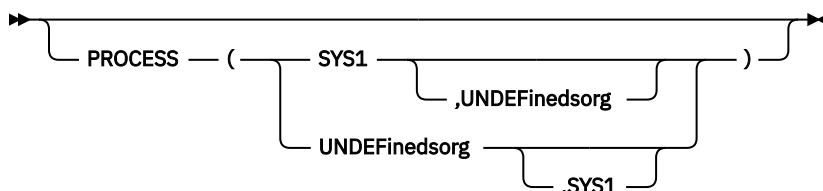
PHYSINDD may be abbreviated to PIDD.

PHYSINDYNAM

The PHYSINDYNAM keyword specifies the input volume that is to be dynamically allocated to a physical copy or dump operation. A nonspecific volume serial number cannot be specified by using an asterisk (*). Only one volume may be specified for a FULL, TRACKS, or DATASET COPY. The device described by the volser must be the same type as the output device specified on the OUTDD or OUTDYNAM keyword.

PHYSINDYNAM may be abbreviated to PIDY.

PROCESS



SYS1

specifies that DFSMSdss is to allow data sets with a high-level qualifier of SYS1 to be copied to a preallocated target and that SYS1 data sets can be deleted and uncataloged. SYS1.VVDS and SYS1.VTOCIX data sets cannot be copied, deleted, or uncataloged. To specify PROCESS(SYS1), RACF authorization may be required.

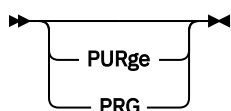
UNDEFInedsorg

PROCESS also specifies that DFSMSdss allow data sets with undefined data set organizations to be copied to an unlike target with a larger capacity. Refer to [Table 24 on page 366](#) and [Table 25 on page 368](#) for the action taken by DFSMSdss.

Note: Even though the data is being copied to a device with a larger track capacity, the data may not fit on the output device. For example, if the source device is a 3380, and 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 the message ADR366W (Invalid Track Format) is issued.

For more information about RACF authorization, see [Chapter 5, "Protecting DFSMSdss functions," on page 35](#).

PURGE



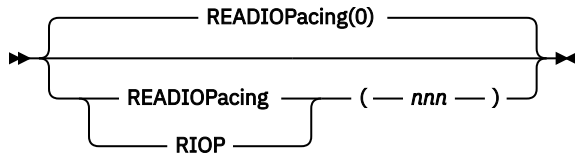
PURGE specifies that unexpired data sets, which reside on the target volume, can be overlaid for a full or track copy operation. If you do not specify PURGE and unexpired data sets exist on the target volume, the copy operation fails.

For data set copy operations, PURGE specifies that unexpired source data sets can be deleted after they have been successfully copied. PURGE is only valid with the DELETE keyword.

Note: You must specify PURGE for a full volume copy operation if the VVDS name on the target volume does not match the target volume serial number (*volser*). This procedure applies to volumes that are created by using full volume copy in conjunction with one of the following conditions:

- When DUMPCONDITIONING is specified
- When you do not specify COPYVOLID or DUMPCONDITIONING

READIOPACING



READIOPACING specifies the pacing (that is, I/O delay) to be used for DFSMSdss DASD read channel programs. You can use this keyword to allow more time for other applications to complete I/O processing. DFSMSdss waits the specified time before issuing each channel program that reads from DASD.

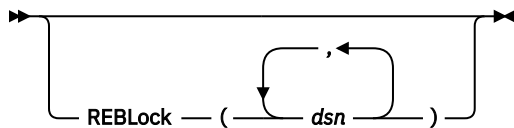
nnn

Specifies the amount of time in milliseconds. The maximum delay that can be specified is 999 milliseconds.

Note:

1. If READIOPACING is not specified, there is no I/O delay.
2. The additional wait time does not apply to error recovery channel programs.
3. READIOPACING does not apply to concurrent copy I/O.

REBLOCK



REBLOCK specifies that DFSMSdss is to reblock one or more of the selected sequential or partitioned data sets.

dsn

Specifies the fully or partially qualified names of a sequential or partitioned data set to be copied and reblocked.

The REBLOCK keyword is ignored for:

- Unmovable data sets
- Data sets with record format of U (except for partitioned load modules)
- Data sets with a record format of V, VS, VBS, or F
- Partitioned data sets with note lists (except for partitioned load modules)
- Partitioned data sets that are also specified in the NOPACKING keyword
- Compressed data sets in zEDC format.
- Encrypted format data sets

Additionally, both the installation options exit and the installation reblock exit can override the specification of the REBLOCK keyword. The installation options exit can specify that no data set is to be reblocked. The installation reblock exit can specify whether a given data set is to be reblocked.

Some sequential and partitioned data sets have an attribute indicated in the VTOC making them capable of being reblocked. These data sets may be automatically reblocked by DFSMSdss independent of the REBLOCK keyword.

When copying partitioned load modules to an unlike device, DFSMSdss uses IEBCOPY with COPYMOD specified. This may result in a reblocked data set. When copying to a like device, IEBCOPY with COPY specified is used.

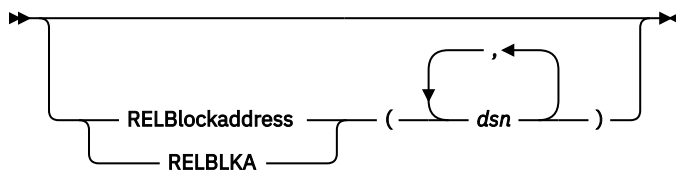
DFSMSDss uses DASD calculation services to determine the optimal block size for the target. The reblocking method used, DFSMSDss or DASD calculation services, is presented to the installation reblock exit.

Note: For source data sets that are defined with RECFM=VB, DFSMSDss may not be able to accurately calculate the required amount of space when REBLOCK is coded. You can remove REBLOCK(**) from your parameters to allow the COPY to proceed without reblocking the data set. If this is not an option you can refer to “Setting the percentage to overallocate target data set space (OW27837)” on page 228 and use the ADRPATCH command to request that DFSMSDss overallocate the VB data set target so the REBLOCK is successful.

RECATALOG

See “CATALOG” on page 318.

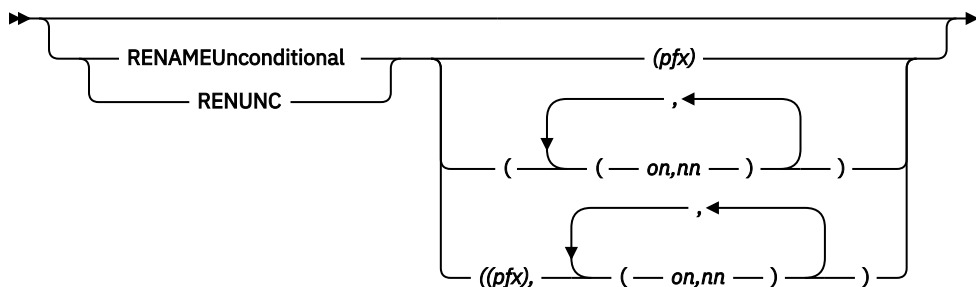
REBLOCKADDRESS



RELBLOCKADDRESS identifies the direct data sets (BDAM) whose names match the fully or partially qualified names specified (*dsn*). These direct data sets are organized by relative block address instead of TTR and are to be copied block by block. DFSMSDss updates the block reference count (the relative position of the physical record as stored on its track) of dummy records. This keyword applies only to direct data sets with fixed record formats and without standard user labels.

Restriction: If the data set is actually organized by TTR, the data set might become unusable.

RENAMEUNCONDITIONAL



RENAMEUNCONDITIONAL specifies that the data set must be copied with the new name, regardless of whether the data set exists on DASD with the old name. If the data set exists on the target volume with the new name and the REPLACEUNCONDITIONAL keyword is not specified, an error message is issued, and the data set is not copied.

pfx

Specifies the prefix used to replace the first-level qualifier of the data set name. It is optional, but if specified, must be the first parameter in the list of subkeywords. The prefix is used only if the (*on,nn*) parameters are not specified or the old name filters do not match the data set name.

on

Specifies the old name to be used as a filtering criterion to check if it matches the data set name.

nn

Specifies the new name to be used to derive the new data set name when the data set name matches the corresponding old name filtering criterion.

The syntax rules for *pfx* (prefix), *on* (old name), and *nn* (new name) are the same as in the RENAME keyword in a restore operation.

Note:

1. If the RENAMEU keyword is specified in conjunction with the REPLACE keyword, only one of the keywords take effect for any particular data set. The RENAMEU keyword takes precedence over the REPLACE keyword. If a source data set name matches the RENAMEU criteria, then rename processing will be performed and replace processing will not be performed. If a preallocated target data set exists with the new name as chosen by the rename criteria, then the copy fails even if the REPLACE keyword was specified. If you want to replace a preallocated target with the new name, specify the REPLACEUNCONDITIONAL keyword. If a source data set name does not match the rename criteria and a preallocated target data set with the source name exists, the preallocated target data set is replaced.
2. If CICSVRBACKUP is also specified, DFSMSdss uses the CICSVR-generated new name instead of the new name that you specified. See [“CICSVRBACKUP” on page 319](#) for more information.
3. You can have up to 255 entries using RENAMEUNCONDITIONAL. You cannot go beyond this limit by trying to use FILTERDD. FILTERDD can not be used with RENAMEUNCONDITIONAL.
4. RENAMEUNCONDITIONAL is not supported for VSAM Alternate Index (AIX) data sets during a physical data set restore.
5. If the preallocated target is multi-volume and SMS, either ensure that the primary volume is specified in the output volume list or remove all output volume specifications.

For more information about renaming, see [“RENAME” on page 481](#).

REPLACE



REPLACE specifies that DFSMSdss is to search the target volumes for usable preallocated data sets. If a usable preallocated target data set is found, it is replaced with the source data set. If no preallocated target is found, DFSMSdss attempts to allocate a data set.

DFSMSdss searches for preallocated data sets as follows:

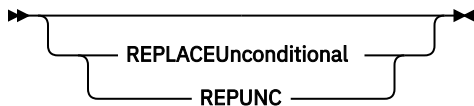
- For SMS-managed data sets, DFSMSdss first searches in the standard order of search for a catalog entry for the data set.
- For VSAM data sets that are not SMS-managed, DFSMSdss searches the output volumes for preallocated data sets. If no output volumes are specified, DFSMSdss searches for a catalog entry for the data set.
- For Non-VSAM data sets that are not SMS-managed, DFSMSdss searches the output volumes for preallocated data sets. If no output volumes are specified, DFSMSdss searches for a catalog entry for the data set.
- If no preallocated target is found, DFSMSdss attempts to allocate a data set. DFSMSdss invokes the ACS routines to determine if the data set should be SMS managed. If it should be SMS managed, then allocation is done according to the SMS constructs. If the data set should not be SMS managed, the output volumes specified are used. If allocation is successful, the data set is copied.

Note:

1. If REPLACE is specified with the COPY command:
 - The SMS constructs already associated with the preallocated target data set remain the same
 - The CA Reclaim attribute already associated with the preallocated target data set remains the same.
2. CATALOG and RECATALOG are ignored for preallocated data sets.
3. If the source data set is an extended-addressable VSAM data set, then the target must also be an extended-addressable VSAM data set.

4. The target data set name must match the source data set name. REPLACEUnconditional must be specified to replace a target data set that has a name matching the rename criteria.
5. The REPLACE and REPLACEUnconditional keywords can not be specified together.
6. If the RENAMEU keyword is specified in conjunction with the REPLACE keyword, only one of the keywords take effect for any particular data set. The RENAMEU keyword takes precedence over the REPLACE keyword. If a source data set name matches the RENAMEU criteria, then rename processing will be performed and replace processing will not be performed. If a preallocated target data set exists with the new name as chosen by the rename criteria, then the copy fails even if the REPLACE keyword was specified. If you want to replace a preallocated target with the new name, specify the REPLACEUNCONDITIONAL keyword. If a source data set name does not match the rename criteria and a preallocated target data set with the source name exists, the preallocated target data set is replaced.
7. If the source data set is a large format sequential data set, but the preallocated target is not a large format sequential data set, the preallocated target data set will be used and turned into a large format sequential data set as long as it has enough allocated space for the source data set.

REPLACEUNCONDITIONAL



REPLACEUNCONDITIONAL specifies that DFSMSdss is to search the target volumes for usable preallocated data sets. If a usable preallocated target data set is found, it IS replaced. When used with the RENAMEUnconditional keyword, usable preallocated data sets with the new name are replaced. When used without the RENAMEUnconditional keyword, usable preallocated data sets with the same name as the source data set are replaced. If no preallocated target is found, DFSMSdss attempts to allocate a data set. The REPLACE and REPLACEUnconditional keywords can not be specified together.

See the REPLACE keyword description for information about how target volume selection is performed.

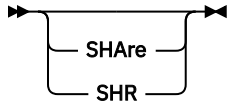
Note:

1. If REPLACEUNCONDITIONAL is specified with the COPY command:
 - The SMS constructs already associated with the preallocated target data set remain the same. If the preallocated target data set is scratched and reallocated, the SMS constructs used are those returned by the ACS routines for the source data set name.
 - The CA Reclaim attribute already associated with the preallocated target data set remain the same.
2. CATALOG and RECATALOG are ignored for preallocated data sets.
3. If the source data set is an extended-addressable VSAM data set, then the target must also be an extended-addressable VSAM data set.
4. If the source is a large format sequential data set, but the preallocated new name target is not a large format sequential data set, the preallocated target will be used and turned into a large format sequential data set as long as it has enough allocated space for the source data set. If the preallocated new name target does not have enough allocated space, it will be scratched and reallocated as a large format sequential data set with enough space for the source data set. If the name of the preallocated target is the same as the source data set, then see the REPLACE keyword.

SELECTMULTI

Refer to the description of SELECTMULTI in [“LOGINDYNAM” on page 342](#).

SHARE



SHARE specifies that DFSMSdss is to share the data sets to be copied for read access with other programs.

SHARE and FULL are mutually exclusive; you cannot specify these keywords together.

Do not specify DELETE if you specify SHARE. You must have exclusive control over data sets that are to be deleted; SHARE does not require such exclusive control.

Note: Unlike the RESTORE command, the COPY command honors the SHARE keyword for VSAM data sets. However, the SHARE keyword is only honored for VSAM data sets that were defined with share options other than (1,3) or (1,4).

Specifying the SHARE keyword does not cause DFSMSdss to honor the share options that are defined for VSAM data sets. For VSAM data sets that are defined with share options other than (1,3) or (1,4), specifying the SHARE keyword allows other programs to obtain read access. It does not, however, allow write access to the data sets while they are being copied. For VSAM data sets that are defined with share options (1,3) or (1,4), neither read access nor write access by other programs is allowed while the data set is being copied.

The SHARE keyword is not honored for PDSEs that are targets of a COPY operation. The SHARE keyword is still honored for source PDSEs of a COPY operation. Exclusive control is obtained for a target PDSE even if the SHARE keyword is specified. Neither read nor write access by other programs are allowed while the PDSE is being copied to.

SPHERE

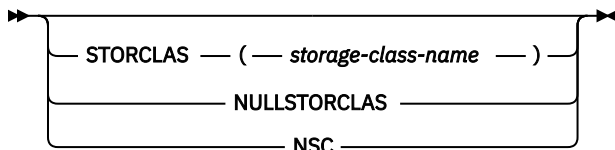


SPHERE specifies that, for any VSAM cluster copied, all associated AIX clusters and paths are to be copied. Individual names of sphere components do not need to be specified. Only the base cluster name is required. If output volumes are specified, the volumes on which the AIX clusters reside do not need to be specified.

Restrictions

- If the sphere is specified but the base cluster name is not, DFSMSdss processes only those components of the sphere whose names are specified.
- Do not specify the SPHERE keyword with the CICSVRBACKUP keyword.

STORCLAS



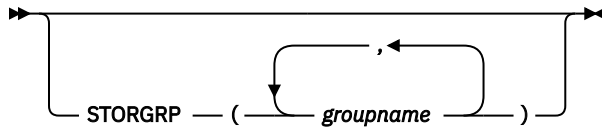
STORCLAS specifies the storage class that you want to replace (the source storage class) as input to the ACS routines. You must have the proper RACF authorization for the specified storage class. The keyword itself does not require authorization.

NULLSTORCLAS/NSC specifies that the input to the ACS routines is to be a null storage class rather than the source data set's storage class.

STORCLAS and NULLSTORCLAS are mutually exclusive; you cannot specify both keywords simultaneously. See [“Assignment of class names by using the RESTORE and COPY commands”](#) on page 496 for information about assigning class names using the copy function.

Note: If BYPASSACS(dsn) is specified, all data sets that pass the BYPASSACS selection criteria are guaranteed the specified storage class. The combination of NULLSTORCLAS and BYPASSACS(dsn) forces the selected data sets to be non-SMS-managed.

STORGRP



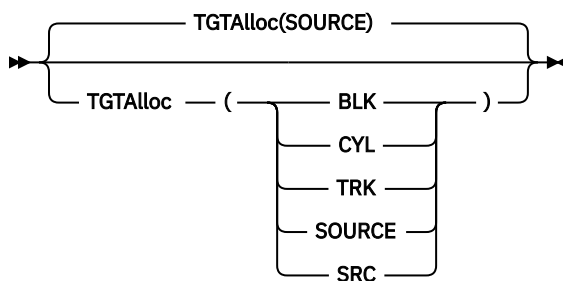
STORGRP specifies that all of the online volumes in the storage group be dynamically allocated. If a volume in the storage group is not online, that volume is not used for processing. Up to 255 storage group names may be specified. Specifying STORGRP with a storage group name is equivalent to specifying LOGINDYNAM with all the online volumes in the storage group included in the list.

You can specify the STORGRP keyword with the SELECTMULTI keyword, but STORGRP is mutually exclusive with the INDDname, INDYnam, LOGINDDname and LOGINDYnam keywords.

For more information, refer to [“Notes for LOGINDDNAME, LOGINDYNAM and STORGRP keywords”](#) on page 343.

See [“LOGINDYNAM”](#) on page 342 for a description of the SELECTMULTI keyword.

TGTALLOC



TGTALLOC specifies how DFSMSdss is to allocate the target data set.

BLK

by blocks

CYL

by cylinders

TRK

by tracks

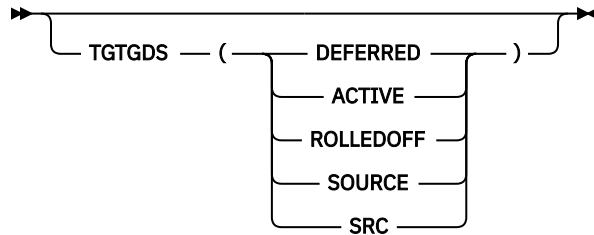
SOURCE/SRC

with the same space allocation type as that of the source data set

Note:

1. If the TGTALLOC keyword is omitted, the target allocation defaults to SOURCE.
2. If SRC is specified and if the source data set is allocated by track or if TRK is specified, then the final VSAM allocation might be different from the requested one because of VSAM allocation rules.
3. If BLK is specified for VSAM data sets, TRK is used instead. The final VSAM allocation might be different from the requested one because of VSAM allocation rules.

TGTGDS



TGTGDS specifies in what status, during a data set operation, that DFSMSdss is to place nonpreallocated SMS-managed GDG data sets:

DEFERRED

Specifies that the target data set is to be assigned the DEFERRED status.

ACTIVE

Specifies that the target data set is to be assigned the ACTIVE status, for example, rolled into the GDG base.

ROLLEDOFF

Specifies that the target data set is to be assigned the rolled-off status.

SOURCE/SRC

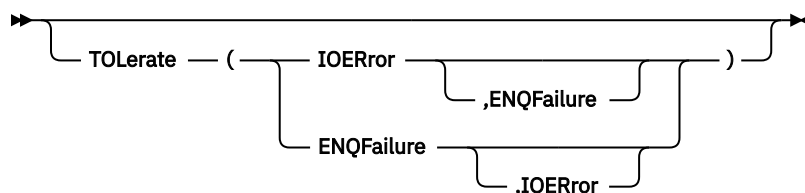
Specifies that the target data set is to be assigned the same status as that of the source data set.

Note:

1. If DELETE is specified without RENAMEUNCONDITIONAL and the source data set is an SMS-managed generation data set on a non-DFSMSdss dump conditioned volume, the TGTGDS keyword is ignored and the source GDS status is copied to the target.
2. If DELETE is specified without RENAMEUNCONDITIONAL and the source data set is an SMS-managed generation data set on a DFSMSdss dump conditioned volume, the DELETE keyword is ignored and the TGTGDS keyword will be honored.
3. The requested target status of generation data sets must not violate generation data group rules.

For more information about the default status when TGTGDS is not specified, see [“Restoring GDG data sets”](#) on page 88.

TOLERATE



TOLERATE specifies that DFSMSdss tolerates certain error conditions. For a copy operation (except of a user catalog or loadlib) in which a utility performs the copy, this keyword is ignored.

ENQFailure

specifies that source and target data sets are to be processed even though shared or exclusive access fails.

Note:

1. Unlike PDS data sets, PDSE data sets that are open for update cannot be copied even if TOL(ENQF) is specified.
2. If you must copy a PDSE data set and it must be open for update, convert the PDSE back to PDS and then copy the PDS data set with TOL(ENQF).

3. For a logical data set COPY command, TOL(ENQF) is ignored for HFS source data sets.

IOERROR

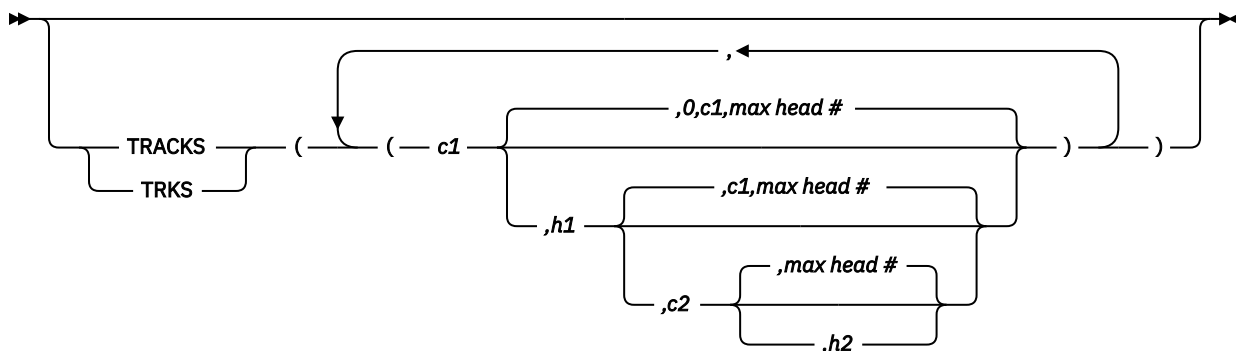
specifies that if the input volume can be opened, DFSMSdss is to continue copying even though permanent input errors (busout parity and equipment checks only) occur. DFSMSdss ends after 100 errors when this keyword is specified. The default ends on permanent input errors. On a data set copy in which a utility performs the copy, DFSMSdss ignores this keyword.

Note:

1. TOL(IOError) is ignored if CANCELerror is specified.
2. You cannot use the TOLERATE(ENQF) keyword when performing a logical copy operation with VSAM extended-format data sets.
3. You cannot use the TOLERATE(ENQF) keyword with a COPY FULL or COPY TRACKS operation.

For more information about using the TOL(ENQF) keyword, see [Chapter 22, “Data integrity—serialization,” on page 561.](#)

TRACKS



TRACKS specifies ranges of tracks to be copied (that is, a tracks copy).

c1,h1

Specifies the cylinder and head number of the beginning of the range. Specify hexadecimal numbers as X'c1' or X'h1'.

c2,h2

Specifies the cylinder and head number of the end of the range. Specify hexadecimal numbers as X'c2' or X'h2'. The c2 must be greater than or equal to c1. If c2 equals c1, h2 must be greater than or equal to h1.

You can enter X'FFFFFF' (or 268435455) as the high cylinder value. This causes DFSMSdss to choose the high cylinder value to be the end of the volume.

DFSMSdss verifies that the range is within the limits of the device. If you do not specify all four values for a range, DFSMSdss provides the missing values unless the omitted value causes a syntax error. No intervening values can be omitted. For example:

Specified Results

None

Syntax error

c1

c1,0,c1,maximum head number

c1,h1

c1,h1,c1,maximum head number

c1,h1,c2

c1,h1,c2,maximum head number

c1,,c2

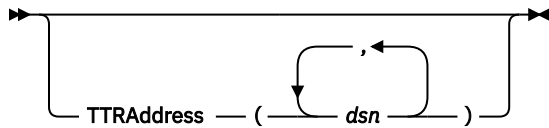
Syntax error

,h1

Syntax error

c1,h1,X'FFFFFF'

c1,h1,maximum cylinder number for the volume, maximum head number

Restriction: You cannot use the TRACKS keyword with the TOL(ENQF) keyword.For more information about using the TRACKS keyword, see [“Physical processing”](#) on page 26.**TTRADDRESS**

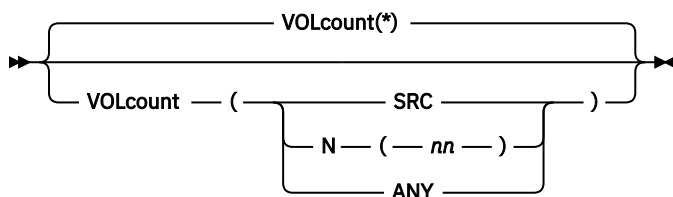
TTRADDRESS identifies the direct access data sets whose names match the fully or partially qualified names specified (dsn). These data sets are organized by TTR rather than by relative block addressing and are to be processed track by track. The target device track capacity must be equal to or greater than the source.

Guideline: The TTRADDRESS keyword takes precedence over the AUTORELBLOCKADDRESS keyword processing for the specified data sets (dsn).

UNCATALOG

UNCATALOG specifies that DFSMSdss is to uncatalog but not scratch successfully copied non-VSAM data sets that are currently cataloged on the source volume. Any non-SMS, non-VSAM data set that has a high-level qualifier of SYS1 cannot be uncataloged unless PROCESS(SYS1) is specified. UNCATALOG is ignored for VSAM data sets and SMS-managed non-VSAM data sets. UNCATALOG will also be ignored when processing data sets that reside on a DFSMSdss dump conditioned volume.

Note: Do not specify UNCATALOG with CONCURRENT, because after the concurrent copy operation has begun, the original data can still be updated.

VOLCOUNT

VOLCOUNT specifies the method DFSMSdss uses to determine the number of volumes (volume count) for allocating the SMS target data set for a copy operation of VSAM or non-VSAM data sets.

*** (asterisk)**

Specifies that DFSMSdss determine the volume count for allocation according to the following conditions:

- If the source data set is a single-volume data set, allocate one volume.
- The source data set is a multivolume data set, and one of the following conditions is present:
 - The OUTDDNAME or OUTDYNAM does not specify a list of volumes.
 - There are no SMS volumes in the list

DFSMSDss allocates the same number of volumes that were in the multivolume source data set.

- The source data set is a multivolume data set. It has an associated volume list (you specified the OUTDDNAME or OUTDYNAM keyword). DFSMSDss designates the volume count as the number of SMS volumes in the list.

DFSMSDss does not adjust the final number of candidate volumes after the allocation is complete.

The * (asterisk) is the default for this keyword.

SRC

Specifies that DFSMSDss *rely on the source volume count* to determine the number of volumes to allocate for the target data set as follows:

- If no output volume list is specified, DFSMSDss allocates the same number of volumes that the source data set had.
- If a volume list is specified through OUTDDNAME or OUTDYNAM, the volumes in the list that are SMS-managed must be in the same storage group, and the allocation must be directed to that storage group.

DFSMSDss does not adjust the final number of candidate volumes after the allocation is complete.

N(nn)

nn represents the number of volumes to be used for SMS data set allocation. Any value between 0 and 59 may be specified with the following conditions:

- If *nn* is not zero and a volume list is specified through OUTDDNAME or OUTDYNAM, DFSMSDss allocates either the number of SMS volumes in the volume list or *nn*, whichever is less.
- If *nn* is zero and a volume list is specified through OUTDDNAME or OUTDYNAM, DFSMSDss allocates either the number of SMS volumes in the volume list or the number of volumes that were allocated for the source data set, whichever is less.
- If a volume list is specified through OUTDDNAME or OUTDYNAM and there are no SMS volumes in the list, or there is no volume list, DFSMSDss allocates either the number of volumes used by the source data set or *nn*, whichever is *more*.

DFSMSDss does not adjust the final number of candidate volumes after the allocation is complete.

ANY

Specifies that DFSMSDss *use a maximum volume count* to allocate the SMS target data set as follows:

- DFSMSDss initially sets a volume count of 59 for the allocation.
- If the data set is allocated on more volumes than were used to allocate the source data set, DFSMSDss reduces the number of volumes used to the number of primary volumes needed to satisfy the allocation.
- If the data set is allocated on the same number or fewer volumes than were used to allocate the source data set, DFSMSDss reduces the number of volumes used to the number of volumes used for allocation of the source data set.

Note:

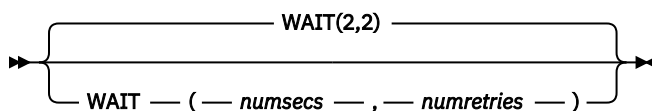
1. VOLCOUNT does not convert any of the following data sets to multivolume: PDS or PDSE data sets, single-volume data sets whose organization is undefined, or empty non-VSAM, single-volume data sets.
2. VOLCOUNT does not change the number of volumes for keyrange KSDS data sets.
3. Guaranteed space is not honored when VOLCOUNT(ANY) is used.

4. VOLCOUNT(ANY) does not support keyed VSAM data sets that have an imbedded index. If VOLCOUNT(ANY) is specified and a data set has an imbedded index, the data set is processed as if VOLCOUNT(*) were specified.
5. VOLCOUNT(ANY) does not support any type of striped data set (physical, sequential, extended, or VSAM). If VOLCOUNT(ANY) is specified and a data set is striped, the data set is processed as if VOLCOUNT(*) were specified.
6. When you specify VOLCOUNT(ANY), the &ANYVOL and &ALLVOL read-only variables are not available to the storage group ACS routine.
7. For nonguaranteed-space, striped VSAM data sets: The minimum number of volumes that DFSMSdss allocates is determined by the number of stripes, which is based on the STORCLAS sustained data rate (SDR). DFSMSdss does not consider the number of volumes in the output volume list or any of the VOLCOUNT specifications. If there are not enough enabled volumes in the STORGRP to support the SDR, DFSMSdss reduces the number of stripes. If there are excess volumes specified, those volumes become nonspecific (*) candidates.
8. For guaranteed-space, striped VSAM data sets: DFSMSdss allocates the number of volumes that are specified in the output list, regardless of the SDR. (To be striped, the SDR must be greater than zero.) The VOLCOUNT rules described above apply.

You can override VOLCOUNT keyword settings with the options installation exit routine.

For more information about overriding VOLCOUNT keyword settings, see [z/OS DFSMS Installation Exits](#).

WAIT



WAIT

Specifies to DFSMSdss the length of the wait in seconds, and the number of passes to be made through the list of selected data sets to obtain control of a data set for the COPY DATASET command.

numsecs

Specifies a decimal number (0–255) that designates the interval, in seconds, to wait before attempting another pass through the entire list of selected data sets.

numretries

Specifies a decimal number (0–99) that designates the number of passes to make through the list of selected data sets in an attempt to obtain control of a data set.

The default for *numsecs,numretries* is (2,2), which specifies two retries at 2-second intervals. If you do not want to wait for a resource, specify 0 for either numsecs or numretries.

For a data set copy operation the WAIT keyword has a different meaning when: (1) data sets are being serialized, (2) multiple data sets are being processed, and (3) WAIT(0,0) is not specified. In this case, DFSMSdss makes multiple passes through the list of data sets. On each pass, DFSMSdss processes the data sets that (1) can be serialized without waiting for the resource and (2) were not processed before. At the end of a pass, if none of the data sets can be processed without waiting for a resource, then, in the next pass, at the first occurrence of a data set that was not processed, a WAIT is issued. That data set and the remainder of the list are processed if possible.

The above procedure is repeated until all data sets are processed or the WAIT limits are reached. For example, if WAIT(3,10) is specified and five data sets are left to be processed, up to ten passes are made. On each pass, an unprocessed data set is waited upon for 3 seconds. Thus, only a 30-second maximum is ever waited, not 150 (5 times 3 times 10).

Note: The WAIT keyword does not control wait/retry attempts for system resources (such as the VTOC and the VVDS). For system resources, the default wait time is 3 seconds and the default retry count is 30. This results in a total wait time of 90 seconds. For information about controlling the wait/retry

attempts for system resources, see [“Controlling the wait/retry time for serialization of system resources \(PN11523\)”](#) on page 215.

WRITECHECK



WRITECHECK specifies that the data copied is to be verified for successful completion. This keyword increases the overall elapsed time.

Note:

1. On a data set copy in which a utility performs the copy, DFSMSdss ignores this keyword.
2. The WRITECHECK keyword is not supported for extended-format sequential data sets.

Data Integrity Considerations for Full or Tracks Copy Operation

For a full or tracks copy operation, DFSMSdss serializes the VTOC to preclude DADSM functions (such as ALLOCATE, EXTEND, RENAME, and SCRATCH) from changing the contents of the VTOC on the volume during the copy operation. Data sets are *not* serialized on these full or tracks operations. Therefore, some data sets might be opened by other jobs during the copy, resulting in copies of partially updated data sets. You can minimize this possibility by performing the copy when there is low system activity.

Full data integrity can only be guaranteed by performing copy operations by data set when TOL(ENQF) or SHARE are not specified.

Examples of Full and Tracks Copy Operations

The following examples are for FULL and tracks COPY operations.

Example 1: Data Set Copy Operation

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD1     DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//DASD2     DD       UNIT=3380,VOL=(PRIVATE,SER=222222),DISP=OLD
//SYSIN     DD       *
command input
/*
```

See command input in [“Example 1A: A Full Copy Operation”](#) on page 360 and [“Example 1B: A Tracks Copy Operation”](#) on page 360.

Example 1A: A Full Copy Operation

```
COPY INDDNAME(DASD1) OUTDDNAME(DASD2) -
ALLDATA(*) ALLEXCP CANCELERROR COPYVOLID
```

Example 1B: A Tracks Copy Operation

```
COPY TRACKS(1,0,1,15) INDDNAME(DASD1) -
OUTDDNAME(DASD2) CANCELERROR
```

The data from DASD volume 111111 is to be copied to DASD volume 222222. For the full copy operation (example 1A), all allocated space in sequential or partitioned data sets, and in data sets with a data set organization that is null, is copied (ALLDATA(*)). The volume serial number (VOLID) of the source volume

will be copied to the target volume. The result is that both volumes will have the same serial number (111111).

The preceding applies only to data sets that are not empty. For data sets that are empty, DFSMSdss copies all data within the allocated space (ALLEXCP). The copy operation is to be ended if a permanent read error occurs (CANCELERROR).

Example 2: A Tracks Copy with Track Relocation

The following example shows a tracks copy operation in which the contents of cylinder 1, tracks 0 through 14, on source volume 338000 are copied to cylinder 3, tracks 0 through 14, on target volume 338001. The operation stops if a permanent error occurs on the source volume (CANCELERROR). The data written to the target volume is to be verified (WRITECHECK).

```
//JOB2      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
COPY TRACKS(1,0,1,14) /* SOURCE TRACKS */ -
OUTTRACKS(3,0)        /* TARGET TRACKS */ -
INDYNAM(338000)        /* ALLOC VOL 338000 DYNAMICALLY */ -
OUTDYNAM(338001)       /* ALLOC VOL 338001 DYNAMICALLY */ -
CANCELERROR           /* STOP ON INPUT ERROR */ -
WRITECHECK            /* VERIFY DATA WRITTEN TO OUT VOL */
/*
```

Examples of Data Set Copy Operations

The following examples are for data set copy operations.

Example 1: A Data Set Move—Only Single Volume Data Sets

```
//JOB3      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
COPY DATASET(         -
INCLUDE(USER1.**))      /* FILTER ON DS W/1ST LEV Q USER1 */ -
BY(MULTI,=,NO))        /* FILTER ON SINGLE VOLUME */ -
INDYNAM(338000,338002) /* ALLOC VOL 338000, 338002 DYNAMICALLY */ -
OUTDYNAM(338001)       /* ALLOC VOL 338001 DYNAMICALLY */ -
DELETE
/*
```

Example 1 shows a data set copy operation in which all single-volume data sets with the first-level qualifier USER1 on the source volumes that are labeled 338000 and 338002 are copied to the target volume that is labeled 338001. All source data sets that are selected and successfully processed are deleted. The copied non-SMS, non-VSAM data sets are not cataloged.

Example 2: A Data Set Copy to Move Data Sets to a Single Volume—Device Conversion

```
//JOB4      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
COPY DATASET(         -
INCLUDE(USER1.**))      /* FILTER ON DS W/1ST LEV Q USER1 */ -
OUTDYNAM(338001)       /* ALLOC VOL 338001 DYNAMICALLY */ -
DELETE CATALOG FORCE -
TGTALLOC(SOURCE)
/*
```

Example 2 shows a data set copy operation in which all cataloged data sets with a first-level qualifier of USER1 (USER1.***) are consolidated on a single target volume that is labeled 338001. The data sets can

reside on multiple source volumes. The target volume can differ in device type from other volumes on which the source data sets reside. A few data sets might already reside on volume 338001. Data sets are cataloged in the standard search order. Expired source data sets are scratched and uncataloged after they are successfully moved to volume 338001. On volume 338001, they have the same allocation type (BLK, TRK, or CYL) that they had on the source volumes. FORCE is specified to include unmovable data sets.

Example 3: A Data Set Copy of a Multivolume Data Set

```
//JOB5      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//IVOL1     DD    UNIT=(SYSDA,2),VOL=SER=(VOL111,VOL222),DISP=SHR
//IVOL2     DD    UNIT=SYSDA,VOL=SER=VOL222,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
COPY DATASET(
  INC(USER.MULTI.VOLUME1)) /* SELECT THIS DATA SET      */ -
  INDD(IVOL1,IVOL2)        /* IDENTIFY INPUT VOLUMES */ -
  OUTDYNAM((338001),(338002),(338003)) /* DYNAM ALLOC VOLS */ -
  PCTU(80,80,80)           /* PERCENTUTIL = 80 PERCENT */ -
  RECATALOG(USERCAT2)
/*
```

Example 3 shows a data set copy operation in which a multivolume data set is copied to a set of target volumes labeled 338001, 338002, and 338003. The source data set is not deleted. The copied data set is cataloged in a new catalog, USERCAT2. The data set currently exists on multiple source volumes. Multiple output volumes are specified for overflow purposes. However, the output cannot be on more volumes than the source data set. These target volumes might already have data sets that reside on them. Space is left on these volumes to allow expansion of the data sets that remain on the volumes.

To include SELECTMULTI processing, you can change Example 3 as shown later in this section. The INCLUDE keyword specifies that you want to select all data sets on input volumes VOL111 and VOL222. The SELECTMULTI(ANY) keyword specifies that a cataloged data set that resides on volumes VOL111, VOL444, and VOL555 is copied even though VOL444 and VOL555 are omitted from the LOGINDD volume list.

```
//JOB5      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//IVOL1     DD    UNIT=(SYSDA,2),VOL=SER=(VOL111,VOL222),DISP=SHR
//IVOL2     DD    UNIT=SYSDA,VOL=SER=VOL222,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
COPY DATASET(
  INC(**)) /* SELECT ALL DATA SETS      */ -
  LOGINDD(IVOL1,IVOL2) /* IDENTIFY INPUT VOLUMES */ -
  OUTDYNAM((338001),(338002),(338003)) /* DYNAM ALLOC VOLS */ -
  SELECTMULTI(ANY)     /* PROCESS MISSING VOLUMES */ -
  PCTU(80,80,80)       /* PERCENTUTIL = 80 PERCENT */ -
  RECATALOG(USERCAT2)
/*
```

Example 4: A Data Set Copy with DELETE and RENAMEU Options

```
//JOB6      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
COPY DATASET(
  INCLUDE(USER1.**)) /* FILTER ON DS W/1ST LEV Q USER1 */ -
  OUTDYNAM((338001),(338002),(338003)) /* DYNAM ALLOC VOLS */ -
  DELETE           -
  RENAMEU(USER2)   -
  RECATALOG(USERCAT2)
/*
```

Example 4 shows a data set copy operation. All the data sets with the high-level qualifier USER1 that are in the standard search order are copied to the target volumes that are labeled 338001, 338002, and 338003. The copied data sets are renamed to the high-level qualifier USER2, followed by the second through last qualifiers of the old names. If data sets with the same name as the new names are on the target volumes or if the data sets are already cataloged in USERCAT2, they are not copied. The copied, expired data sets are deleted from the source volumes, uncataloged, and recataloged in the USERCAT2 catalog. This process moves the data sets from one set of volumes to another set of volumes, from one catalog to another catalog, and renames them.

Example 5: A Data Set Copy With REBLOCK Option

```
//JOB7      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU,PARM='UTILMSG=YES'
//SYSPRINT  DD    SYSOUT=A
//DISK      DD    UNIT=3380,VOL=(PRIVATE,,,,SER=338001),DISP=SHR
//DISK2     DD    UNIT=3390,VOL=(PRIVATE,,,,SER=339001),DISP=SHR
//SYSIN     DD    *
COPY DATASET(
  INCLUDE(**) )          /* INCLUDE ALL DATA SETS  */ -
  LOGINDDNAME(DISK)      /* INPUT VOLUME          */ -
  OUTDDNAME(DISK2)       /* OUTPUT VOLUME         */ -
  REBLOCK(**.USER1.**))
/*
```

Example 5 shows a data set copy operation in which all data sets on the 3380 source volume that is labeled 338001 are copied to the 3390 target volume that is labeled 339001. If data sets with the same name are on the target volume, they are not copied. The sequential and partitioned data sets that meet the filtering criteria that is specified in the REBLOCK keyword are reblocked on the target volume. The block size is selected by DFSMSdss, unless it is modified by the user reblock exit routine.

Example 6: A Data Set Copy to a Preallocated Target Data Set

```
//JOB8      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=A
//DASD1     DD    UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//DASD2     DD    UNIT=3380,VOL=(PRIVATE,SER=222222),DISP=OLD
//SYSIN     DD    *
COPY DATASET(
  INCLUDE(USER.TEST.DATA) ) -
  LOGINDDNAME(DASD1) -
  OUTDDNAME(DASD2) -
  REPLACE -
  DELETE
/*
```

Example 6 shows a data set copy in which a source data set (USER.TEST.DATA) allocated on volume 111111 and cataloged in catalog USERCAT is copied to a preallocated target data set (with the same name as the source) on volume 222222. The REPLACE keyword specifies that you want DFSMSdss to search the target volume for a usable preallocated data set. The data set is deleted from the source volume, if it is expired.

Example 7: Using the COPY Command to Convert to SMS

Example 7 shows non-SMS-managed volumes that are converted to SMS-managed volumes. It is a two-step process.

```
//JOB9      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
COPY -
  DS(INC(**)) -
    LOGINDYNAM ( -
      (338001) -
      (338002) -
    ) -
    STORCLAS(DB2PERF) -
    MGMTCLAS(DBBACKUP) -
    BYPASSACS(**) -
    DELETE -
    PURGE
/*
```

In step 1 of Example 7, all data sets on the non-SMS-managed volumes 338001 and 338002 are copied to SMS-managed volumes on the system. DELETE and PURGE processing is used to avoid duplicate catalog entries. ACS routines are not invoked to determine the target data set classes for this copy operation. Instead, users provide storage and management classes with the STORCLAS and MGMTCLAS keywords. In addition, users can suppress calls made to the ACS routines with the BYPASSACS(**) keyword. All data sets that are supported by SMS are given these new storage and management classes. All data sets that cannot be SMS-managed (for example, unmovable data sets) are not copied.

```
//JOB9      JOB   accounting information,REGION=nnnnK
//STEP2     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
COPY -
  DS(INC(**)) -
    LOGINDYNAM ( -
      (338001) -
      (338002) -
    ) -
    RENUNC(AUG0387)
/*
```

In step 2 of Example 7, all data sets on the non-SMS-managed volumes 338001 and 338002 are copied to SMS-managed volumes on the system. The RENUNC command is used to avoid duplicate catalog entries. The ACS routines select a target storage and management class for each data set. Those data sets that cannot be SMS-managed (storage class ACS routine returns a null storage class) are not copied because no output volume is specified. Each data set that is copied is given a new high-level qualifier (AUG0387) and automatically cataloged.

Example 8: A Data Set Copy Using CONVERT PDSE

```
//JOB2      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    *
COPY -
  DS(INC(USER.PDS.**)) -
    LOGINDYNAM ( -
      (338001) -
      (338002) -
    ) -
    CONVERT (PDSE(**)) -
    RENUNC (USER.PDS.**, USER.PDSE.**)
/*
```

Example 8 shows all data sets with the first two qualifiers of USER.PDS on non-SMS-managed volumes 338001 and 338002 are copied to SMS-managed volumes on the system. CONVERT PDSE is used to convert data sets to PDSE. The RENUNC keyword is used to avoid duplicate catalog entries. The ACS routines select a target storage and management class for each data set. Each data set that is copied and converted is given a new secondary qualifier (PDSE) and automatically cataloged.

Example 9: A Data Set Copy with CONCURRENT

```
//DSSJOB JOB   accounting information,REGION=nnnnK
//COPYSTEP EXEC PGM=ADRDSSU
//SYSPRINT DD   SYSOUT=*
//SYSIN   DD   *
COPY DATASET(INCLUDE(USER.LOG,USER.TABLE,USER.XREF)) -
  OUTDYNAM(0VOL01,0VOL02,0VOL03,0VOL08) -
  ALLDATA(*) ALLEXCP CONCURRENT -
  STORCLAS(BACKUP) RENAMEUNCONDITIONAL(USERX)
/*
```

Example 9 shows the required JCL for DFSMSDss to perform a logical data set copy using the concurrent copy feature. This job continues (with a warning message) if concurrent copy initialization fails.

Example 10: Copying a HFS using logical COPY

```
//DSSJOB JOB   accounting information,REGION=nnnnK
//COPYSTEP EXEC PGM=ADRDSSU
//SYSPRINT DD   SYSOUT=*
//SYSIN   DD   *
COPY DATASET(INCLUDE(OMVS.SB.MVS090.JV390.HFS, -
  OMVS.SB.MVS090.XML.HFS)) -
  RENAMEU((OMVS.SB.MVS090.JV390.HFS, -
  OMVS.SB.MVS030.ROOT.HFS) -
  (OMVS.SB.MVS090.XML.HFS, -
  OMVS.SB.MVS030.XML.HFS)) -
  NULLSTORCLAS BYPASSACS(**) -
  ALLDATA(*) ALLEXCP CANCELERROR -
  LOGINDDNAME(MVS091) OUTDDNAME(MVS023) -
  SHARE -
  WRITECHECK
/*
```

Example 10 shows the required JCL for DFSMSDss to perform a logical copy of a HFS.

ALLDATA and ALLEXCP Interactions

Table 24 on page 366 and Table 25 on page 368 describe the functions of the ALLDATA and ALLEXCP keywords during a data set copy to LIKE and UNLIKE devices, respectively.

COPY Command

Table 24. ALLDATA and ALLEXCP Interactions When Copying to LIKE Device. Read the first eleven columns to find the row that matches your situation. Read the last column to find what DFSMSdss does.

Y

Yes

N

No

X

Either

-

Not Applicable

1

Allocate and copy all the allocated space

2

Allocate and copy only the used space

3

Allocate and copy only one track

4

Allocate all the allocated space, and copy only the used space

5

Allocate all the allocated space, and copy only one track

6

Do not process the data set

Empty Data Set	EOF as First Record	DSORG	Load Module	ALLDATA(*)	ALLDATA(dsn)	ALLEXCP	NOPACKING	REBLOCK	Action
X	X	Undefined	-	X	X	X	-	X	1
N	X	Sequential ¹	-	N	N	X	-	X	2
N	X	Sequential ¹	-	Y	-	X	-	N	1
N	X	Sequential ¹	-	Y	-	X	-	Y	4
N	X	Sequential ¹	-	-	Y	X	-	N	1
N	X	Sequential ¹	-	-	Y	X	-	Y	4
N	X	Partitioned ²	X	N	N	X	N	X	2
N	X	Partitioned ²	X	N	N	N	Y	X	1
N	X	Partitioned ²	N	Y	-	X	N	X	4
N	X	Partitioned ²	N	-	Y	X	N	X	4
N	X	Partitioned ²	Y	Y	-	X	N	N	1
N	X	Partitioned ²	Y	Y	-	X	N	Y	4
N	X	Partitioned ²	Y	-	Y	X	N	N	1
N	X	Partitioned ²	Y	-	Y	X	N	Y	4
Y	N	Sequential ¹	-	X	X	N	-	X	6
Y	N	Sequential ¹	-	X	X	Y	-	N	1
Y	N	Sequential ¹	-	X	X	Y	-	Y	6
Y	Y	Sequential ¹	-	X	X	Y	-	N	1
Y	Y	Sequential ¹	-	X	X	Y	-	Y	3
Y	Y	Sequential ¹	-	N	N	N	-	X	5
Y	Y	Sequential ¹	-	Y	-	N	-	X	3
Y	Y	Sequential ¹	-	-	Y	N	-	X	3

Table 24. ALLDATA and ALLEXCP Interactions When Copying to LIKE Device. Read the first eleven columns to find the row that matches your situation. Read the last column to find what DFSMSDss does.

Y	Yes
N	No
X	Either
-	Not Applicable
1	Allocate and copy all the allocated space
2	Allocate and copy only the used space
3	Allocate and copy only one track
4	Allocate all the allocated space, and copy only the used space
5	Allocate all the allocated space, and copy only one track
6	Do not process the data set

(continued)

Empty Data Set	EOF as First Record	DSORG	Load Module	ALLDATA(*)	ALLDATA(dsn)	ALLEXCP	NOPACKING	REBLOCK	Action
----------------	---------------------	-------	-------------	------------	--------------	---------	-----------	---------	--------

Notes:

- ¹ When ALLDATA or ALLEXCP is specified for a sequential extended format 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.
- ² Partitioned data sets that have directories (whether or not there are empty members in the directory) are treated as not empty.

COPY Command

Table 25. ALLDATA and ALLEXCP Interactions When Copying to UNLIKE Device. Read the first twelve columns to find the row that matches your situation. Read the last column to determine what DFSMSDss does.

Y

Yes

N

No

X

Either

-

Not Applicable

1

Allocate the same number of tracks as the source, and make a track image copy.

2

Allocate and copy only the used space

3

Allocate and copy only one track

4

Allocate all the allocated space, and copy only the used space

5

Allocate all the allocated space, and copy only one track

6

Do not process the data set

Empty Data Set	EOF as First Record	BLKSIZE=0 Data Set	DSORG	Load Module	Target Tracksize > Source	PROCESS (UNDEF)	ALLDATA (*)	ALLDATA (dsn)	ALLEXCP	Action
X	X	X	Undefined	-	X	N	X	X	X	6
X	X	X	Undefined	-	N	Y	X	X	X	6
X	X	X	Undefined	-	Y	Y	X	X	X	1
N	X	N	Sequential ¹	-	X	-	N	N	N	2
N	X	N	Sequential ¹	-	X	-	Y	-	X	4
N	X	N	Sequential ¹	-	X	-	-	Y	X	4
N	X	Y	Sequential ¹	-	X	-	X	X	X	6
N	X	X	Partitioned ²	N	X	-	N	N	N	2
N	X	X	Partitioned ²	N	X	-	Y	-	X	4
N	X	X	Partitioned ²	N	X	-	-	Y	X	4
N	X	N	Partitioned ²	Y	X	-	N	N	X	2
N	X	N	Partitioned ²	Y	X	-	Y	-	X	4
N	X	N	Partitioned ²	Y	X	-	-	Y	X	4
N	X	Y	Partitioned ²	Y	X	-	X	X	X	6
Y	N	X	Sequential ¹	-	X	-	X	X	X	6
Y	Y	N	Sequential ¹	-	X	-	N	N	N	3
Y	Y	N	Sequential ¹	-	X	-	N	N	Y	5
Y	Y	N	Sequential ¹	-	X	-	Y	-	N	3
Y	Y	N	Sequential ¹	-	X	-	-	Y	X	5
Y	Y	Y	Sequential ¹	-	X	-	X	X	X	6

Table 25. ALLDATA and ALLEXCP Interactions When Copying to UNLIKE Device. Read the first twelve columns to find the row that matches your situation. Read the last column to determine what DFSMSdss does.

Y	Yes
N	No
X	Either
-	Not Applicable
1	Allocate the same number of tracks as the source, and make a track image copy.
2	Allocate and copy only the used space
3	Allocate and copy only one track
4	Allocate all the allocated space, and copy only the used space
5	Allocate all the allocated space, and copy only one track
6	Do not process the data set

(continued)

Empty Data Set	EOF as First Record	BLKSIZE=0 Data Set	DSORG	Load Module	Target Tracksize > Source	PROCESS (UNDEF)	ALLDATA (*)	ALLDATA (dsn)	ALLEXCP	Action
----------------------	---------------------------	-----------------------	-------	----------------	---------------------------------	--------------------	----------------	------------------	---------	--------

Notes:

- ¹ When ALLDATA or ALLEXCP is specified for an extended-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.
- ² Partitioned data sets that have directories (whether or not there are empty members in the directory) are treated as not empty.

COPYDUMP command for DFSMSdss

With the COPYDUMP command, you can make from 1 to 255 copies of DFSMSdss-produced dump data. The data to be copied, a sequential data set, can be on a tape or a DASD volume, and copies can be written to a tape or a DASD volume. If the dump data is produced from multiple DASD volumes by using a physical data set dump operation, you can selectively copy the data from one or more of those volumes.

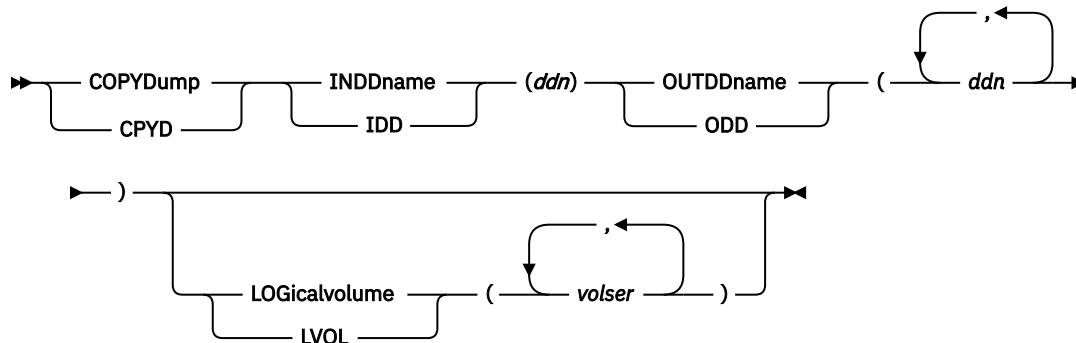
The COPYDUMP command cannot change the block size of the DFSMSdss dump data set. If you are copying a dump data set to a DASD device, the source block size must be small enough to fit on the target device.

You can use the COPYDUMP command to convert a sequential data set between extended format and non-extended format.

Note:

- Extra dump tapes can be used for such things as disaster recovery backup or distribution of dumped data (for example, a newly generated system).
- COPYDUMP is the only supported method for copying DFSMSdss dump data sets. Using a copy produced by any other method or utility as input to a RESTORE operation can produce unpredictable results.

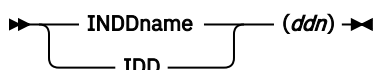
COPYDUMP syntax



Explanation of COPYDUMP command keywords

This section describes the keywords for the COPYDUMP command.

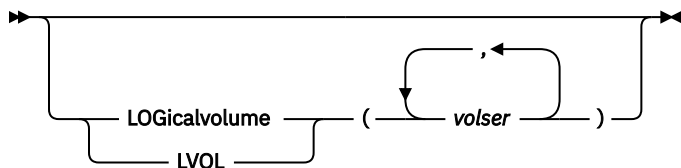
INDDNAME



ddn

Specifies the name of the DD statement that identifies the sequential data set to be copied. This data set can reside on one or more tapes, or on DASD volumes.

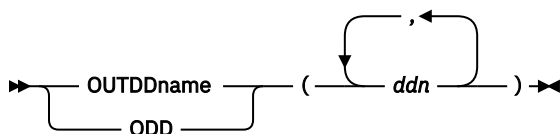
LOGICALVOLUME



volser

Specifies the source DASD volume serial number from which dumped data is to be copied. Omission of the LOGICALVOLUME keyword causes DFSMSdss to copy data from all logical volumes in the dump data set. This keyword is useful only if the data being copied was created by a physical data set dump operation from multiple DASD volumes. When copying a *logical* dump, LOGICALVOLUME is ignored.

OUTDDNAME



ddn

Specifies the name of the DD statement that identifies the output sequential data set. This data set can be on a tape or a DASD volume.

Examples of COPYDUMP operations

The following are examples of the COPYDUMP command.

Example 1: making two copies of a dump

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//BACKUP    DD       UNIT=3480,VOL=SER=TAPE05,DISP=OLD,
//           DSN=V111111.BACKUP
//COPY1     DD       UNIT=3480,VOL=SER=TAPE06,
//           DISP=(NEW,CATLG),DSNAME=V111111.BACKUP1
//COPY2     DD       UNIT=3480,VOL=SER=TAPE07,
//           DISP=(NEW,CATLG),DSNAME=V111111.BACKUP2
//SYSIN     DD       *
COPYDUMP -
  INDD(BACKUP) -
  OUTDD(COPY1,COPY2)
/*
```

In this example, two copies are to be made from a DFSMSdss dump tape (OUTDD(COPY1,COPY2)).

Example 2: copying a dump created by using physical data set processing

```
//JOB2      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//TAPE2     DD       UNIT=3480,VOL=SER=TAPE20,
//           LABEL=(1,SL),DISP=(OLD,KEEP),DSN=USER.BACKUP.REL3A
//OUTT2     DD       UNIT=3480,VOL=SER=TAPE21,
//           LABEL=(1,SL),DISP=(NEW,CATLG),DSN=USER.BACKUP.REL3A.A
//SYSIN     DD       *
COPYDUMP -
  INDD(TAPE2)           /* DUMP TAPE TO BE COPIED      */ -
  OUTDD(OUTT2)          /* NEW DUMP TAPE      */ -
  LVOL(338001)          /* SER NO OF VOL TO BE COPIED */
/*
```

Assume that a *physical* data set dump operation was used to create a dump tape, volume TAPE20. Also assume that source DASD volumes 338000, 338001, and so on were specified, resulting in VTOCs being used for data set selection. Only the dump data from DASD volume 338001 is to be copied (LVOL(338001)).

Example 3: copying a dump created by using logical data set processing

```
//JOB3      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//TAPE3     DD       UNIT=3480,VOL=SER=TAPE30,
//           LABEL=(1,SL),DISP=(OLD,KEEP),DSN=USER.BACKUP.REL3B
//OUTT3     DD       UNIT=3480,VOL=SER=TAPE31,
//           LABEL=(1,SL),DISP=(NEW,CATLG),DSN=USER.BACKUP.REL3B.A
//SYSIN     DD       *
COPYDUMP -
  INDD(TAPE3)           /* DUMP TAPE TO BE COPIED      */ -
  OUTDD(OUTT3)          /* NEW DUMP TAPE      */
/*
```

Assume that a *logical* data set dump operation was used to create a dump tape, volume TAPE30. All dump data is copied.

DEFRAG command for DFSMSdss

When you enter the DEFRAG command, DFSMSdss relocates data set extents on a DASD volume to reduce or eliminate free space fragmentation. You can specify which data sets, if any, should be excluded from relocation. When the DEFRAG operation completes, a summary report is created to show the before and after statistics of the volume. The report includes the fragmentation index, the size of the largest free space extent, and so on.

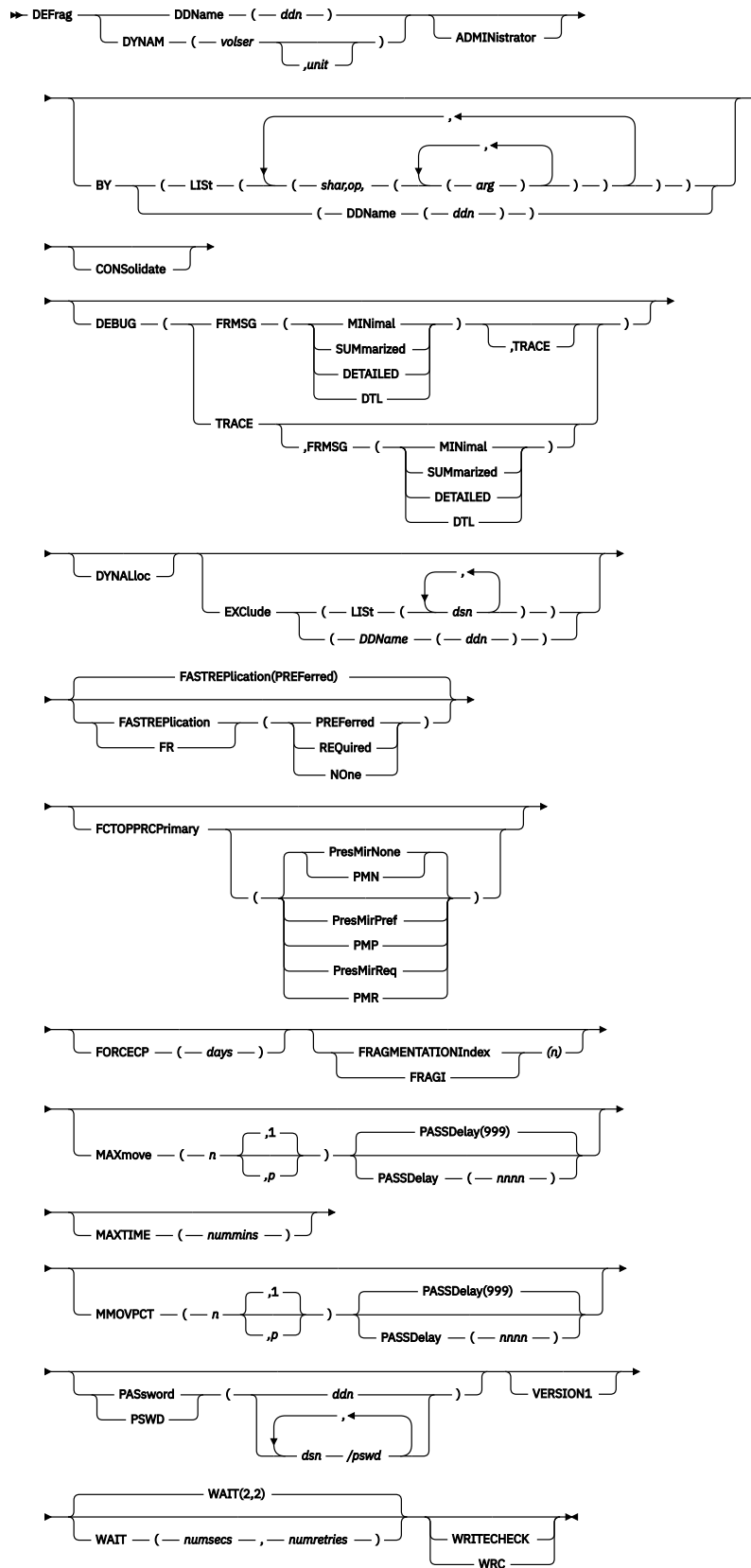
The amount of time it takes for a DEFRAG operation to complete depends on the size and fragmentation of the volume being processed. In general, larger or more fragmented volumes take longer to process.

Before using DEFRAG on a volume, you might want to use the CONSOLIDATE command to consolidate the extents of any multiple extent data sets on the volume.



Attention: Canceling the DEFRAG command is strongly discouraged, because doing so can damage data in numerous and unpredictable ways. Before using the DEFRAG command, consider how long the operation will take by evaluating the size and the fragmentation of the volume being processed. To limit the duration of the DEFFRAG operation, using the MAXTIME keyword is recommended.

DEFrag syntax



Explanation of DEFRAG command keywords

This section describes the keywords for the DEFRAG command.

ADMINISTRATOR



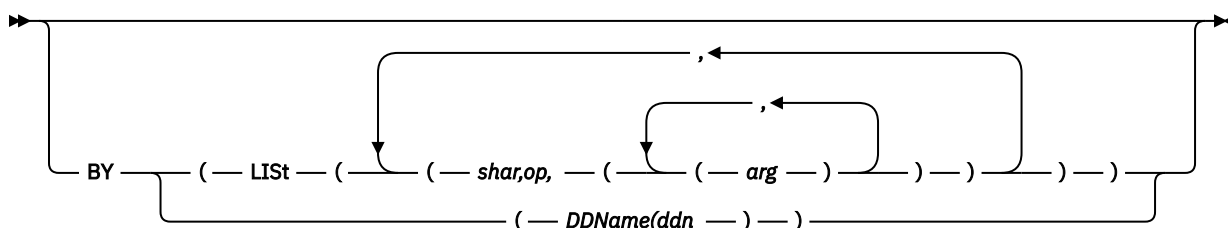
The ADMINISTRATOR keyword allows you to act as a DFSMSdss authorized storage administrator for the DEFRAG command. DFSMSdss-initiated access checking to data sets and catalogs is bypassed.

To use the ADMINISTRATOR keyword, all of the following must be true:

- The RACF FACILITY class is active
- The applicable FACILITY class profile is defined
- You have READ access to that profile.

For more information about using the ADMINISTRATOR keyword see [“ADMINISTRATOR keyword” on page 542](#).

BY



BY specifies data set filtering criteria.

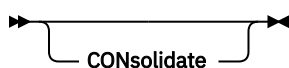
DDNAME(ddn)

Specifies the name of the data definition (DD) statement that identifies a sequential data set or member of a partitioned data set that contains the filtering criteria to use. This is in the form of card-image records, in DFSMSdss command syntax, that contain the BY keywords that are described below.

LIST((schar,op,((arg))))

Specifies data set filtering. To select the data set for inclusion in the DEFRAG operation, *all* BY criteria must be met. See [“Filtering by data set characteristics” on page 258](#) for a full discussion of *schar*, *op*, and *arg*.

CONSOLIDATE



CONSOLIDATE specifies that DEFRAG perform extent reduction by combining the extents of data sets with multiple extents. When you specify CONSOLIDATE, DFSMSdss consolidates data set extents where possible and then continues with normal free space defragmentation processing.

Note: The process of combining data set extents can cause the free space to be more fragmented than it was before the operation began. And, although DFSMSdss performs free space 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.



Attention: Use the CONSOLIDATE command instead of the DEFRAG command with the CONSOLIDATE keyword, which is obsolete. Otherwise, DFSMSdss issues an informational message

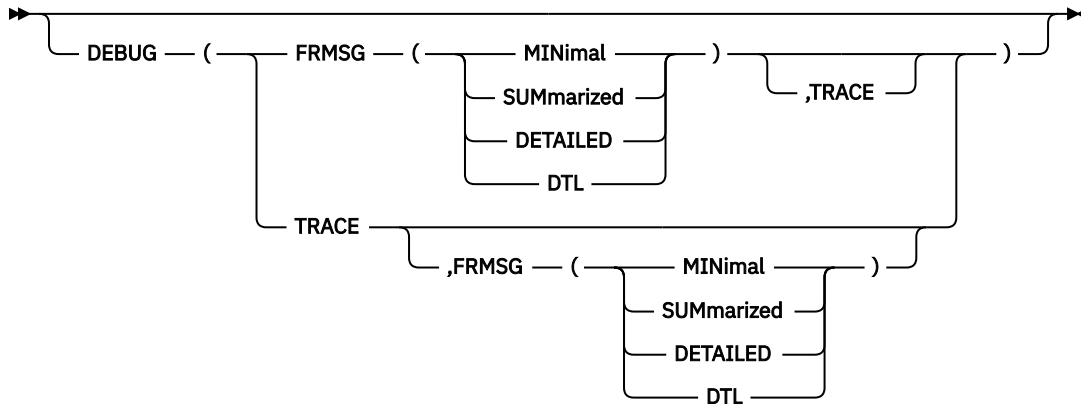
and runs the CONSOLIDATE command instead, for all files on the device not excluded through the EXCLUDE or BY filtering keywords. When CONSOLIDATE processing completes, DEFRAG then runs as expected. To have consolidation proceed as it did in previous releases, you must specify the VERSION1 keyword with the CONSOLIDATE keyword.

DDNAME

➤ DDName — (— *ddn* —) ➤

Specifies the name of the DD statement, *ddn*, that describes the volume to be processed.

DEBUG



FRMSG(SUMMARIZED)

Specifies that DFSMSdss is to issue an informational message with summary information. When applicable, summary information regarding ineligible volumes is provided in the message text. The following examples show the messages that are issued when you use this keyword:

Example 1

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
VOLUME SRCV01, RETURN CODE F
```

Return code F indicates that the volume does not support data set fast replication.

In this example, the `DEBUG(FRMSG(MINIMAL))` keyword provides the same level of informational message as when you specify the `DEBUG(FRMSG(SUMMARIZED))` or `DEBUG(FRMSG(DETAILED))` keyword.

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
VOLUME SRCV01, RETURN CODE 3
      VOLUME SRCV01 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 7 - VERSION 1 FC
      RELATION EXISTS
```

In this example, the `DEBUG(FRMSG(SUMMARIZED))` keyword provides the same level of informational message as when you specify the `DEBUG(FRMSG(DETAILED))` keyword.

When the `FASTREPLICATION(REQUIRED)` keyword is specified and the `DEBUG(FRMSG(MIN | SUM | DTL))` keyword is not specified, DFSMSdss still issues an informational message when a data set fast replication method cannot be used in DEFRAG operation. It is as if the `DEBUG(FRMSG(SUMMARIZED))` keyword had been specified.

FRMSG(DETAILED)

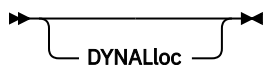
Specifies that DFSMSdss is to issue an informational message with detailed information. When applicable, detailed information about ineligible volumes is provided in the message text.

Note: DFSMSdss issues an informational message with summary of reasons why fast replication is ineligible for this processing when applicable.

Example 2

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR
VOLUME SRCV01, RETURN CODE 3
      VOLUME SRCV01 WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 7 - VERSION 1 FC
      RELATION EXISTS
```

In this example, the `DEBUG(FRMSG(DETAILED))` keyword provides the same level of informational message as when you specify the `DEBUG(FRMSG(SUMMARIZED))` keyword.

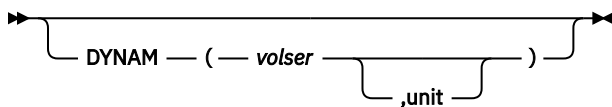
DYNALLOC

DYNALLOC specifies dynamic allocation, instead of enqueue, to serialize the use of data sets. The data sets whose extents are relocated are serialized throughout the DEFRAG operation. This allows cross-system serialization with the following considerations:

- The serialization is of value only when the dynamic allocation/JES3 interface is not disabled.
- Run time increases when you use the DYNALLOC keyword to serialize data sets (as opposed to enqueue) because overhead is involved in dynamic allocation and serialization across multiple processors.

- If you are running on a system using JES3 with MDS enabled and are not using multisystem GRS (or an equivalent function), you can use the DEFRAG command DYNALLOC keyword to provide serialization for data sets on shared DASD. However, not all data sets allocated within a JES3 environment are known to the global. The following are two cases where 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. You can prevent DEFRAG processing for these data sets 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.

DYNAM



DYNAM specifies that the volume to be processed is to be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*).

Using the DYNAM keyword instead of DD statements to allocate DASD volumes does not appreciably increase run time and permits easier coding of JCL and command input.

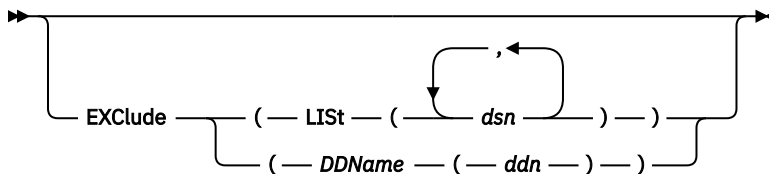
volser

Specifies the volume serial number of a DASD volume to be processed.

unit

Specifies the device type of a DASD volume to be processed. This parameter is optional.

EXCLUDE



LIST(dsn)

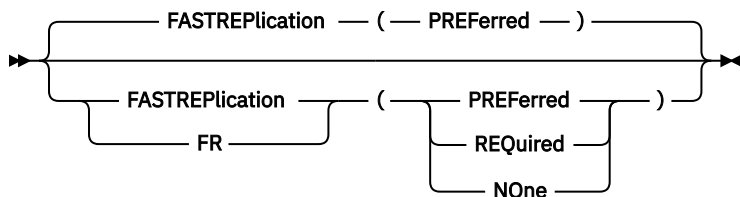
Specifies a fully or partially qualified name of a data set to be excluded from the DEFRAG operation. You can specify either a cluster or component name for VSAM data sets.

DDNAME(ddn)

Specifies the name of the DD statement that identifies a sequential data set or member of a partitioned data set, which contains the list of data sets to be excluded.

For additional information about specific types of data sets that must be placed in the EXCLUDE list if they are present on the volume being defragmented, see the [“Data sets excluded from DEFRAG or CONSOLIDATE processing”](#) on page 149.

FASTREPLICATION



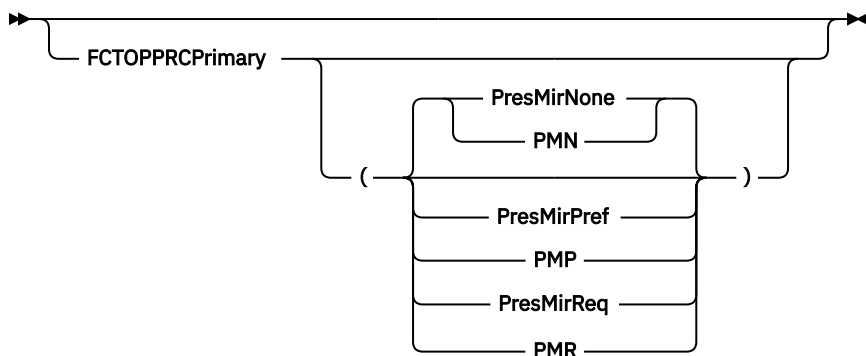
FASTREPLICATION specifies whether the use of fast replication is preferred, required, or not desired. This keyword applies to fast replication methods such as FlashCopy and SnapShot.

PREFERRED specifies that you want to use a fast replication method, if possible. If fast replication cannot be used, DFSMSdss completes the operation using traditional data movement methods. PREFERRED is the default (unless changed to NONE by the installation).

REQUIRED specifies that fast replication must be used. If fast replication cannot be used, DFSMSdss fails the operation. DFSMSdss issues an informational message regarding why a fast replication method cannot be used even if the DEBUG(FRMSG(MIN|SUM|DTL)) keyword is not specified.

NONE specifies that DFSMSdss not use fast replication methods for the operation. Instead, DFSMSdss uses traditional data movement methods to complete the operation.

FCTOPPRCPRIMARY



FCOTPPRCPrimary specifies that if FlashCopy is used to perform the copy operation, a Peer-to-Peer Remote Copy (PPRC) primary volume is allowed to become a FlashCopy target volume. Use the following sub-keywords to specify whether the device pair is allowed to go to duplex pending state if the target volume of the FlashCopy operation is a metro mirror primary device:

PRESMIRREQ

specifies that if the target volume is a Metro Mirror primary device, the pair must not go into a duplex pending state as the result of a FlashCopy operations.

PRESMIRPREF

specifies that if the target volume is a Metro Mirror primary device, it would be preferable that the pair does not go into a duplex pending state as the result of a FlashCopy operation. However, if a Preserve Mirror operation cannot be accomplished, the FlashCopy operation is still to be performed.

The PRESMIRPREF option is not valid for DS888x and newer, and compatible, storage subsystems. If the PRESMIRPREF option is specified and the volumes involved in the operation reside on DS888x or newer storage subsystems, the command results in a warning or error message. This is also true of the corresponding option provided in byte 33 (X'21') in the ADRUFO data area.

PRESMIRNONE

specifies that Preserve Mirror operation is not to be done, even if all of the configuration requirements for a Preserve Mirror operation are met. If the target specified is a Metro Mirror primary device, the

pair is to go into a duplex pending state while the secondary device is updated with the tracks to be copied. PRESMIRNONE is the default if you specify FCTOPPRCPrimary without a subkeyword.

Attention: When you specify FCTOPPRCPrimary or FCTOPPRCPrimary(PRESMIRNONE), the FlashCopy operation causes a PPRC primary volume to become a FlashCopy target volume. A Metro Mirror pair currently in full duplex state, goes into a duplex pending state when the FlashCopy relationship is established. When Metro Mirror completes the copy operation, the Metro Mirror pair goes to full duplex state. To prevent Metro Mirror pairs from going to duplex pending state during FlashCopy operation, you must specify FCTOPPRCPrimary(PRESMIRREQ).

For more information on PPRC options and volume states see: [z/OS DFSMS Advanced Copy Services](#).

Note:

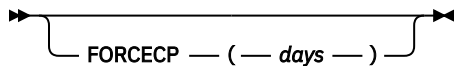
1. Using FCTOPPRCPRIMARY might require RACF authorization.
2. Do not specify the FCTOPPRCPrimary keyword with the FASTREPLICATION(NONE) keyword.
3. When FlashCopy is not used to perform the defrag operation, the FCTOPPRCPrimary keyword is ignored.
4. When FCTOPPRCPrimary is not specified or if the capability is not supported by the ESS, a PPRC primary volume is not eligible to become a FlashCopy target volume.

For additional information about RACF authorization see, [Chapter 5, “Protecting DFSMSdss functions,”](#) on page 35.

For additional information about RACF FACILITY class profiles see, [“Protecting DFSMSdss functions with RACF FACILITY class profiles”](#) on page 37.

For additional information about PPRC (PPRC-SYNC), PPRC-XD, and PPRC V2 see: [z/OS DFSMS Advanced Copy Services](#) and IBM Redbook *TotalStorage Enterprise Storage Server Implementing ESS Copy Services*.

FORCECP

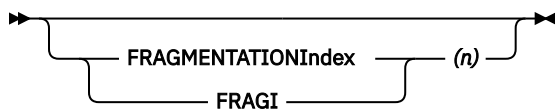


FORCECP specifies that extents of checkpointed data sets resident on the SMS volume or volumes can be moved. The checkpoint indication is left in place for IMS generalized sequential access method (GSAM) data sets, as the data set is still usable for a restart. The checkpoint indication is removed from all volumes for MVS checkpointed data sets, as the data sets are no longer usable for a restart.

days

Specifies a one-to-three digit number in the range of zero to 255. It also specifies the number of days that must have elapsed since the last referenced date before the data set can be defragmented.

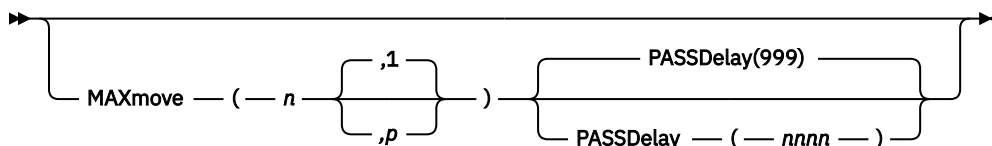
FRAGMENTATIONINDEX



FRAGMENTATIONINDEX specifies that the DEFRAG operation is to end if the fragmentation index is less than n , where n is a 1- to 3-digit number. DFSMSdss prefixes your number, n , with a decimal point. For example, 1 becomes .1, 999 becomes .999, 001 becomes .001, and so forth.

For additional information about the fragmentation index of a volume, see [“DEFRAG options”](#) on page 149.

MAXMOVE



n

Specifies a 1- to- 6 digit number that specifies that DFSMSdss is to attempt to assemble up to *n* free tracks in a contiguous area. If *n* is greater than the total number of free tracks on the volume, for example MAXMOVE(999999), a message is issued and the DEFRAG operation adjusts *n* to the number of free tracks.

p

Specifies a 1- to- 2 digit number that specifies that DFSMSdss is to make up to *p* passes in attempting to assemble the tracks. If *p* is not specified, for example MAXMOVE(99), only one pass is made, that is, MAXMOVE(99,1).

PASSDelay

specifies the time delay between the passes (*p*) specified in MAXMOVE (*n,p*). PASSDELAY is only meaningful if (*p*) is greater than one.

nnnn

Specifies a 1- to- 4 digit number from 0 to 9999 that specifies the time delay in milliseconds (1/1000 of a second). When MAXMOVE (*n,p*) is specified and PASSDELAY is not specified, a default PASSDELAY value of 999 (almost a second) is used to allow access to the volume between MAXMOVE passes.

DEFRAG processing attempts to move a fraction of the total free tracks during each pass. DEFRAG processing calculates an integer limit that is the larger of *n* divided by *p* or the value of 15. The limit is compared to the cumulative number of tracks moved after each extent is moved. If the limit is reached or exceeded, the current pass ends, and a new pass starts. Between each pass, the DEFRAG function releases and re-obtains volume serialization. This action reduces the overall time that a DEFRAG operation makes a volume unavailable to other applications.

The DEFRAG operation may end before completing *p* passes. If the DEFRAG operation cannot assemble *n* contiguous free tracks without moving more than *n* tracks, DFSMSdss issues a message and ends the DEFRAG operation. If *n* contiguous free tracks exist, the DEFRAG operation still attempts to reduce the fragmentation of the volume if it can do so without relocating more than *n* tracks.

The operation or the current pass also ends when DEFRAG processing criteria (for example, FRAGI) are satisfied. If you specified more than one pass, only the current pass ends. DFSMSdss issues message ADR233W for the current pass when the FRAGI criteria has been met. The DEFRAG function continues to run and attempts to complete the specified number of passes. DFSMSdss issues message ADR233W for each pass that meets the FRAGI criteria. The function continues because activity on the volume between the DEFRAG passes may change the fragmentation index and the subsequent DEFRAG passes may further reduce the fragmentation of the volume.

In order to re-obtain volume serialization between DEFRAG passes, DFSMSdss uses a default value of WAIT(3,30). This means that DFSMSdss is to retry the volume serialization 30 times at 3 second intervals for a total wait time of 90 seconds. The DEFRAG operation issues a message and ends if DFSMSdss cannot obtain volume serialization; any remaining passes do not run. [“The WAIT option” on page 563](#) explains how to change the wait/retry values for the volume serialization if the default is not satisfactory.

If MAXMOVE is not specified, DFSMSdss tries to assemble a contiguous free area of a size equal to the total number of free tracks on the volume in a single pass. The DEFRAG function uses two methods to assemble free tracks on the volume. The first method attempts to assemble the largest contiguous amount of free space with a minimum amount of data movement. The second method attempts to assemble multiple groups of large areas of contiguous free space and generally moves more data around than the first method. When MAXMOVE is specified, only the first method is used during each pass.

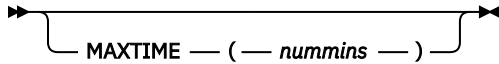
Note:

1. The MMOVPCT keyword is recommended instead of MAXMOVE when running DEFRAG on an EAV. The MMOVPCT will apply separately to the track-managed space and the cylinder-managed space.
2. MAXMOVE and MMOVPCT are mutually exclusive.

MAXTIME

Specifies the maximum time, in minutes, for the DEFRAG operation to complete. MAXTIME is checked after each data set is processed. When the MAXTIME value is reached, the DEFRAG operation ends.

MAXTIME



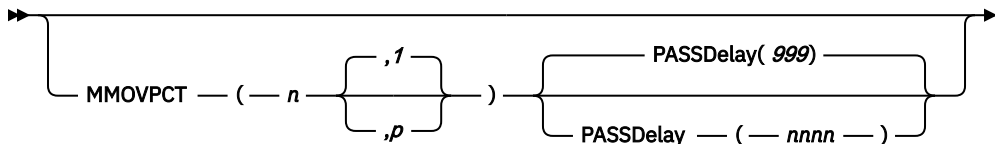
nummins

Specifies the maximum number of minutes (0-9999 in decimal) a DEFRAG operation can run. A value of 0 is ignored.

Note:

1. The actual elapsed time of the DEFRAG operation might be slightly longer than the MAXTIME value; this value is checked only after each data set is processed.
2. The CONSOLIDATE and the MAXTIME keywords are mutually exclusive. If you specify the CONSOLIDATE keyword when MAXTIME is specified, the DEFRAG command terminates and DFSMS issues syntax error message ADR139E.

MMOVPCT



n

This 1- to 3-digit number specifies that DFSMSdss is to attempt to assemble up to *n*% of tracks as free tracks in a contiguous area. If *n*% is greater than the total number of free tracks available to the task, the DEFRAG operation resets this value to the number of free tracks available to the task.

p

This 1- or 2-digit number specifies that DFSMSdss is to make up to *p* passes in attempting to assemble the tracks. If *p* is not specified, only one pass is made.

PASSDelay

Specifies the time delay between the passes (*p*) specified in MMOVPCT(*p*,*n*). PASSDELAY is meaningful only if (*p*) is greater than one.

nnnn

This 1- to 4-digit number specifies the time delay in milliseconds (1/1000 of a second). When MMOVPCT (*n*,*p*) is specified and PASSDELAY is not specified, a default PASSDELAY value of 999 (almost a second) is used to allow access to the volume between MMOVPCT passes.

DEFRAG processing attempts to move a fraction of the free tracks during each pass. It calculates an integer limit that is the larger of *n*% of tracks divided by *p* or the value of 15. The limit is compared to the cumulative number of tracks moved after each extent is moved. If the limit is reached or exceeded, the current pass ends, and a new pass starts. Between each pass, the DEFRAG function releases and re-obtains volume serialization. This action can reduce the overall time in which a DEFRAG operation makes a volume unavailable to other applications.

The DEFRAG operation might end before completing *p* passes. If the DEFRAG operation cannot assemble *n*% contiguous tracks as free without moving more than *n*% tracks, DFSMSdss issues a message and ends

the DEFRAG operation. If $n\%$ contiguous free tracks exist, the DEFRAG operation continues to reduce the fragmentation of the volume, if it can do so without relocating more than $n\%$ tracks.

The operation or the current pass also ends when DEFRAG processing criteria (for example, FRAGI) are satisfied. If you specify more than one pass, only the current pass ends. DFSMSdss issues message ADR233W for the current pass when the FRAGI criteria has been met. The DEFRAG function continues to run and attempts to complete the specified number of passes. DFSMSdss issues message ADR233W for each pass that meets the FRAGI criteria. The function continues because activity on the volume between the DEFRAG passes might change the fragmentation index and the subsequent DEFRAG passes might further reduce the fragmentation of the volume.

To restore volume serialization between DEFRAG passes, DFSMSdss uses a default value of WAIT(3,30). This means that DFSMSdss is to retry the volume serialization 30 times at 3 second intervals for a total wait time of 90 seconds. The DEFRAG operation issues a message and ends if DFSMSdss cannot obtain volume serialization; any remaining passes do not run. [“The WAIT option” on page 563](#) explains how to change the wait and retry values if the defaults are not suitable for your installation.

If you do not specify MMOVPCT, DFSMSdss attempts to assemble a contiguous free area of a size equal to the total number of free tracks on the volume in a single pass. The DEFRAG function uses two methods to assemble free tracks on the volume. The first method attempts to assemble the largest contiguous amount of free space with a minimum amount of data movement. The second method attempts to assemble multiple groups of large areas of contiguous free space and generally moves more data than the first method. If you specify MMOVPCT, only the first method is used during each pass.

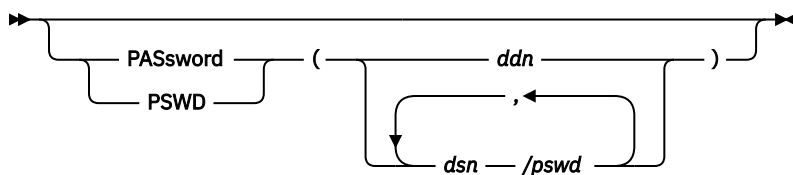
Note:

1. Use MMOVPCT, rather than MAXMOVE, when running DEFRAG on an extended address volume. The MMOVPCT value applies separately to the track-managed space and the cylinder-managed space.
2. MMOVPCT and MAXMOVE are mutually exclusive.
3. MMOVPCT and VERSION1 are mutually exclusive.

PASSDELAY

See [“MAXMOVE” on page 380](#) or [“MMOVPCT” on page 381](#).

PASSWORD



PASSWORD specifies the passwords that DFSMSdss is to use for password-protected data sets. (Password checking is bypassed for RACF-protected data sets.) This keyword must be specified when:

- You do not have the required RACF DASDVOL or RACF DATASET access.
- The installation authorization exit does not bypass the checks.
- You do not want to be prompted for the password for VSAM data sets.

Note: You should specify the passwords for all data sets that do not have RACF protection but do have password protection. During processing, a utility invoked by DFSMSdss may have to prompt the operator for a password. You can control authorization checking by using the installation authorization exit.

For VSAM data sets, password checking is done only at the cluster level.

ddn

Specifies the name of the DD statement that identifies the sequential data set, or member of a partitioned data set that contains data set names and their passwords. This data set must contain card-image records in DFSMSdss command syntax format.

dsn/pswd

dsn is a fully qualified data set name. *pswd* is its password. If no password follows the slash (/), *dsn* is treated as though it were *ddn*.

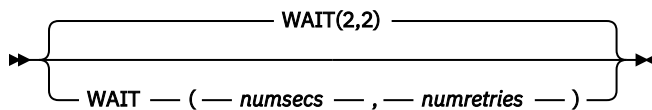
Printing of actual data set passwords specified in the input command stream is suppressed in the SYSPRINT output.

VERSION1

VERSION1 indicates that DFSMSdss is to use the DEFRAG command available with earlier releases of z/OS. DEFRAG with VERSION1 specified can support only volumes of 65,520 or fewer cylinders.

VERSION1

Note: VERSION1 is mutually exclusive with the MAXTIME and MMOVPCT keywords.

WAIT

WAIT specifies to the length of a wait in seconds and the number of retries to obtain control of a data set.

numsecs

Specifies a decimal number from 1 to 255 that specifies the interval, in seconds, between retries.

numretries

Specifies a decimal number (0–99) that specifies the number of times an attempt to gain control of a data set is to be retried.

The default for *numsecs*, *numretries* is (2,2), which specifies two retries at 2-second intervals. If you do not want to wait for a data set, specify 0 for either *numsecs* or *numretries*.

Note: The WAIT keyword does not control wait/retry attempts for system resources (such as the VTOC and the VVDS). For system resources, the default wait time is 3 seconds and the default retry count is 30. This results in a total wait time of 90 seconds.

For additional information about controlling the wait/retry attempts for system resources, see [“Controlling the wait/retry time for serialization of system resources \(PN11523\)”](#) on page 215.

WRITECHECK

WRITECHECK specifies that the data moved by the DEFRAG operation is to be verified for successful completion. This keyword increases the overall elapsed time.

Examples of DEFRAG operations

Example 1: a DEFRAG operation with excluded data sets

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD      DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//A1        DD       DSN=USER2.EXCLUDE,DISP=SHR
//SYSIN     DD       *
command input
/*
```

See command input in “[Example 1A: with the names of excluded data sets in the input stream](#)” on page 384 and “[Example 1B: with the names of excluded data sets in a data set](#)” on page 384.

Example 1A: with the names of excluded data sets in the input stream

```
DEFRAG DDNAME(DASD) -
      EXCLUDE(LIST(USER2.**.LIST,*.LOAD))
```

Example 1B: with the names of excluded data sets in a data set

```
DEFRAG DDNAME(DASD) -
      EXCLUDE(DDNAME(A1))
```

In examples 1A and 1B, DASD volume 111111 is defragmented. All data sets whose first and last qualifiers are USER2 and LIST, respectively, are to be excluded from this operation, as are data sets with two qualifiers whose second qualifier is LOAD. In example 1B, cataloged data set USER2.EXCLUDE contains a single card-image record with the following in columns 2 through 72:

```
USER2.**.LIST,*.LOAD
```

Example 2: a DEFRAG operation using a BY criterion

```
//JOB2      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD      DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//SYSIN     DD       *
DEFRAG DDNAME(DASD) /* VOLUME TO BE PROCESSED */ -
BY(LIST(REFDT LT *,-1)) /* DATE LAST REF LT RUN DATE -1 */
/*
```

In Example 2, only data sets that were last referenced more than one day before the run date are included in the DEFRAG operation. That is, those that were last referenced one day before or on the run date are excluded.

Results of a successful DEFRAG operation

Figure 19 on page 385 is the printout from a DEFRAG run for a DASD volume. It gives an indication of the free space fragmentation before and after a DEFRAG operation, as well as the distribution of data set extents by size.

The following JCL was used for this job:

```
//STEPT010  EXEC     PGM=ADRDSSU,PARM='RACFLOG=YES,TRACE=YES'
//SYSPRINT  DD       SYSOUT=*
//SYSIN     DD       *
DEFRAG DYNAM(D9S060)
/*
```

```

PAGE 0001      5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES      1999.211 14:55
  DEFrag
  DYNAM(D9S060)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'DEFrag '
ADR109I (R/I)-RI01 (01), 1999.211 14:55:33 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 14:55:33 EXECUTION BEGINS
ADR208I (001)-EANAL(01), 1999.211 14:55:34 BEGINNING STATISTICS ON D9S060:
                           FREE CYLINDERS                000058
                           FREE TRACKS                   000003
                           FREE EXTENTS                  000002
                           LARGEST FREE EXTENT (CYL,TRK) 000053,0003
                           FRAGMENTATION INDEX          0.050
                           PERCENT FREE SPACE           97
ADR220I (001)-EANAL(01), INTERVAL BEGINS AT CC:HH 00001:0000 AND ENDS AT
                           CC:HH 00006:000C
ADR209I (001)-EFrag(01), 1999.211 14:55:34 MOVED EXTENT 001 FROM
                           00006:0000-00006:0004 TO 00001:0000-00001:0004
                           FOR PUBSEXMP.ESDS.S01.DATA
ADR209I (001)-EFrag(01), 1999.211 14:55:34 MOVED EXTENT 001 FROM
                           00006:0005-00006:0006 TO 00001:0005-00001:0006
                           FOR PUBSEXMP.KSDS.S01.DATA
ADR209I (001)-EFrag(01), 1999.211 14:55:35 MOVED EXTENT 001 FROM
                           00006:0007-00006:0008 TO 00001:0007-00001:0008
                           FOR PUBSEXMP.SAM.S01
ADR209I (001)-EFrag(01), 1999.211 14:55:35 MOVED EXTENT 001 FROM
                           00006:0009-00006:000A TO 00001:0009-00001:000A
                           FOR PUBSEXMP.PDS.S01
ADR209I (001)-EFrag(01), 1999.211 14:55:35 MOVED EXTENT 002 FROM
                           00006:000B-00006:000B TO 00001:000B-00001:000B
                           FOR PUBSEXMP.PDS.S01
ADR213I (001)-EANAL(01), 1999.211 14:55:35 ENDING STATISTICS ON D9S060:
                           DATA SET EXTENTS RELOCATED    000005
                           TRACKS RELOCATED               000012
                           FREE CYLINDERS                000058
                           FREE TRACKS                   000003
                           FREE EXTENTS                  000001
                           LARGEST FREE EXTENT (CYL,TRK) 000058,0003
                           FRAGMENTATION INDEX           0.000

PAGE 0002      5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES      1999.211 14:55
ADR212I (001)-EANAL(01), EXTENT DISTRIBUTION MAP FOR D9S060:
      EXTENT *FREE SPACE BEFORE* *FREE SPACE AFTER* *ALLOCATED *
      SIZE
      IN      NO.      CUM.      NO.      CUM.      NO.      CUM.
      TRACKS  EXTS  PCT/100  EXTS  PCT/100  EXTS  PCT/100
      1              4      0.148
      2              4      0.444
      5              1      0.629
      10             1      1.000
      75             1      0.085
      >499           1      1.000      1      1.000
ADR006I (001)-STEND(02), 1999.211 14:55:35 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 1999.211 14:55:35 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 1999.211 14:55:35 DFSMSDSS PROCESSING COMPLETE. HIGHEST
      RETURN CODE IS 0000

```

Figure 19. Printed Output Resulting from a Successful DEFrag Run on nonEAV:

Note: The following section of the printed output resulting from a successful DEFrag run changes for EAV to include two columns: first column refers to the entire volume, the second column refers to the cylinder-managed space, as follows:

```

ADR213I (001)-DFANL(01), 2007.226 12:50:17 ENDING STATISTICS ON C9NS05:
      *STATISTICS*      *VOLUME*      *TRKAREA*
      DATA SET EXTENTS RELOCATED    00000037    00000033
      TRACKS RELOCATED              00003212    00001007
      FREE CYLINDERS                00086950    00065236
      FREE TRACKS                   00000022    00000022
      FREE EXTENTS                  00000005    00000004
      LARGEST FREE EXTENT (CYL,TRK) 00063426.03 00063426.03
      FRAGMENTATION INDEX           0.050      0.010

```

Figure 20. A Section of the Printed Output Resulting from a Successful DEFrag Run on EAV:

The value in the first column of message ADR212I is the size of the extent in tracks (free space or data set) and is printed only if an extent of that size occurs. The second and third columns show the number of free space extents existing *before* processing that were of the size shown in the first column, along with their cumulative percentage divided by 100. The fourth and fifth columns give the same information for free space, but *after* processing. The sixth and seventh columns give the distribution of allocated data set extents, which do not change during a run.

DUMP command for DFSMSdss

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), or as objects in an object storage cloud. The storage medium for the sequential data set can be tape or DASD. When the output resides on DASD, it may be a basic, large or extended format sequential data set. When the output resides in cloud storage, data is stored as a set of objects.

Restriction: When using the DUMP command to create a backup in an object storage cloud, DFSMSdss only supports logical and full volume processing.

The FULL keyword for the DUMP command specifies that an entire DASD volume is to be dumped. The TRACKS keyword with the DUMP command specifies ranges of tracks to be dumped.

SHARE

Do not use the SHARE keyword during a physical dump of HFS or zFS data sets.

For an HFS data set, DFSMSdss obtains both an ADRDSN enqueue and a SYSZDSN enqueue. SHARE determines only whether the ADRDSN enqueue is shared or exclusive.

For a zFS data set, DFSMSdss obtains an ADRDSN enqueue, a SYSDSN enqueue, and a number of SYSVSAM enqueues. SHARE determines only whether the ADRDSN enqueue is shared or exclusive. When specifying DELETE, DFSMSdss attempts to obtain exclusive SYSDSN and SYSVSAM enqueues. If you do not specify DELETE, DFSMSdss attempts to obtain shared SYSDSN and SYSVSAM enqueues.

TOLERATE

ENQFailure specifies that data sets are to be processed even though shared or exclusive access fails. TOL(ENQF) and FULL or TRACKS are mutually exclusive; you cannot specify these keywords together.

Unlike PDS data sets, PDSE data sets that are open for update cannot be dumped even if TOL(ENQF) is specified.

If you must dump a PDSE data set and it must be open for update, process the data set with concurrent copy (specify CONCURRENT). If you cannot use concurrent copy, convert the PDSE back to PDS and then dump the PDS data set with TOL(ENQF).

TOL(ENQF) is not honored when processing a logical dump of HFS or zFS data sets.

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

DFSMSdss offers several ways to process DUMP commands:

- *Logical processing* is data set-oriented, which means it operates against data sets independently of physical device format.
- *Physical processing* can operate against data sets, volumes, and tracks, but is oriented toward moving data at the track-image level.
- *File processing* is z/OS UNIX-oriented and can process individual UNIX files that are specified as absolute paths.

The processing method is determined by the keywords specified on the command.

DFSMSdss logical dump processing cannot be used to process partitioned data sets containing location-dependent information that does not reside in note lists or in the directory. Furthermore, DFSMSdss cannot be used to dump migrated data sets.

DFSMSdss file processing can only be performed on files that are known to z/OS UNIX and that reside within a zFS physical file system.

Integrated catalog facility catalogs should not have a high-level qualifier of SYSCTLG because this causes DFSMSdds to treat them as a control volume (CVOL).

For more information about options and keywords

Special considerations for dump

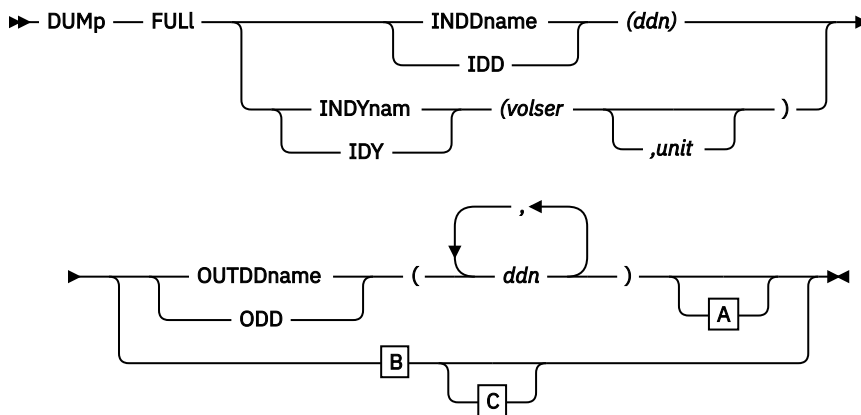
The following special considerations apply when you are performing a dump operation:

- A logical data set dump cannot be performed on the following data sets:
 - 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
- A physical data set dump cannot be performed on the following data sets:
 - KSDSs with key ranges. Use logical processing for this type of data set.
 - VSAM data sets not cataloged in an integrated catalog facility catalog.
 - Page, swap, and SYS1.STGINDEX data sets.
- A File dump cannot be performed on the following file types:
 - Character special files
 - Sockets

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

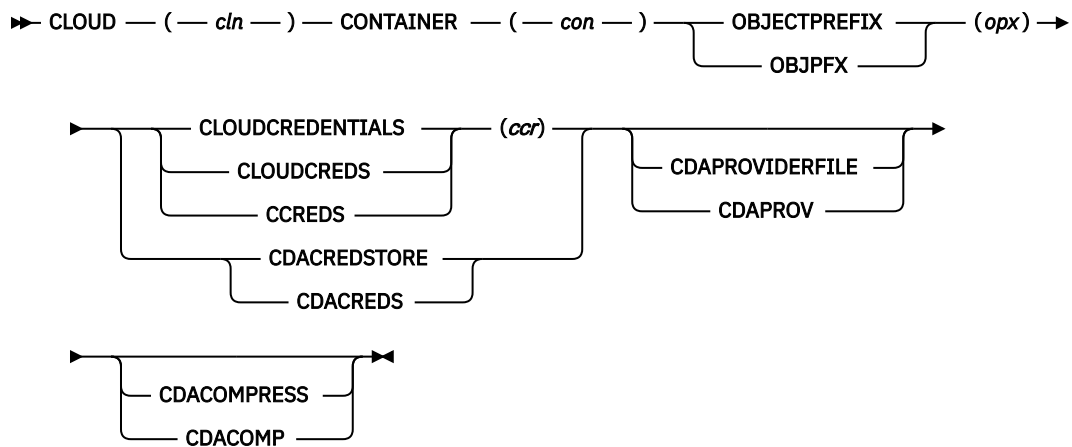
For more information about dumping data sets, see *z/OS DFSMSdss Storage Administration*.

DUMP FULL command syntax

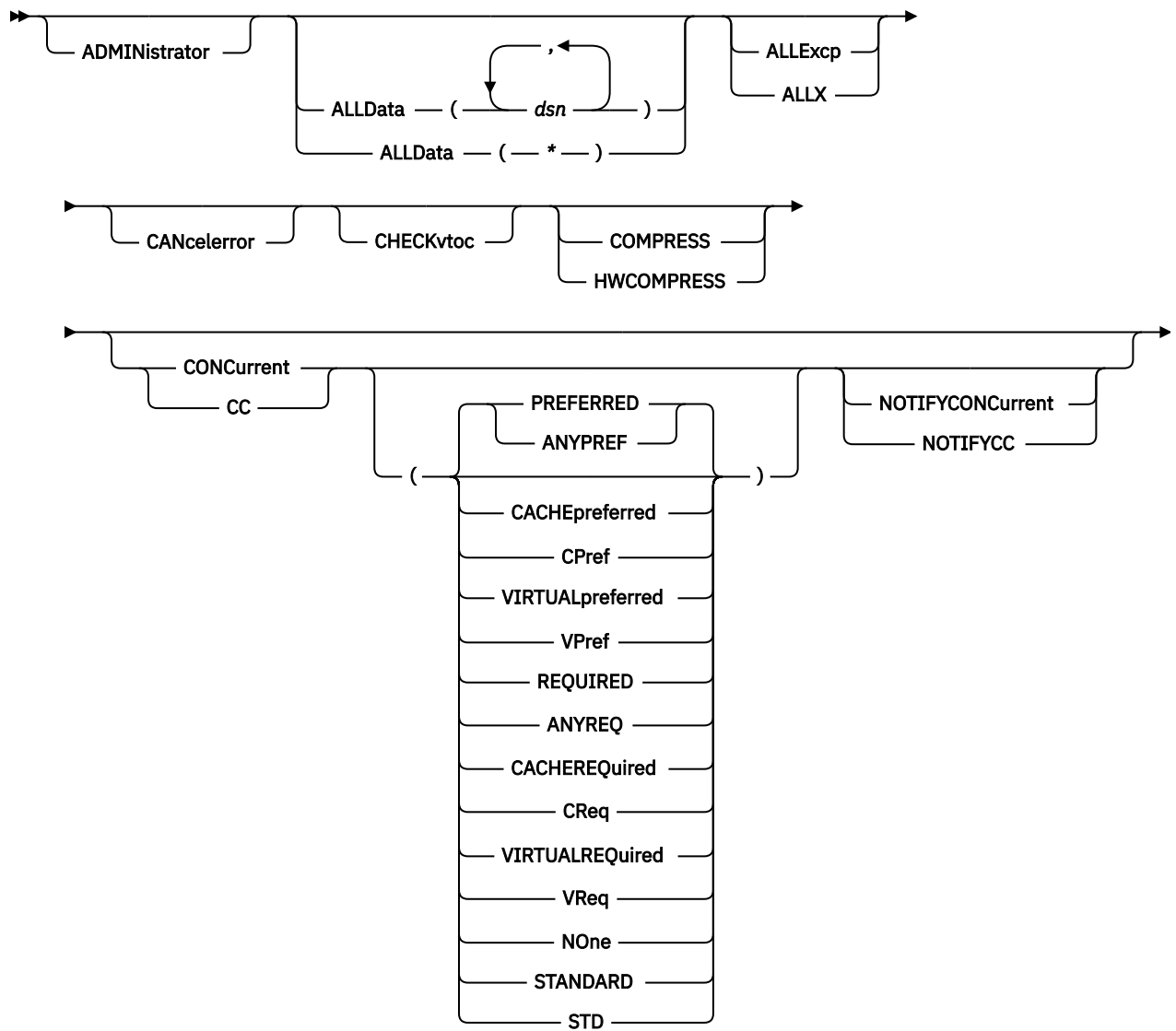


B: Additional keywords for DUMP FULL

DUMP Command

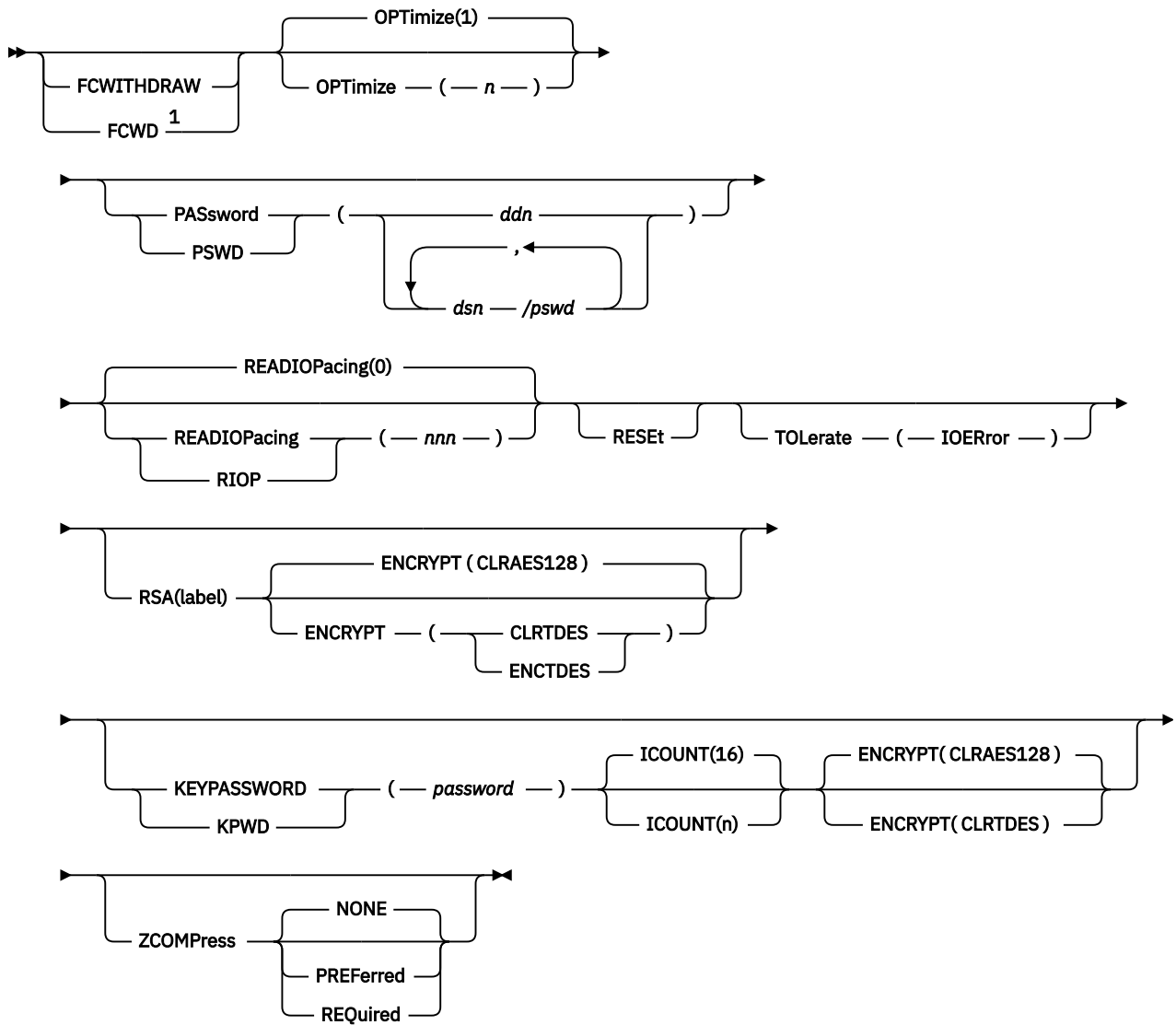


A: Optional keywords for DUMP FULL

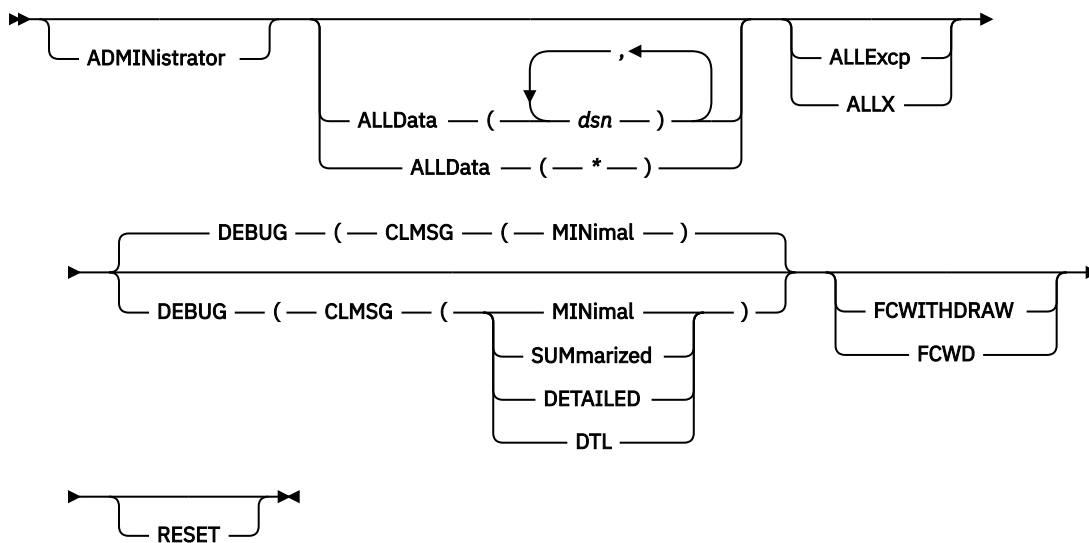


Fragment 1-A

Fragment 1-A



C: Optional keywords with DUMP FULL and CLOUD specified



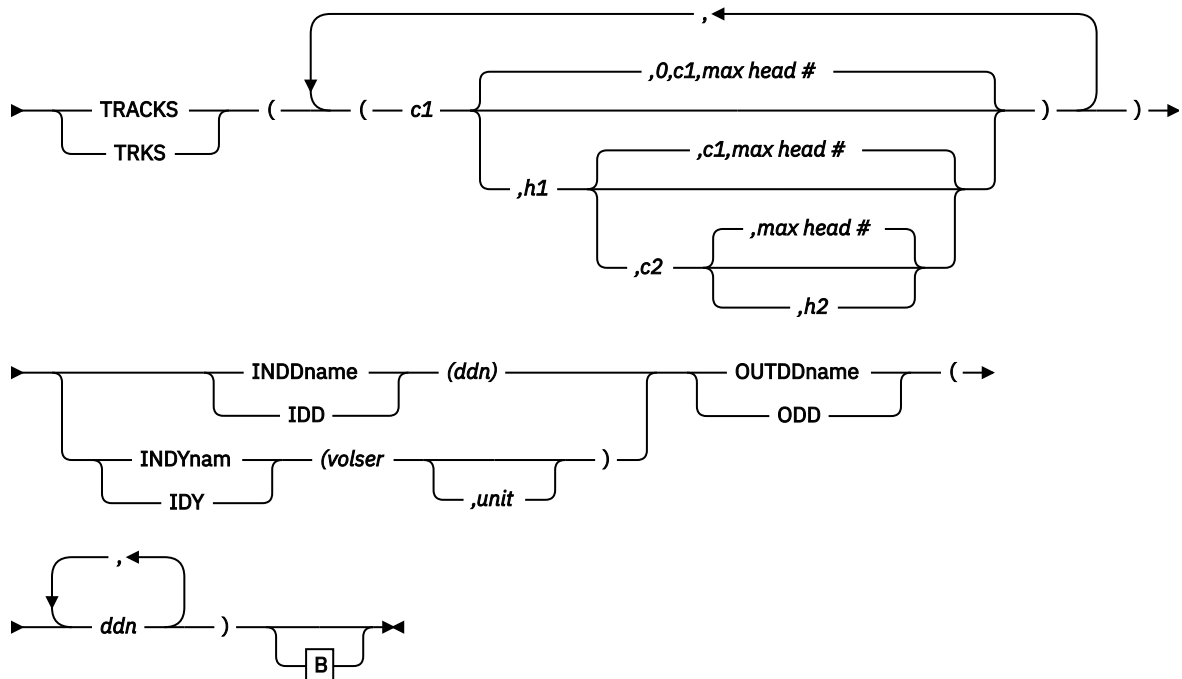
Notes:

DUMP Command

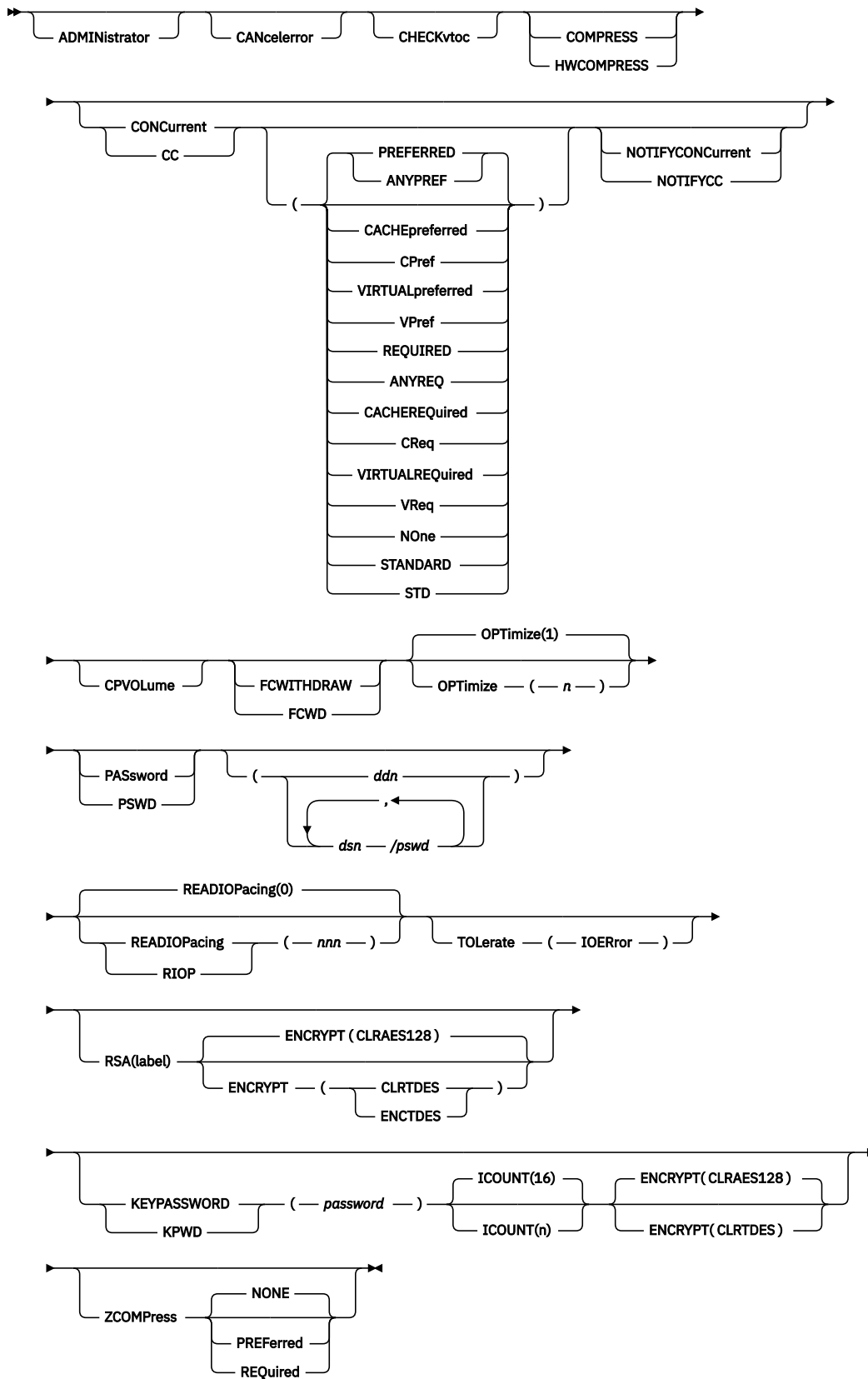
¹ You cannot specify the FCWITHDRAW keyword with the CONCURRENT(ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ) or RESET keywords at the same time.

DUMP TRACKS command syntax

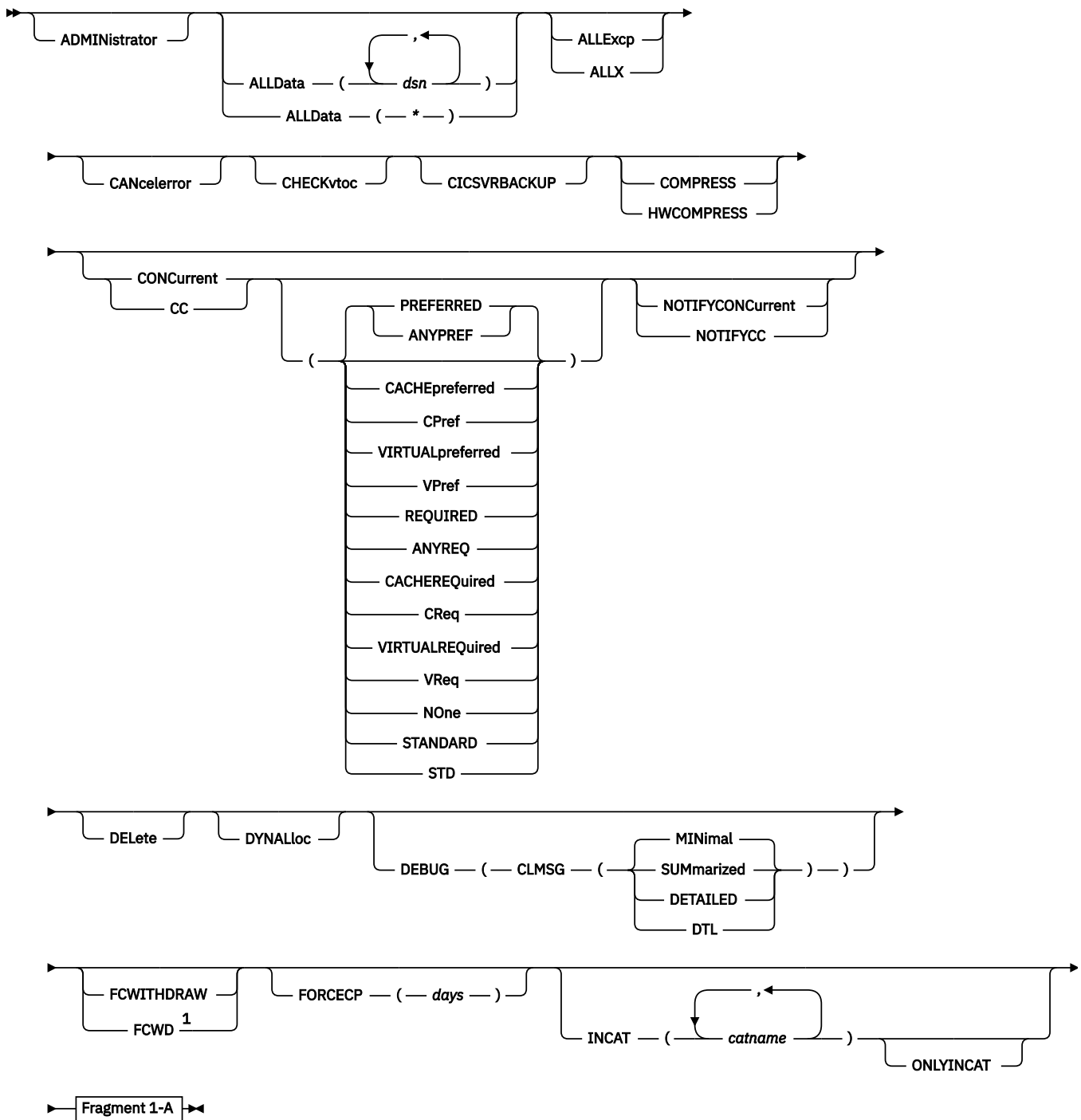
►► DUMP ►►



B: Optional keywords for DUMP TRACKS

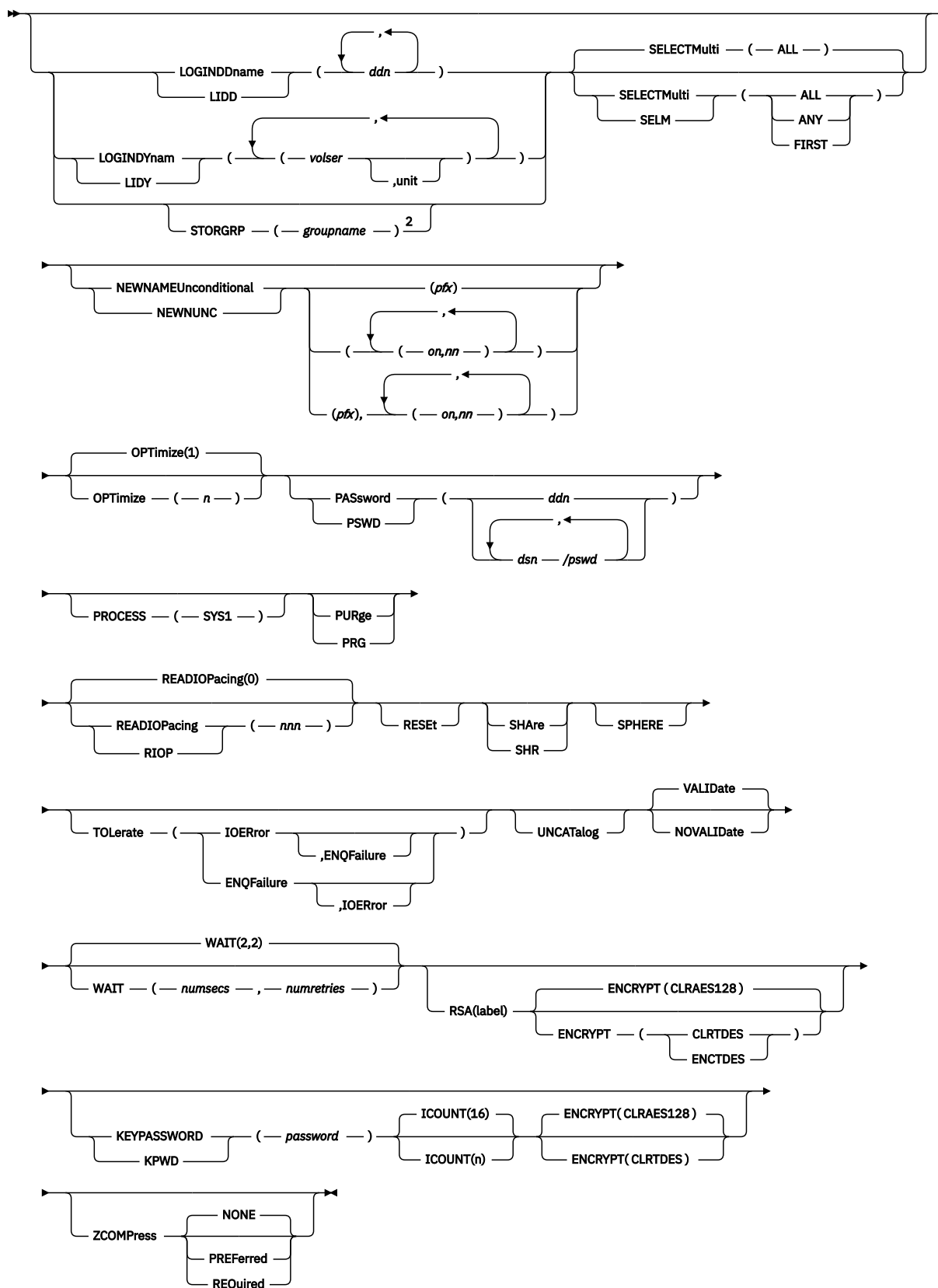






Fragment 1-A

DUMP Command

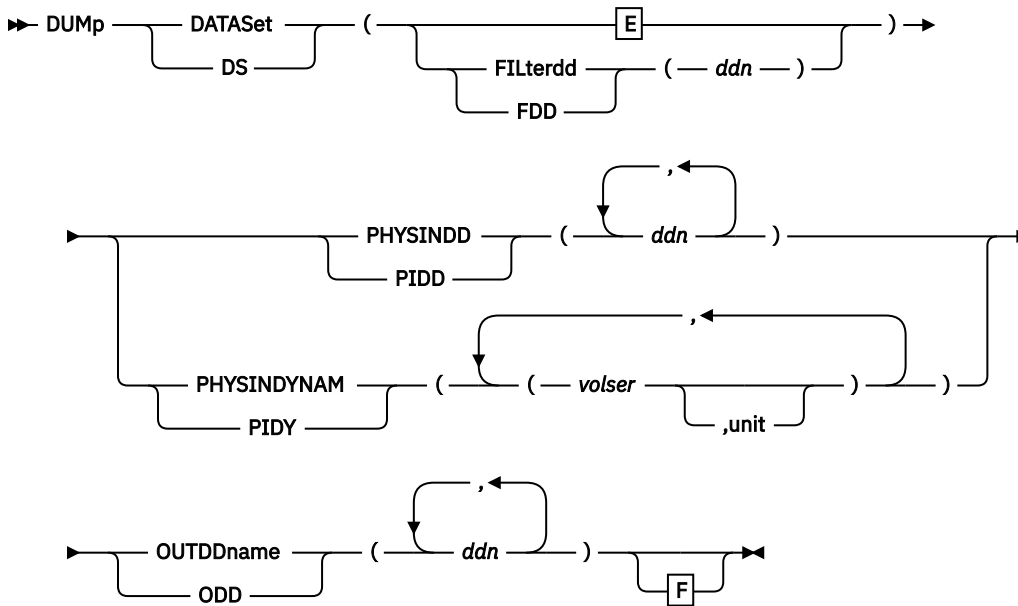


Notes:

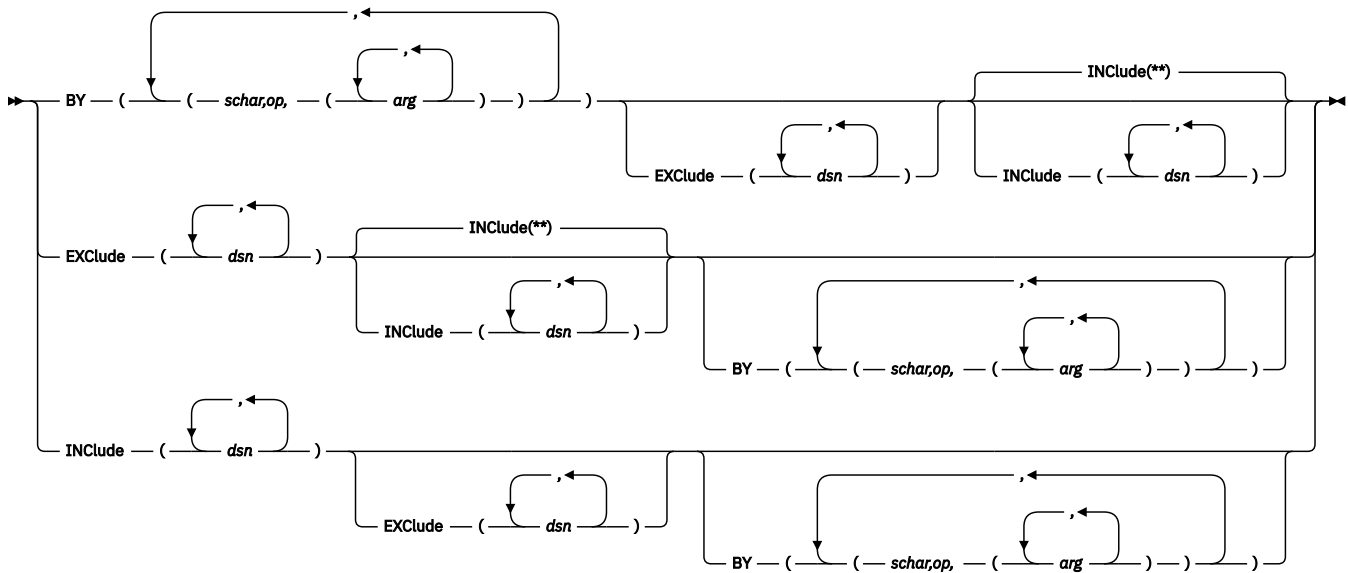
¹ You cannot specify the FCWITHDRAW keyword with the CONCURRENT(ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ) or RESET keywords at the same time.

² You cannot specify the STORGRP keyword with the INDDNAME, INDYNAM, LOGINDDNAME or LOGINDDYNAM keywords at the same time.

DUMP DATASET syntax for physical data set

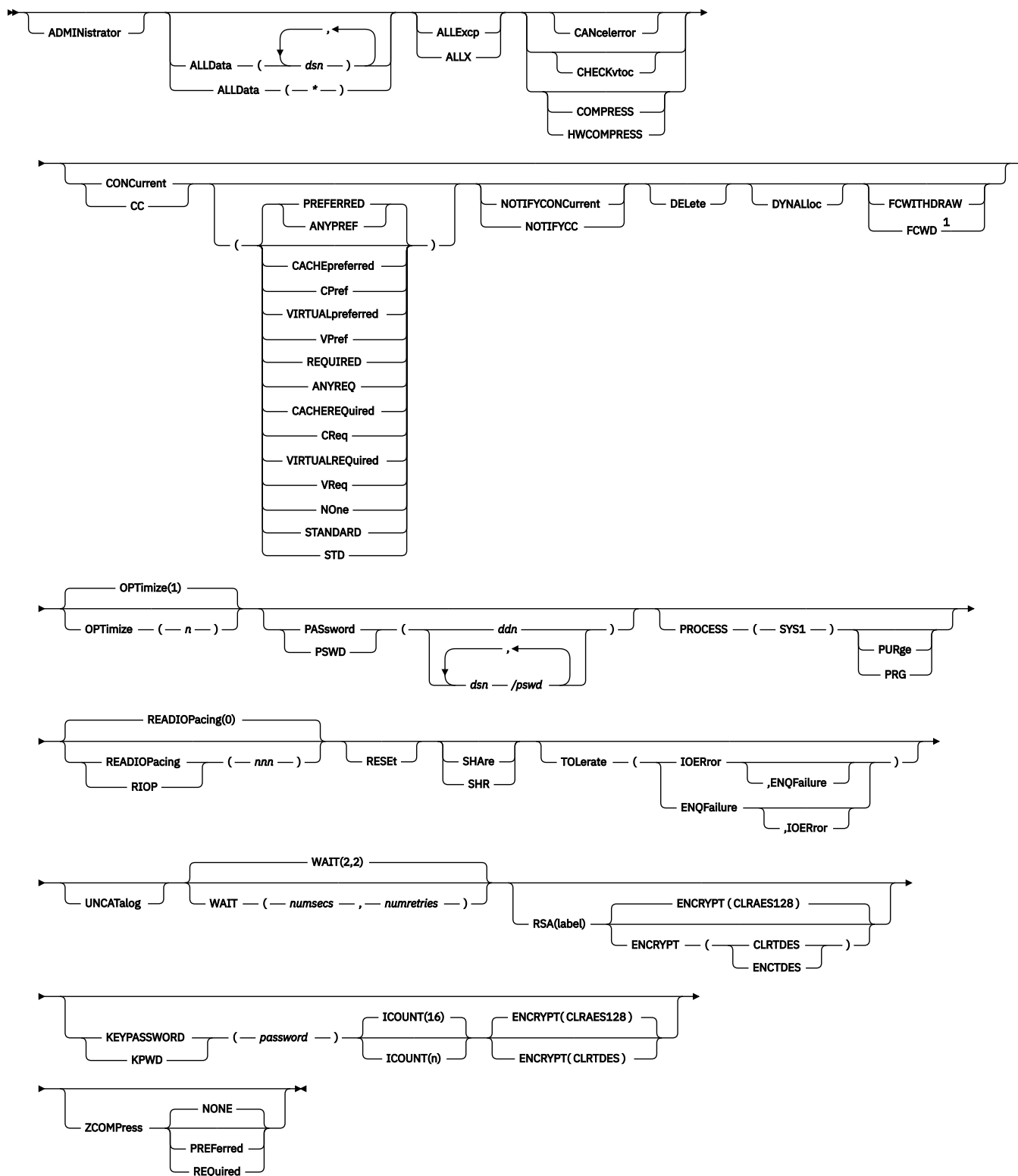


E: Additional Keywords with DUMP DATASET for Physical Data Set



F: Optional Keywords with DUMP DATASET for Physical Data Sets

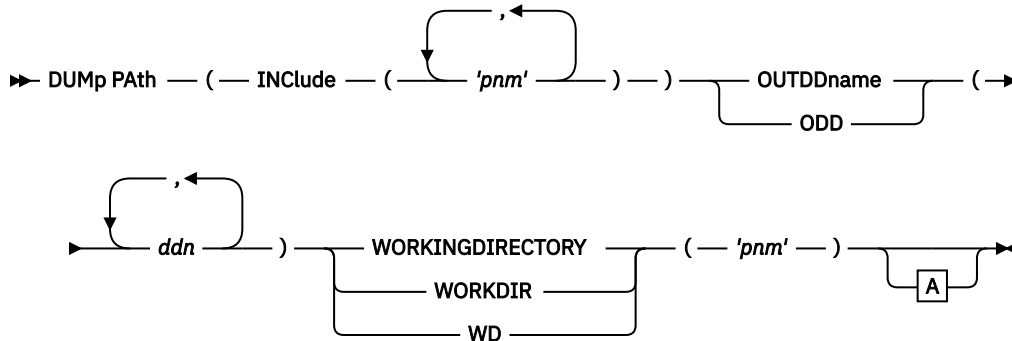
DUMP Command



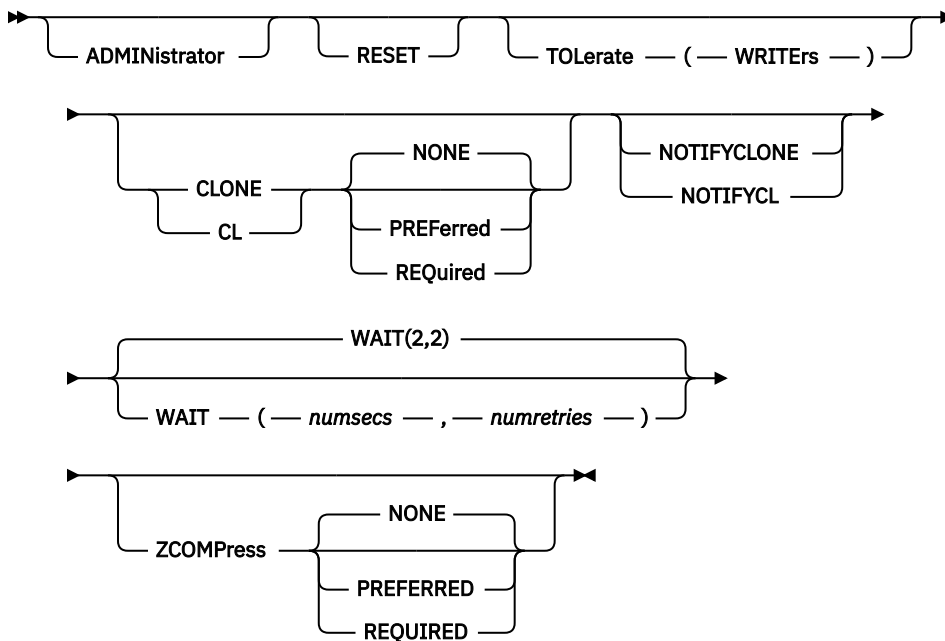
Notes:

- ¹ You cannot specify the FCWITHDRAW keyword with the CONCURRENT(ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ) or RESET keywords at the same time.

DUMP PATH syntax



A: Optional keywords for DUMP PATH:



Explanation of DUMP command keywords

This section describes the keywords for the DUMP command.

ADMINISTRATOR



ADMINISTRATOR lets you act as a DFSMSdss-authorized storage administrator for the DUMP command. If you are not authorized to use the ADMINISTRATOR keyword, the command is ended with an error message. Otherwise, access checking to z/OS UNIX files, data sets and catalogs that are initiated by DFSMSdss are bypassed.

To use the ADMINISTRATOR keyword all of the following must be true:

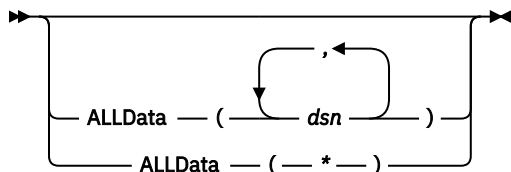
- FACILITY class is active
- Applicable FACILITY-class profile is defined
- You have READ access to that profile.

Note:

- When specified while processing UNIX files (PATH keyword), the ADMINISTRATOR keyword allows you to act as a DFSMSdss-authorized z/OS UNIX administrator.

For more information about how to use the ADMINISTRATOR keyword, see [“ADMINISTRATOR keyword”](#) on page 542.

ALLDATA



ALLDATA applies to full and data set dump operations.

dsn

Specifies the fully qualified name of a data set whose data set organization is PS, PSU, PO, POU, or null, for which all allocated space is to be dumped.

*** (asterisk)**

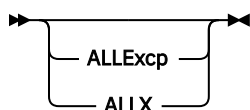
Specifies that all allocated space is to be dumped for the following data sets:

- All sequential and partitioned data sets that are not empty, and
- Data sets whose data set organization is null and are not empty. (The last used block pointer in the data set's volume table of contents (VTOC) entry is not zero.)

Note:

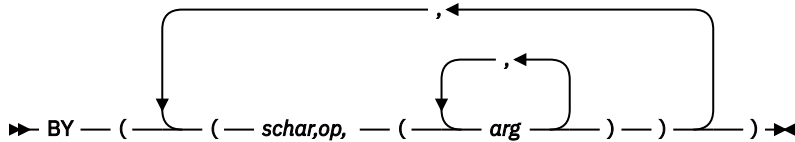
1. Data beyond the last used block pointer is not retained when ALLDATA or ALLEXCP is specified for an extended-format sequential data set during a logical dump operation. During a subsequent restore operation, the target data set is allocated with the same amount of space as the source data set.
2. All of the allocated space is always dumped for a physical dump of PDSE and HFS data sets.
3. For a logical dump of PDSE or HFS data sets, the following conditions are true:
 - If DFSMSdss can determine the amount of used space, DFSMSdss dumps only the used space, regardless of the ALLDATA keyword.
 - If you specify ALLDATA, DFSMSdss remembers the amount of allocated space. During a subsequent restore operation, DFSMSdss will allocate the target data set with the same amount of space as the source data set.
 - If DFSMSdss cannot determine the amount of used space, it assumes that the used space is equal to the allocated space. DFSMSdss then dumps all of the allocated space.

ALLEXCP



ALLEXCP is an option for full and data set dump operations. It instructs DFSMSdss to dump all allocated space for data sets whose data set organization is PS, PSU, PO, POU, or null and are empty (the last used block pointer in the data set's VTOC entry is zero).

Note: Using the ALLEXCP keyword will result in all the allocated space being dumped for applicable data sets. However, whether or not all the allocated space is restored depends on data set characteristics and device characteristics during the restore. If you require that all of the unused space is restored, then you should be sure 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.

BY

BY specifies that the data sets selected up to this point by the processing of the INCLUDE and EXCLUDE keywords are to be further filtered. To select the data set, all BY criteria must be met. See [“Filtering by data set characteristics”](#) on page 258 for a full discussion of *schar*, *op*, and *arg*. See the separate discussions of INCLUDE and EXCLUDE for information on how these keywords are specified.

Note: You must use FILTERDD when you have more than 255 entries in INCLUDE, EXCLUDE, or BY list keywords.

CANCELERROR

CANCELERROR specifies that the dump operation is to be ended if a track-related permanent read error occurs. If this keyword is *not* specified and a permanent read error occurs, the track image record is flagged on the output volume as having an I/O error, and the dump operation continues. This track is not restored in a restore operation. When CANCELERROR is specified, the TOLERATE(IOERROR) keyword is ignored.

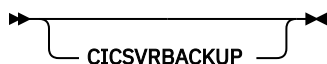
This keyword may be used in conjunction with the CHECKVTOC keyword to specify whether or not an operation is to continue in the event of terminating VTOC errors discovered during VTOC checking. Refer to the CHECKVTOC keyword.

Note: CANCELERROR has no effect on the following types of errors on a DASD volume:

- Equipment check
- Command reject
- Intervention required
- Busout parity

CHECKVTOC

CHECKVTOC specifies that a VTOC analysis of the source volume be performed during dump processing. In the event of terminating VTOC errors found during analysis, operation continues unless the CANCELerror keyword is specified. CHECKVTOC is ignored if CPVOLUME is also specified.

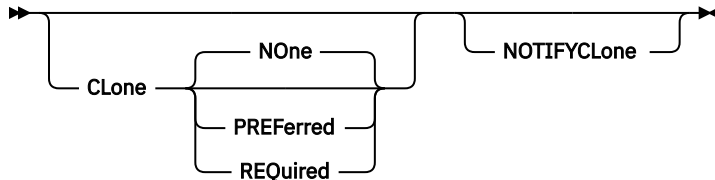
CICSVRBACKUP

CICSVRBACKUP specifies that for a logical data set dump operation, DFSMSdss creates backups for use by CICSVR. DFSMSdss notifies the CICSVR server address space when a CICSVR backup is made for a VSAM base cluster or RLS catalog. This enables CICSVR to manage backups that are made by DFSMSdss.

Note:

1. CICSVRBACKUP is intended to be used with CICSVR. The minimum required CICSVR release is z/OS 3.1. To use CICSVRBACKUP, the CICSVR server address space must be active.
2. CICSVRBACKUP applies to a logical data set dump only.
3. CICSVR manages VSAM base clusters backed up using the DFSMSdss DUMP command. The CICSVRBACKUP keyword is ignored for non-VSAM data sets and VSAM alternate indexes. When you specify CICSVRBACKUP, DFSMSdss DUMP processes AIXs as usual but does not notify CICSVR of backups that are made for AIXs.

CLONE



CLONE specifies that DFSMSdss creates a point-in-time backup of a regular file. This is performed by requesting the file system to create a virtual clone (snapshot) of the file and dumping from the clone instead of the base file. This allows applications to continue using the base file while the file is being backed up.

PREFERRED

Specifies that DFSMSdss process the file records from a clone. If clone processing cannot be completed successfully, then the dump is processed from the base file.

REQUIRED

Specifies that DFSMSdss must process the file records from a clone. If clone processing cannot be completed successfully, then the file is not processed.

NONE

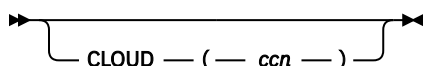
Specifies that DFSMSdss process the file records from the base file. This is the same as not specifying the CLONE keyword.

1. CLONE is only supported for regular files. Only a single clone per file data is allowed.
2. Hard-linked regular files are not supported.
3. CLONE performs I/O in units of 4K block multiples.
4. The RESET keyword is ignored if you specify CLONE(PREFERRED | REQUIRED) unless you specify a patch to allow the RESET function.
5. CLONE processing must be requested to ensure sparse files remain sparse in the dump data set.

NOTIFYCLONE can be specified in conjunction with CLONE(PREFERRED | REQUIRED).

NOTIFYCLONE specifies that DFSMSdss issues an information message for every file that is successfully cloned. If you do not specify NOTIFYCLONE, DFSMSdss issues messages only for files that are not successfully cloned.

CLOUD



ccn

Specifies the name of either a CDA provider file or an SMS construct that identifies the cloud storage the dump to be written on.

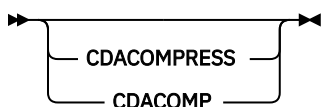
If the keyword **CDAPROVIDERFILE** is present, DFSMSdss will look for a z/OS Cloud Data Access (CDA) provider file with the name *ccn.json*. The provider file will not be used if it does not contain the key-value pair "enableDFSMSdss=YES".

If the keyword **CDAPROVIDERFILE** is absent, DFSMSdss will look for a network connection construct by the name *ccn* in the active SMS configuration.

Note:

1. Do not specify CLOUD during DUMP with OUTDD, CONCURRENT, COMPRESS, HWCOMPRESS, ZCOMPRESS, or RSA.
2. If the cloud solution being used is TCT, the following keywords will be ignored if CLOUD is specified: VALIDATE and OPTIMIZE.
3. You must specify the CONTAINER and OBJECTPREFIX keywords when specifying the CLOUD keyword.
4. The name can be up to 30 characters in length.
5. To specify CLOUD, RACF authorization may be required.
6. DFSMSdss will look for a z/OS Cloud Data Access (CDA) provider file with the specified CLOUD name if CDAPROVIDERFILE keyword is specified.

CDACOMPRESS



When specified, DSS will request CDA to use zEDC services to compress user data prior to storing it in the cloud. During RESTORE, DSS will automatically request decompression if an object had been compressed.

Note:

1. CDACOMPRESS is only supported with the Direct to Cloud solution; it is otherwise ignored.
2. Compression is not performed when the data is already compressible and/or encrypted.

CDACREDSTORE

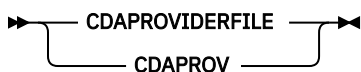


Specifies that the cloud credentials have been stored by using the z/OS Cloud Data Access (CDA) Authorization Facility. When specified, DFSMSdss will request the cloud provider credentials from CDA using the cloud name specified in the CLOUD keyword.

Note:

1. Do not specify CLOUDCREDENTIALS if you are specifying CDACREDSDSTORE.

CDAPROVIDERFILE

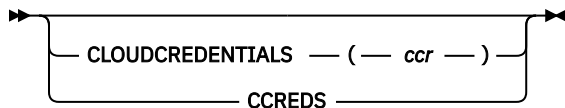


Specifies that DFSMSdss should look for a z/OS Cloud Data Access (CDA) provider file with the specified CLOUD name. If a valid CDA provider file does not exist or is empty, DFSMSdss will search for a network connection construct with the same name in the active SMS configuration.

Note:

- This keyword may be set to a default specification using ADRPATCH offset X'62'. For more information on DFSMSdss patch bytes, see [Chapter 14, “DFSMSdss patch area,” on page 213](#)
- For more information about the CDA Provider File, see [“Provider file” on page 12.](#)

CLOUDCREDENTIALS



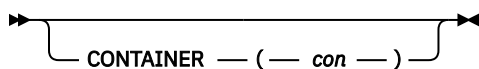
ccr

Specifies an up to 64 character credential (in EBCDIC) that is used when authenticating with a cloud. Valid characters are uppercase and lowercase letters A through Z, numerals 0-9, and the following characters: @#\$%&*~_=:<>?|{. You cannot use embedded spaces, commas (,), forward slash (/), parentheses (()), or semi-colons. DFSMSdss removes leading and trailing blanks.

Note:

1. Do not specify CDACREDSTORE when you specify CLOUDCREDENTIALS.
2. The credentials that are specified in your input command stream are not printed to the SYSPRINT output.
3. The credentials must be kept secure. Batch jobs that specify this keyword must be in a data set or library that has limited access and is controlled by a security product. Do not use this keyword if the credentials cannot be kept secure.

CONTAINER



con

Specifies the container in which objects are stored. The container specified as input does not have to exist at the time of the backup. If necessary, DFSMSdss will create it for you.

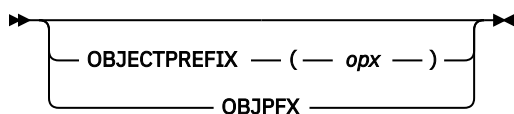
A DFSMSdss specific prefix may be prepended to the specified container name, according to the cloud storage solution being used:

- When using a TCT cloud storage solution, when no prefix is specified the prefix prepended will be 'SYSZADR.'
- When using the Direct to Cloud storage solution, the prefix prepended will be 'SYSZADR-'
- If either 'SYSZADR.' or 'SYSZADR-' is specified as part of the container name, no prefix will be prepended regardless of the cloud storage solution used.

Note:

1. The name can be up to 128 characters in length. If the CDACREDSTORE keyword is used, the container name may be folded to lowercase.
2. The allowable characters are uppercase letters A-Z, numbers 0-9, special characters \$ @ # - _ , and . in the nonfirst position.
3. Specify the CLOUD and OBJECTPREFIX keywords when you specify the CONTAINER keyword.

OBJECTPREFIX



opx

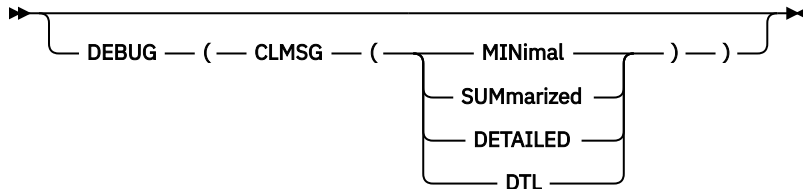
Specifies a unique prefix that DFSMSdss is to use for all of the objects that make up a particular backup. The prefix provides uniqueness amongst multiple backups in the same container. At the beginning of the backup process, DFSMSdss will determine if a backup using the same *opx* exists in

the specified cloud and container. If a backup already exists with the same *opx*, then DFSMSdss will fail the backup. To overwrite a backup using the same *opx*, you can set a patch at offset x'5D'.

Note:

1. The name can be up to 44 characters in length.
2. You must specify to the CLOUD and CONTAINER keywords when specifying the OBJECTPREFIX keyword.

DEBUG



You can use DEBUG as a diagnostic tool. When you specify the CLMSG subkeyword, DFSMSdss issues messages that provide details on the progress of a backup to an object storage cloud. When DEBUG(CLMSG) is not specified, MINimal is the default. Specify DEBUG(CLMSG) with one of the following sub-keywords:

CLMSG(MINimal)

Specifies that DFSMSdss is not to issue any messages that provide detail on the progress of a backup to an object storage cloud.

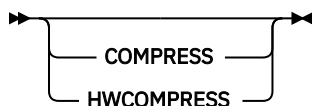
CLMSG(SUMmarized)

Specifies that DFSMSdss is to issue an informational message for each object that is stored in an object storage cloud.

CLMSG(DETAILED)

Specifies that DFSMSdss is to provide detailed information about each HTTP request that is made to an object storage cloud. If the CDAPROVIDERFILE keyword is used, debug information from Cloud Data Access (CDA) will be printed.

COMPRESS



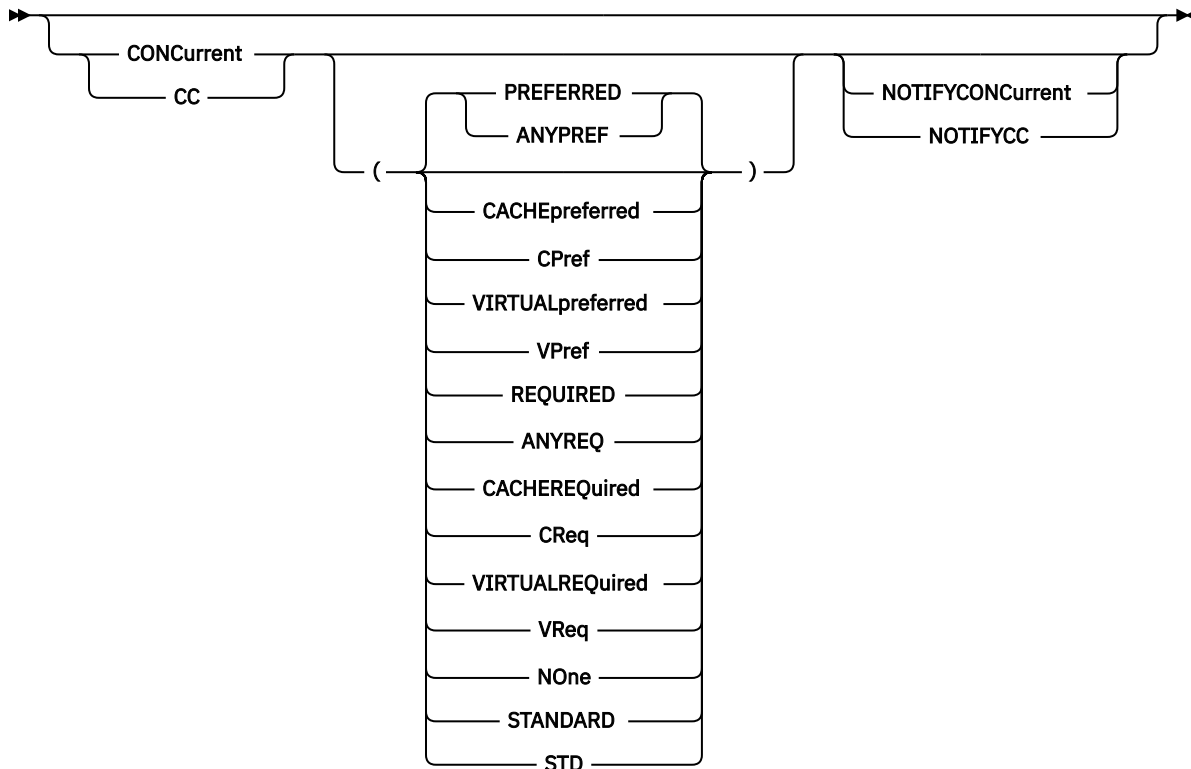
COMPRESS specifies that the dumped data is to be written in compressed form to the output medium. This decreases the space occupied by the dump data at the expense of increased processor and elapsed times.

Note:

1. The COMPRESS keyword is ignored if it is specified during a logical data set dump for either compressed-format sequential data sets or compressed-format VSAM data sets.
2. If you have a tape drive with the compaction feature and you want to use hardware data compaction, you do not need to specify the COMPRESS keyword. If software data compression is desired, you do not need to specify DCB=TRTCH=COMP in the JCL. However, you may specify both the COMPRESS keyword and DCB=TRTCH=COMP.
3. If you want to use compressed format sequential data sets on DASD, you do not need to specify the COMPRESS keyword. To obtain software data compression, you do not need to use a compressed format sequential data set on DASD. However, you may specify both the COMPRESS keyword and a data class on the output DD that requests compressed format sequential data sets.

4. To improve performance and save dump space, specify the OPTIMIZE keyword with the COMPRESS keyword.
5. When encrypted data sets are dumped by data set, or reside within a volume that is dumped, the compression ratio can show a decreased ratio.

CONCURRENT



The **CONCURRENT** keyword specifies that the data is to be processed with concurrent copy. You can specify one of the following optional sub-keywords to indicate the type of concurrent copy to be used and whether DFSMSdss can use other methods of data movement when concurrent copy cannot be used or fails:

ANYPREFERRED or PREFERRED

Specifies that data is to be processed with concurrent copy. Virtual concurrent copy is attempted first, if the storage subsystem on which the data resides is capable of it and working-space data sets have been defined. Otherwise, cache-based concurrent copy is attempted if the storage subsystem is capable of it. If neither type of concurrent copy is possible or both fail, the data is processed with standard I/O. **PREFERRED** is the default if you specify the **CONCURRENT** keyword without a sub-keyword.

ANYREQUIRED or REQUIRED

Specifies that data is to be processed with concurrent copy. Virtual concurrent copy is attempted first, if the storage subsystem on which the data resides is capable of it and working-space data sets have been defined. Otherwise, cache-based concurrent copy is attempted, if the storage subsystem is capable of it. If neither type of concurrent copy is possible or both fail, the data is not processed.

CACHEPREFERRED

Specifies that data is to be processed with cache-based concurrent copy. If cache-based concurrent copy cannot be used or fails, the data is processed with standard I/O. DFSMSdss does not attempt to use virtual concurrent copy.

CACHEREQUIRED

Specifies that data is to be processed with cache-based concurrent copy. If cache-based concurrent copy cannot be used or fails, the data is not processed. DFSMSdss does not attempt to use virtual concurrent copy or standard I/O.

STANDARD or NONE

Specifies that data is to be processed with standard I/O as if the CONCURRENT keyword was not specified.

VIRTUALPREFERRED

Specifies that data is to be processed with virtual concurrent copy. If virtual concurrent copy cannot be used or fails, the data is processed with standard I/O. DFSMSdss does not attempt to use cache-based concurrent copy.

VIRTUALREQUIRED

Specifies that data is to be processed with virtual concurrent copy. If virtual concurrent copy cannot be used or fails, the data is not processed. DFSMSdss does not attempt to use cache-based concurrent copy or standard I/O.

For a logical data set dump operation, you can also specify the NOTIFYCONCURRENT keyword, as follows:

NOTIFYCONCURRENT

Specifies that DFSMSdss is to issue an informational message for each data set that is successfully included in the concurrent copy operation. If you do not specify NOTIFYCONCURRENT, DFSMSdss issues messages only for data sets that are not included in the concurrent copy operation.

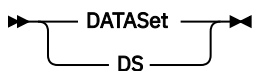
Note:

1. Do not specify NOTIFYCONCURRENT with CONCURRENT(STANDARD | NONE).
2. You cannot use CONCURRENT(ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ) with the DELETE, UNCATALOG, or FCWITHDRAW keywords.
3. The RESET keyword is ignored if you specify CONCURRENT(ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ), unless you apply a patch to allow the reset function.
4. The use of concurrent copy and virtual concurrent copy with the DFSMSdss DUMP command is controlled by the RACF FACILITY class profile, STGADMIN.ADR.DUMP.CNCURRNT.

For information about determining concurrent copy storage requirements and about virtual concurrent copy, see [“Performance considerations” on page 59](#). For more information about working space data sets, see [z/OS DFSMS Advanced Copy Services](#).

CPVOLUME

CPVOLUME specifies that the input volume is a VM-format volume and that the OS-compatible VTOC must begin on track zero, record five. You must specify the track range to be copied with the TRACKS keyword, as the OS-compatible VTOCs do not describe the extents of any data on the volume. You must also specify the ADMINISTRATOR keyword with CPVOLUME because DFSMSdss cannot check access authorization for VM data.

DATASET

DATASET specifies a data set dump operation, using filtering. See Chapter 16, “DFSMSdss filtering—choosing the data sets you want processed,” on page 255 for an explanation of the filtering process used. Unless the ALLDATA or ALLEXCP keyword is specified, only *used tracks* are dumped for sequential and

partitioned data sets and for data sets with no defined data set organization (for example, JES2/JES3 data sets). If the free space map in the VTOC is invalid, all tracks are dumped.

Note: Either the FILTERDD, INCLUDE, EXCLUDE, or BY keyword must be specified when DATASET is selected.

DELETE



For a *physical data set dump*, DELETE instructs DFSMSdss to delete expired single-volume, non-VSAM data sets that are successfully serialized and dumped. In addition, DFSMSdss is to uncatalog successfully deleted data sets. DELETE is ignored for VSAM data sets.

For a *logical data set dump*, DELETE can be used to delete expired single- and multivolume VSAM and non-VSAM data sets that are successfully serialized and dumped. Unmovable data sets can also be deleted. User catalogs cannot be deleted. *Unexpired* source data sets are deleted only if you also specify PURGE.

Note:

1. For both a *physical* and a *logical* data set dump operation, you cannot delete data sets with a high-level qualifier of SYS1 unless you specify PROCESS(SYS1). Even if PROCESS(SYS1) is specified, SYS1.VVDS and SYS1.VTOCIX data sets cannot be dumped and deleted. To delete or scratch a password-protected data set, the operator must supply the password for DADSM scratch password checking.
2. Do not specify SHARE if you specify DELETE.
3. Do not specify DELETE with CONCURRENT, because after the concurrent copy operation has begun, the original data can still be updated.
4. Do not specify DELETE for a mounted HFS data set. DFSMSdss can delete an HFS data set only if DFSMSdss can enqueue the data set exclusively. This is only possible if the data set is unmounted.
5. If the data set being processed is a generation data set (GDS), an exclusive enqueue on the GDG BASE is required in addition to the exclusive enqueue on the GDS.

DYNALLOC



DYNALLOC specifies dynamic allocation, instead of enqueue, to serialize the use of data sets. This allows cross-system serialization in a JES3/MVS environment.

Note:

1. The serialization is of value only when the dynamic allocation/JES3 interface is not disabled.
2. Run time increases when you use DYNALLOC to serialize data sets (as opposed to enqueue) because overhead is involved in dynamic allocation and serialization across multiple processors.
3. If a data set passes INCLUDE/EXCLUDE filtering and is migrated before BY filtering and DYNALLOC is used, the dynamic allocation causes the data set to be recalled. DFSMSdss waits for the recall processing to complete. If the data set is recalled to a different volume, a message is issued indicating that the VTOC entry was not found.
4. For an HFS data set, DFSMSdss ignores DYNALLOC and attempts to get a SYSZDSN enqueue. If DFSMSdss cannot enqueue the HFS data set, DFSMSdss attempts to quiesce the data set.

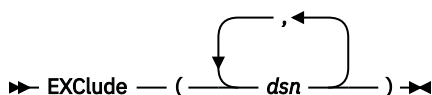
ENCRYPT

ENCRYPT is a subparameter of the RSA and KEYPASSWORD keywords.

Restriction: You cannot use the stand-alone restore program with a dump that was created with the ENCRYPT keyword.

For more information, see [“KEYPASSWORD” on page 411](#) and [“RSA” on page 419](#).

EXCLUDE



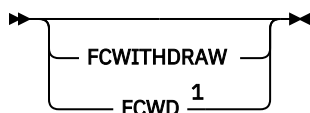
dsn

Specifies the name of a data set to be excluded from the data sets selected by INCLUDE. Either a fully or a partially qualified data set name can be used.

Note: You must use FILTERDD when you have more than 255 entries in INCLUDE, EXCLUDE, or BY list keywords.

For more information, see [“BY” on page 399](#) and [“INCLUDE” on page 409](#).

FCWITHDRAW



Notes:

- ¹ You cannot specify the FCWITHDRAW keyword with the CONCURRENT or RESET keywords at the same time.

FCWITHDRAW specifies that if the volume that is dumped is the target volume of a FlashCopy relationship, the relationship is withdrawn when the dump has successfully completed. Withdrawing the FlashCopy relationship releases the subsystem resources that maintain the FlashCopy relationship.

During DUMP FULL and DUMP TRACKS operations, DFSMSdss might invoke ICKDSF INITIALIZE to ensure that the source volume of the DUMP operation remains online to the system at the end of dump processing. If the VTOC tracks of a source volume that are part of the withdrawn FlashCopy relationship do not look like VTOC tracks, programs might not be able to process the volume. DFSMSdss invokes ICKDSF INITIALIZE only when all of the following conditions are true:

- FCWITHDRAW is specified
- The VTOC tracks on the source volume of the DUMP operation are the target of a FlashCopy relationship or the volume is dump conditioned
- If TRACKS is specified, it designates one extent range that represents the entire volume
- The volume is not a VM-format volume (CP volume)
- The volume supports FlashCopy Version 2.

If the dumped volume is not FlashCopy capable, the FCWITHDRAW keyword is ignored.

Note:

1. When the FlashCopy relationship is withdrawn, the data on the volume that was dumped becomes invalid. Therefore, only use FCWITHDRAW if you no longer need the data on the volume that is dumped, after the dump has completed.
2. If you use the FCNOCOPY keyword, you must withdraw the FlashCopy relationship when the copy is no longer needed to free up the subsystem resources that maintain the FlashCopy relationship. You can withdraw the FlashCopy relationship by performing one of the following tasks:
 - Initiate a dump of the target of the copy and specify the FCWITHDRAW keyword on the DUMP command

- Initiate the TSO FCWITHDR command.
3. The FCWITHDRAW keyword is not supported for devices that are attached at device address X'0000'. If you specify FCWITHDRAW for a volume attached at address X'0000', the system fails the FCWITHDRAW request with a warning message.

For information about the TSO FCWITHDR command, see [z/OS DFSMS Advanced Copy Services](#)

Restrictions: For cases in which the FlashCopy target volume could not be initialized during FCWITHDRAW processing, message ADR288W is issued and FCWITHDRAW processing is not performed. This prevents leaving the volume in an indeterminate state. You can consider adding return code checking in your DUMP FULL/TRACKS job steps to perform an ICKDSF INIT when the return code of the DUMP FULL/TRACKS job steps ends in a return code of 4.

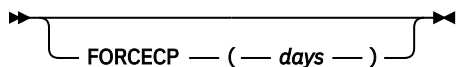
FILTERDD



FILTERDD 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 use. The filtering criteria are in the form of card-image records, in DFSMSdss command syntax, containing the INCLUDE, EXCLUDE, and BY keywords that complete the DUMP command syntax.

Note: You must use FILTERDD when you have more than 255 entries in the INCLUDE, EXCLUDE, or BY list of subkeywords.

FORCECP



FORCECP specifies that checkpoint data sets resident on the SMS volume or volumes can be logically dumped. Checkpoint indications are not removed from the data set during conversion.

days

Specifies a one-to-three digit number in the range of zero to 255. It also specifies the number of days that must have elapsed since the last referenced date before the data set can be dumped.

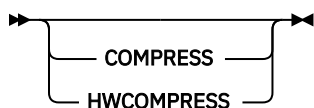
FULL



FULL specifies that an entire DASD volume is to be dumped. This is the default. Unallocated tracks are not dumped. Unless the ALLDATA or ALLEXCP keyword is specified, only *used tracks* are dumped for sequential and partitioned data sets and for data sets with no defined data set organization. (for example, JES2/JES3 data sets). If the free space map in the VTOC is invalid, all tracks are dumped. Used tracks consist of the tracks from the beginning of the data set to the last-used track (as indicated by the last used block pointer in the data set's VTOC entry).

Note: You cannot specify SHARE or TOL(ENQF) for FULL operations.

HWCOMPRESS



HWCOMPRESS specifies that DFSMSdss writes the dumped data in compressed form to the output medium. This decreases the space that is occupied by the dump data. Hardware assisted compression is performed on the dumped data using the IBM Z CMPSC instruction.

Logical Data Set DUMP: Data sets that are less than five tracks in size (used space or allocated space when ALLDATA is specified) are not compressed.

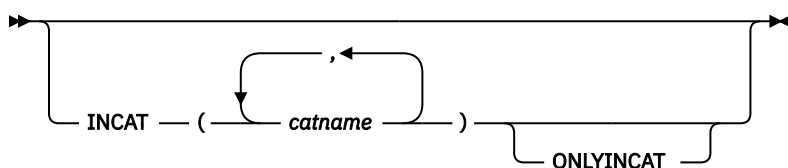
Physical Full Volume, Tracks, and Data Set DUMP: DFSMSdss rebuilds the compression dictionary after 15 consecutive compressions that do not have a new size of at most 94% the original size. The values 15, and 94% may be tuned with patch bytes as described in the DFSMSdss Storage Administration Guide.

Restriction: The HWCOMPRESS keyword cannot be specified with the COMPRESS keyword.

Notes:

1. When encrypted data sets are dumped by data set, or reside within a volume that is dumped, the compression ratio can show a decreased ratio.
2. You cannot use the stand-alone restore program with a dump that has been compressed with the HWCOMPRESS keyword.

INCAT



INCAT(*catname*) specifies that DFSMSdss search the user catalogs specified by the INCAT(*catname*) keyword, then follow the standard search order to locate data sets. INCAT allows you to identify specific source catalogs. To specify INCAT, RACF authorization may be required.

catname

Specifies a fully qualified catalog name.

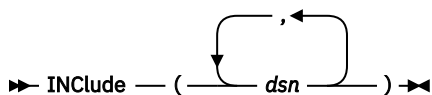
ONLYINCAT

Specifies that DFSMSdss only searches catalogs specified in the INCAT catalog name list.

DFSMSdss does not process an SMS-managed data set that is cataloged outside the standard search order. This is the case even if it is cataloged in one of the catalogs that is specified with the INCAT(*catname*) keyword. Ensure that the SMS-managed data sets are cataloged under standard catalog search order.

For more information about RACF authorization, see [Chapter 5, “Protecting DFSMSdss functions,”](#) on page 35.

INCLUDE



Data set

dsn

Specifies the name of a data set eligible to be dumped. You can use either a fully qualified data set name or a partially qualified data set name. See [“Filtering by data set names”](#) on page 256. If INCLUDE is omitted (but EXCLUDE or BY is specified) or if INCLUDE(**) is specified, all data sets are eligible to be selected for dumping.

Note:

1. You must use FILTERDD when you have more than 255 entries in a list that is created by using the INCLUDE, EXCLUDE, or BY keywords.
2. DFSMSdss does not support INCLUDE filtering of non-VSAM data sets using an alias.

For more information, see [“BY” on page 399](#) and [“EXCLUDE” on page 407](#).

z/OS UNIX files

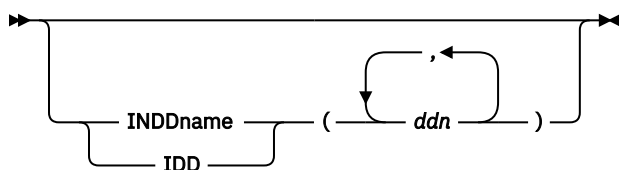
pnm

Specifies a path name, which is concatenated to the WORKINGDIRECTORY to form an absolute path to the file to be backed up. See [Chapter 17, “DFSMSdss filtering—choosing the z/OS UNIX files you want processed,” on page 265](#).

Note:

1. Relative path names are not supported.
2. Attributes are stored in the backup for any files that are selected during filtering for directories beginning after the WORKINGDIRECTORY. This includes attributes for any directories within a path specification.
3. If a path name resolves to a directory, only the attributes for the directory are processed. None of its members are processed. Recursion is not supported.

INDDNAME



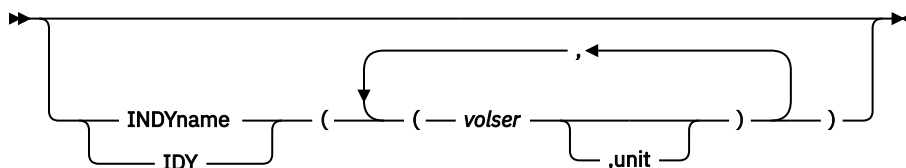
ddn

Specifies the name of the DD statement that identifies the input volume to be processed during FULL or TRACKS dump. To assure correct processing, each of the DD statements corresponding to a ddname (*ddn*) must identify only one volume serial number.

Note:

1. Only one *ddn* can be specified for INDDname when you use a full or tracks dump operation. One or more are allowed for a physical data set dump operation.
2. Specifying INDDNAME or INDYNAM results in a physical dump. To do a logical data set dump, either do not specify any input volumes with the DATASET keyword or use the LOGINDDNAME or LOGINDYNAM keywords.

INDYNAM



INDYNAM specifies that volumes to be dumped are to be dynamically allocated for a full or tracks dump.

volser

Specifies the volume serial number of a DASD volume to be dumped.

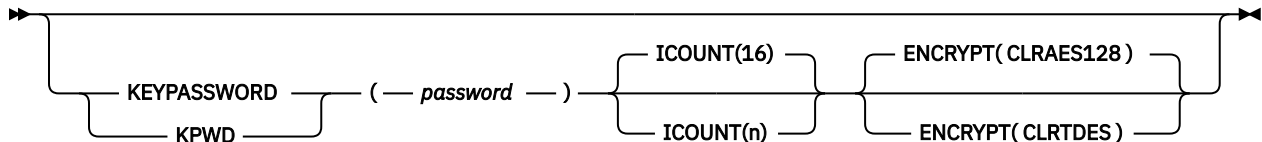
unit

Specifies the device type of a DASD volume to be dumped. This parameter is optional.

Note:

1. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*).
2. Only one volume is allowed for a full or tracks dump operation; one or more volumes, up to 511, are allowed for a physical data set dump operation.
3. Using INDYNAM instead of DD statements to allocate DASD volumes does not appreciably increase run time and permits easier coding of JCL and command input.
4. If either INDDNAME or INDYNAM is specified, *physical* processing is used to perform the dump. If both INDDNAME and INDYNAM are omitted, a *logical* data set dump is performed. A logical data set dump is also performed if you specify the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword.

KEYPASSWORD



KEYPASSWORD

Specifies the 8 to 32 character password (in EBCDIC) that is used to generate a clear TDES triple-length key or a clear 128-bit AES key.

Valid characters are upper and lower-case letters A through Z, numerals 0-9, and the following characters: !@#\$%&*-_=:<>?|{ }. You cannot use imbedded spaces, commas (,), forwardslash (/), parentheses (()), or semi-colons. DFSMSdss removes leading and trailing blanks.

ICOUNT

The ICOUNT optional parameter specifies how many times DFSMSdss performs the SHA-1 hash algorithm in the generation of the data key and initial chaining vector for encryption. n is an integer between 1 and 10000.

If you do not specify ICOUNT, the default number of iterations is 16.

ENCRYPT

The ENCRYPT keyword allows you to specify the type of encryption to use. The data key used is generated from the password you specified on the KEYPASSWORD keyword. If the same password is specified on separate DUMP commands, the same data key will be generated for a particular encryption type. The types of encryption are:

CLRAES128

Specifies that the dumped data is encrypted with a clear 128-bit AES key. It will be done using CPACF on a z9 or z10 processor. On any other processor (z900, z800, z990, or z890), the AES cryptography is done by the ICSF software.

CLRTDES

Specifies that the dumped data is encrypted with a secure triple-length DES key. It will use CPACF on a z890, z990, z9, and z10 processor. On a z900 and z800, you will need to start ICSF in order to perform the DES cryptography.

If you do not specify ENCRYPT, the default type of encryption is CLRAES128.

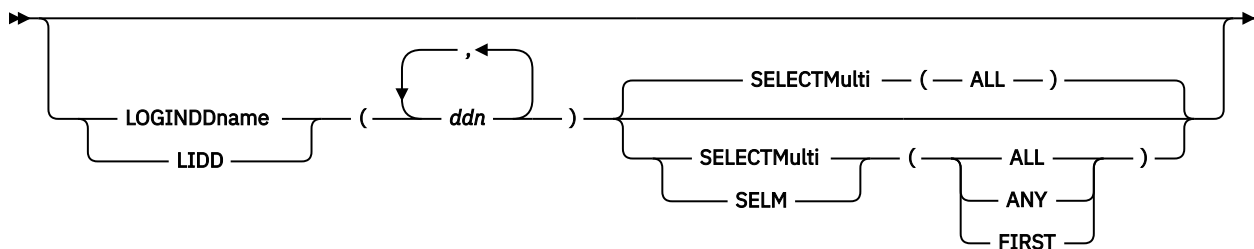
Note:

1. When you specify KEYPASSWORD, the only types of encryption that are allowed are CLRTDES and CLRAES128. Secure Triple DES (ENCTDES) is not allowed.
2. When using the KEYPASSWORD keyword, you must take care to ensure that the password is not lost or forgotten. If you lose or forget the password, DFSMSdss cannot decrypt the encrypted data on the dump data set. No password recovery mechanism exists. Neither the password or the generated data key is stored on the output medium.

3. Use of the HWCOMPRESS keyword is recommended when using the ENCRYPT keyword.
4. The KEYPASSWORD keyword is mutually exclusive with the RSA keyword.
5. The KEYPASSWORD password that is specified in your input command stream is not printed in the SYSPRINT output.
6. The ICSF address space must be started up successfully regardless of the processor you are running DFSMSdss on and the ENCRYPT sub-parameter you use.

For more information on the ENCRYPT keyword, see [“RSA” on page 419](#).

LOGINDDNAME



ddn

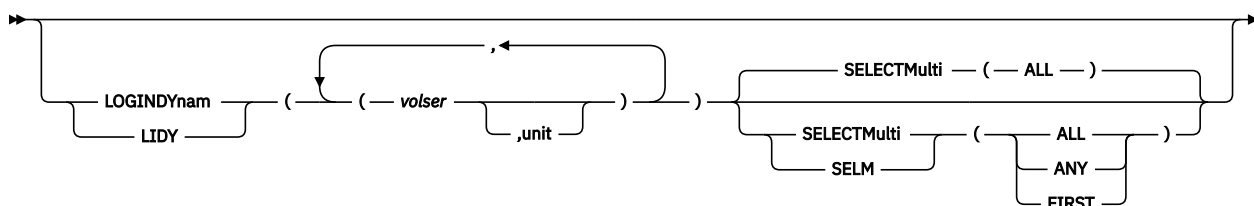
Specifies the name of the DD statement that identifies the input volume that contains the data sets for a logical dump operation. To ensure correct processing, each of the DD statements corresponding to a DDNAME (*ddn*) must identify only one volume serial number.

Notes for LOGINDDNAME, LOGINDYNAM, and STORGRP keywords:

1. When you specify the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword, DFSMSdss uses logical processing to perform the dump operation. Logical processing also occurs when no input volume is specified.
2. A multivolume data set that has extents on volumes that are not specified with the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword will not be dumped unless you specify SELECTMULTI.

For information about the SELECTMULTI keyword, see [“LOGINDYNAM” on page 412](#).

LOGINDYNAM



LOGINDYNAM specifies that volumes that contain the data sets to be dumped using logical processing are to be dynamically allocated. You can specify up to 511 volumes.

Note: The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*).

volser

Specifies the volume serial number of a DASD volume to be dumped.

unit

Specifies the device type of a DASD volume to be dumped. This parameter is optional.

SELECTMULTI

Specifies the method for determining how cataloged multivolume data sets are to be selected during a logical data set dump operation. SELECTMULTI is accepted only when logical volume filtering is specified with one of the following keywords:

- LOGINDDNAME
- LOGINDYNAM
- STORGRP

If logical volume filtering is not used, the specification of SELECTMULTI is not accepted.

ALL

Specifies that DFSMSdss *not dump* a multivolume data set unless the following criteria are met:

- The volume list that is created by the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword must list all the volumes that contain a part of the data set.
- The volume list that is created by the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword must list all the volumes that contain a part of the VSAM cluster.

ALL is the default.

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

- Specify SPHERE and you only need to 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.

ANY

Specifies that DFSMSdss dump a multivolume data set when any part of the non-VSAM data set or VSAM base cluster is on a volume in the volume list created by the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword.

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

- Specify SPHERE and you only need to 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.

FIRST

Specifies that DFSMSdss dump a multivolume data set only when the volume list includes the volume that contains the first part of the data set. The volume list is created by the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword.

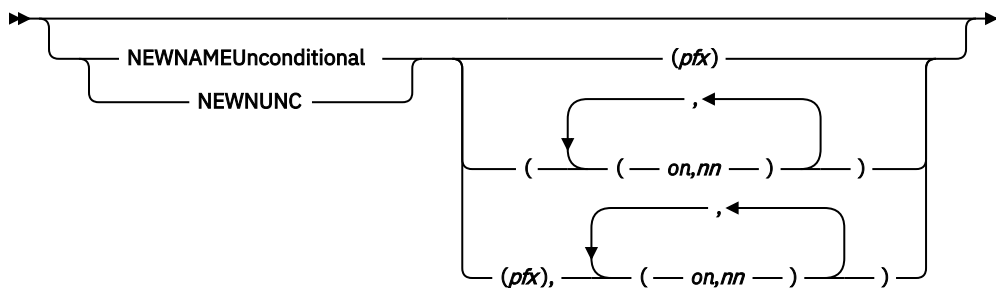
For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:

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

Notes for LOGINDDNAME, LOGINDYNAM, and STORGRP keywords:

1. If the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword is specified, DFSMSdss uses logical processing to perform the dump operation. Logical processing is also used if no input volume is specified.
2. A multivolume data set that has extents on volumes not specified with the LOGINDDNAME, LOGINDYNAM or STORGRP keyword will not be dumped unless you specify SELECTMULTI.

NEWNAMEUNCONDITIONAL



NEWNAMEUNCONDITIONAL specifies a new name for a source data set during logical dump processing. This keyword provides an alternative to renaming data sets during RESTORE processing. You might also find this keyword to be useful in avoiding name contention between data sets that are backed up (dumped) and data sets that are active on your system.

NEWNAMEUNCONDITIONAL can only be used when invoking DFSMSdss through the application programming interface (API).

NEWNAMEUNCONDITIONAL specifies that a source data set should be given a new name during dump processing, regardless of whether or not a data set already exists with the same name.

pfx

Specifies the prefix to be used to replace the first-level qualifier of the data set name. It is optional. If you use it, you must specify it as the first parameter in the list of sub-keywords. You might want to use a prefix only if you do not specify the (*on,nn*) parameters, or if the old name filters do not match the data set name.

on

Specifies the old name to be used as a filtering criterion.

nn

Specifies the new name to be used if the data set name matches the old name filtering criterion (*on*).

The syntax rules for the (*on,nn*) parameters are the same as those for the RENAME keyword in a RESTORE operation. If the new name filter has errors, the data set is not dumped and an error message is issued. For data set naming conventions, see [“INCLUDE” on page 409](#).

Note:

1. DFSMSdss does not verify whether the attributes of the source data set match those of the new-named data set. It is the responsibility of the program invoking DFSMSdss to perform this verification. This could cause the RESTORE operation to fail when attempting to restore to an existing data set when the attributes of the data set you are trying to restore to do not match.
2. If the new name is not fully qualified, it must contain the same number of qualifiers as the old name. For example, given the old-name filter DATE.** and the new-name filter DATE.**.LIST, DATE.MARCH.TODAY.OLDLIST is renamed, but DATE.MARCH.OLDLIST is not.
3. To change the number of qualifiers, specify fully-qualified names, for example, NEWNAMEU((A.B.C,A.B.C.D)).
4. If necessary, DFSMSdss truncates the new name to 44 characters. If the new name ends with a period, that period is also truncated.
5. You cannot use GDG relative generation filtering for old or new names.

NOTIFYCLONE

Refer to the [“CLONE” on page 400](#) description.

NOTIFYCONCURRENT

See [“CONCURRENT” on page 320](#).

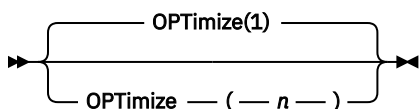
NOVALIDATE

See [“VALIDATE” on page 424](#).

ONLYINCAT

See [“INCAT” on page 409](#).

OPTIMIZE



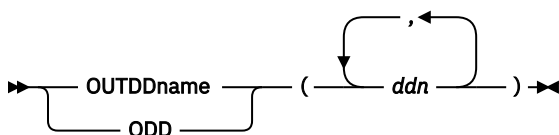
OPTIMIZE specifies the number of tracks to be read at a time, as follows:

- If *n* is 1, DFSMSdss reads one track at a time.
- If *n* is 2, DFSMSdss reads two tracks at a time.
- If *n* is 3, DFSMSdss reads five tracks at a time.
- If *n* is 4, DFSMSdss reads one cylinder at a time.

If OPTIMIZE is not specified, OPTIMIZE(1) is the default. Specifying OPTIMIZE (2), (3), or (4) reduces the read time for a dump. Notice that this keyword uses more real and virtual storage. It also keeps the channel busy for longer blocks of time. See [“Performance considerations” on page 59](#) for more information.

Recommendation: To improve performance and save dump space, specify the OPTIMIZE keyword with the COMPRESS keyword.

OUTDDNAME



ddn

Specifies the name of the DD statement that identifies the (output) dump data set. This data set can be on a tape or a DASD volume. Up to 255 ddnames can be specified; that is, up to 255 dump copies can be made.

Note:

1. The default block size for output records that are written to tape is determined by obtaining the optimum block size for the device. The maximum is 262 144. You can change this default to 32 760 bytes by using the installation options exit routine
2. The default block size for output records written to DASD is the track length for devices whose track length is less than 32KB (KB equals 1 024 bytes), or one-half the track length for devices whose track length is greater than 32KB. If the data set is a PS-E, then you may need to specify suitable block size to avoid inefficient DASD space usage.
3. If the DCB keyword BLKSIZE is specified on the DD statement for tape, it must be in the range of 7 892 through 262 144. If the DCB keyword BLKSIZE is specified on the DD statement for DASD, it must be in the range of 7 892 through 32 760.
4. The COPYDUMP command cannot change the block size of the DFSMSdss dump data set. If you intend to copy the dump data set to a DASD device, you must ensure that the block size will be small enough to fit on the target device.

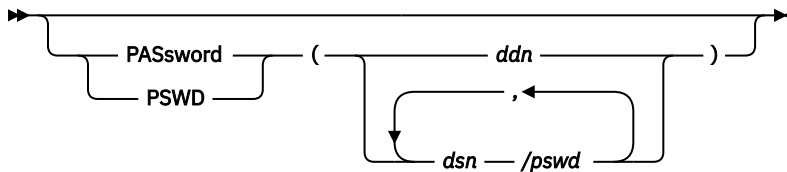


Attention: COPYDUMP is the only supported method for copying DFSMSdss dump data sets. Using a copy produced by any other method or utility as input to a RESTORE operation can produce unpredictable results.

5. If the DCB keyword RECFM is specified on the DD statement, it must have a value of "U".
6. If the DCB keyword LRECL is specified on the DD statement, it must have a value of "0" (zero).
7. When creating multiple dump copies in one step, DFSMSdss uses the largest 'common' block size that the output DDs can handle, regardless of the block size determined at open time. For example, if an output dump data set is opened with a block size of 65520 and a second output dump data set is opened with a block size of 262144, DFSMSdss writes at most 65520 bytes of data in a block to each output dump data set.

For more information about the installation options exit routine, see [z/OS DFSMS Installation Exits](#).

PASSWORD



PASSWORD specifies the passwords that DFSMSdss uses for password-protected data sets for all dump operations. (DFSMSdss bypasses password checking for RACF-protected data sets.) DFSMSdss requires this keyword only when the following apply:

- You do not have the required access for volume-level RACF DASDVOL or RACF DATASET.
- The installation authorization exit does not bypass the checks.
- You do not want a prompt for the VSAM data sets password.

Note:

1. Specify the passwords for all data sets that do not have RACF protection but do have password protection. A utility invoked by DFSMSdss may prompt the operator for a password during processing. You can control authorization checking by using the installation authorization exit.
2. Actual data set passwords that are specified in your input command stream are not printed in the SYSPRINT output.

The preferred method of protection is catalog protection through an access control facility, such as RACF. Catalog passwords are not supported to facilitate disaster recovery operations, application data transfers, and data set migration.

ddn

Specifies the name of the DD statement that identifies the sequential data set or member of a partitioned data set that contains data set names and their passwords. This data set must contain card-image records in DFSMSdss command syntax format.

dsn/pswd

dsn is a fully qualified data set name. *pswd* is its password. If no password follows the slash (/), *dsn* is treated as though it were *ddn*.

PATH

➡ Path ➡

PATH specifies a z/OS UNIX file dump operation using filtering. See [Chapter 17, “DFSMSdss filtering—choosing the z/OS UNIX files you want processed,” on page 265](#) for an explanation of how to specify what z/OS UNIX files are to be processed.

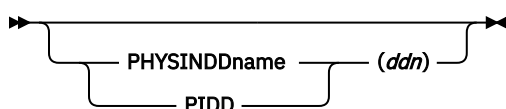
It is important to state that processing of z/OS UNIX files (regardless of I/O interface4), would require that a file's user data be read-in to buffers in a decompressed/de-ciphered format. The data can then be compressed when being stored in the DFSMSdss backup (using PATH processing supported DFSMSdss keywords).

PATH is mutually exclusive with FULL, TRACKS and DATASET.

Note:

1. INCLUDE and WORKINGDIRECTORY must be specified when PATH is selected.
2. Relative path naming is not supported.
3. The number of files processed is limited by the maximum number of tokens UNIX System Services (USS) imposes via its callable services. For further information on USS callable services, refer to the z/OS UNIX System Services File System Interface Reference.
4. I/O interface refers to whether CLONE processing is requested or not.

PHYSINDD

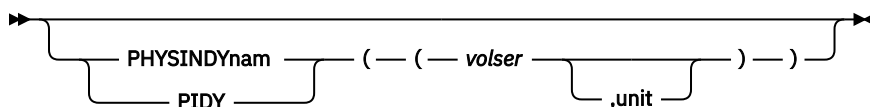


PHYSINDD

may be specified to request that DFSMSdss perform a physical copy or dump operation. It specifies a ddname that describes an input volume to be used for the copy or dump operation. Only one ddname may be specified. The device described by the ddname must be the same type as the output device specified on the OUTDD or OUTDYNAM keyword.

Note: PHYSINDD may be abbreviated to PIDD.

PHYSINDYNAM

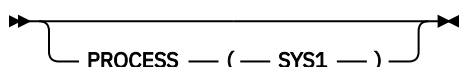


PHYSINDYNAM

specifies the input volume that is to be dynamically allocated to a physical copy or dump operation. A nonspecific volume serial number cannot be specified by using an asterisk (*). Only one volume may be specified for a FULL, TRACKS, or DATASET COPY. The device described by the volser must be the same type as the output device specified on the OUTDD or OUTDYNAM keyword.

Note: PHYSINDYNAM may be abbreviated to PIDY.

PROCESS

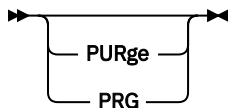


SYS1

specifies that DFSMSdss allows data sets with a high-level qualifier of SYS1 to be dumped and that SYS1 data sets can be deleted and uncataloged. SYS1.VVDS and SYS1.VTOCIX data sets can be physically, but not logically, dumped. SYS1.VVDS and SYS1.VTOCIX data sets cannot be deleted or uncataloged. To specify PROCESS(SYS1), RACF authorization may be required.

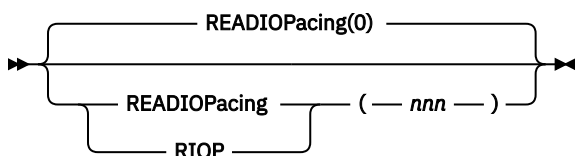
For more information about RACF authorization, see [Chapter 5, “Protecting DFSMSdss functions,”](#) on page 35.

PURGE



PURGE for a data set dump operation, specifies deletion of unexpired data sets that are dumped successfully. This keyword is valid only when DELETE has been specified.

READIOPACING



READIOPACING specifies the pacing (that is, I/O delay) to be used for DFSMSdss DASD read channel programs. You can use this keyword to allow more time for other applications to complete I/O processing. DFSMSdss waits the specified time before issuing each channel program that reads from DASD.

nnn

Specifies the amount of time in milliseconds. The maximum delay that can be specified is 999 milliseconds.

Note:

1. If READIOPACING is not specified, there will be no I/O delay.
2. The additional wait time does not apply to error recovery channel programs.
3. READIOPACING does not apply to concurrent copy I/O.

RESET



Data set and volume

RESET specifies that the data set changed flag in the VTOC entry is reset for all data sets that are serialized and dumped successfully. This applies to both a full dump and a data set dump operation.

Note:

1. Do not specify SHARE or FCWITHDRAW if you specify RESET.
2. You might not want to specify RESET if you use a storage management program, such as DFSMSHsm.
3. DFSMSdss ignores the RESET keyword when a data set is dumped using record-level sharing (RLS) access.
4. Using the RESET keyword for a logical data set dump operation causes the enqueue on a data set to be held until all data sets are dumped. DFSMSdss does not reset the data set change indicator until after all data sets are dumped. This may be of particular interest when dumping user catalogs. That is because delays for other jobs that need access to the user catalog may be caused by DFSMSdss holding the enqueue for the user catalog until all the data sets are dumped. This may cause a lockout condition when another job is dumping the same catalog at the same time.
5. You may specify both CONCURRENT and RESET, but RESET is ignored and a warning message is printed unless your installation uses a patch to tell DFSMSdss to accept RESET with CONCURRENT and to reset the data set changed indicator after the concurrent copy initialization is complete. If your installation uses the patch, it is possible that the data set changed indicator is left reset (off) even

when the dump of the data set is not successful. If this is unacceptable, do not specify RESET with CONCURRENT.

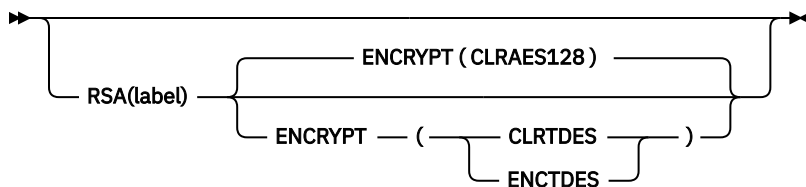
For more information about DFSMSdss patches, see [Chapter 14, “DFSMSdss patch area,”](#) on page 213.

z/OS UNIX files

RESET specifies the regular file's reference time in the attributes (ATTRREFTIME), which indicates the date of the last backup be updated for all regular files that are serialized and dumped successfully.

1. CLONE processing is restricted with RESET, See the description for Clone for further details.
2. Regular files that reside within a READ-only mounted file system cannot be updated.
3. User must have WRITE access to the file.

RSA



RSA

The RSA keyword allows you to specify the label of an existing RSA public key that is present in the ICSF PKDS. The RSA public key is used to encrypt a randomly generated data key, so that the encrypted data key can be stored on the output medium.

ICSF only allows labels for RSA keys to be up to 64 characters long. The first character must be alphabetic or a national character (#, \$, @). The remaining characters may be alphabetic, numeric, national, or a period.

Note:

1. You can also specify the label of an RSA public/private key pair. ICSF uses the public key when encrypting the data key.
2. The RSA keyword cannot be specified with the KEYPASSWORD keyword.
3. When using ENCTDES, or running on z800/z900 hardware, ensure that the RSA key is an internal key. Under these scenarios, an external RSA key will not be accepted by ICSF during the restore of the data.

ENCRYPT

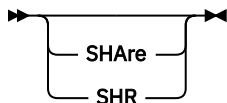
The ENCRYPT keyword allows you to specify the type of encryption key and the type of encryption that DFSMSdss performs on the dumped data. You can specify one of the following options. If you do not specify the ENCRYPT keyword, CLRAES128 is the default. If you specify ENCRYPT with the RSA keyword, the data key is randomly generated for each DUMP command.

- CLRTDES - This option specifies that the dumped data is encrypted with a clear triple-length DES key.
- CLRAES128 - This option specifies that the dumped data is encrypted with a clear 128-bit AES key
- ENCTDES - This option specifies that the dumped data is encrypted with a secure triple-length DES key.

SELECTMULTI

See [“LOGINDYNAM”](#) on page 412.

SHARE



Data sets

SHARE specifies that DFSMSdss is to share the data sets to be dumped for read access with other programs. Do not specify the DELETE, RESET, or UNCATALOG keyword if you specify SHARE. Use SHARE carefully to ensure that the contents of the dumped copy of the data set are valid.

Restriction: You cannot use the SHARE and FULL keywords at the same time.

Note:

1. Unlike the RESTORE command, the DUMP command honors the SHARE keyword for VSAM data sets. However, the SHARE keyword is only honored for VSAM data sets that were defined with share options other than (1,3) or (1,4).

Specifying the SHARE keyword does not cause DFSMSdss to honor the share options that are defined for VSAM data sets. For VSAM data sets that are defined with share options other than (1,3) or (1,4), specifying the SHARE keyword allows other programs to obtain read access, but not write access, to the data sets while they are being dumped. For VSAM data sets that are defined with share options (1,3) or (1,4), neither read access nor write access by other programs is allowed while the data set is being dumped, regardless of whether SHARE was specified.

2. Do not use the SHARE keyword during a physical dump of HFS or zFS data sets.
3. The SHARE keyword is required to logically dump mounted HFS data sets in DFSMSdss releases prior to DFSMSdss Release 1.5. The SHARE keyword is no longer required to logically dump mounted HFS data sets beginning in DFSMSdss Release 1.5.
4. For an HFS data set, DFSMSdss obtains both an ADRDSN enqueue and a SYSZDSN enqueue. SHARE determines only whether the ADRDSN enqueue is shared or exclusive.
5. For a zFS data set, DFSMSdss obtains an ADRDSN enqueue, a SYSDSN enqueue, and a number of SYSVSAM enqueues. SHARE determines only whether the ADRDSN enqueue is shared or exclusive. When specifying DELETE, DFSMSdss attempts to obtain exclusive SYSDSN and SYSVSAM enqueues. If you do not specify DELETE, DFSMSdss attempts to obtain shared SYSDSN and SYSVSAM enqueues.

For more information about dumping HFS or zFS data sets, see [“Backing up data sets with special requirements” on page 48](#).

For more information about dumping HFS data sets, see [“Dumping HFS data sets” on page 564](#).

For more information about dumping zFS data sets, see [“zFS data sets” on page 565](#).

z/OS UNIX files

SHARE specifies that DFSMSdss is to share the regular file to be dumped for read access with other programs. SHARE is the default specification when processing z/OS UNIX regular files. This option cannot be changed.

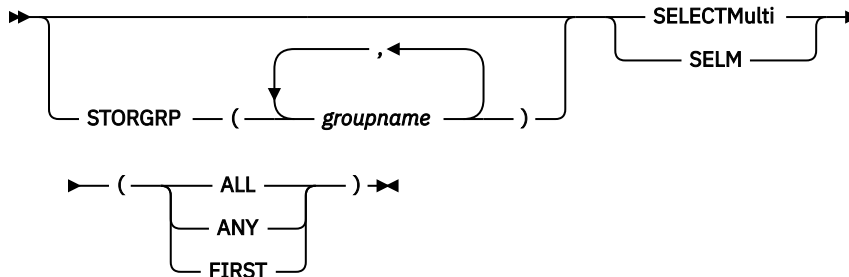
SPHERE



SPHERE is an option for a logical data set dump. SPHERE specifies that for any VSAM cluster dumped DFSMSdss must also dump all associated AIX clusters and paths. Individual sphere components need not be specified, only the base cluster name.

Note:

1. The base cluster name must be specified to process the entire sphere. If the SPHERE keyword is specified but the base cluster name is not, none of the associations will be processed.
2. If an AIX is dumped without the SPHERE keyword, during a restore DFSMSdss treats the AIX as a normal VSAM KSDS.

STORGRP

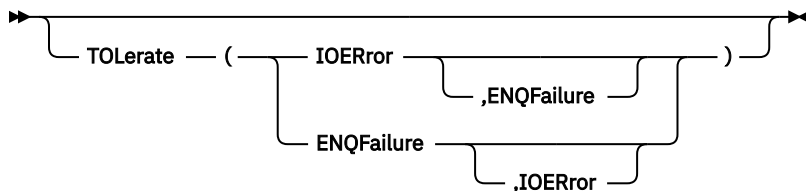
STORGRP specifies that all of the online volumes in the storage group be dynamically allocated. If a volume in the storage group is not online, that volume is not used for processing. You can specify up to 255 storage group names. Specifying STORGRP with a storage group name is equivalent to specifying LOGINDYNAM with all the online volumes in the storage group included in the list.

You can specify the STORGRP keyword with the SELECTMULTI keyword, but STORGRP cannot be used at the same time with the INDDname, INDYnam, LOGINDDname, or LOGINDYnam keywords.

See “LOGINDYNAM” on page 342 for a description of the SELECTMULTI keyword.

Notes for LOGINDDNAME, LOGINDYNAM, and STORGRP keywords:

1. If the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword is specified, DFSMSdss uses logical processing to perform the dump operation. Logical processing is also used if no input volume is specified.
2. A multivolume data set that has extents on volumes not specified with the LOGINDDNAME, LOGINDYNAM, or STORGRP keyword will not be dumped unless you specify SELECTMULTI.

TOLERATE**Data set and volume****ENQFailure**

specifies that data sets are to be processed even though shared or exclusive access fails.

TOL(ENQF) and FULL or TRACKS are mutually exclusive; you cannot specify these keywords together.

Note:

1. Unlike PDS data sets, PDSE data sets that are open for update cannot be dumped even if TOL(ENQF) is specified.
2. If you must dump a PDSE data set and it must be open for update, process the data set with concurrent copy (specify CONCURRENT). If you cannot use concurrent copy, convert the PDSE back to PDS and then dump the PDS data set with TOL(ENQF).

3. TOL(ENQF) cannot be used when processing a **logical** dump of HFS or zFS data sets. HFS or zFS data sets cannot be dumped when the HFS or zFS data set is mounted on a system other than the system where the dump is being performed. TOL(ENQF) is not required when dumping an HFS data set or zFS data set is mounted on the same system where the dump is being performed.
4. Exercise caution if you use TOL(ENQF) during a **physical** dump of HFS data sets. Unlike other types of data sets, if an HFS data set is updated during a physical dump with TOL(ENQF), a subsequent restore can likely result in an unusable data set.

For more information about dumping HFS or zFS data sets, see [“Backing up data sets with special requirements”](#) on page 48. For more information about TOL(ENQF), see [Chapter 22, “Data integrity—serialization,”](#) on page 561.

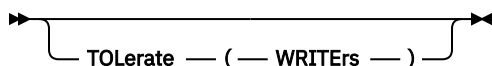
IOError

specifies that DFSMSdss is to continue processing even though input errors occur, but is to end after 100 errors. This applies only to input errors and only to equipment check and busout parity.

Note:

1. TOL(IOError) is ignored if CANcelerror is specified.
2. If a permanent read error occurs, the track image record is flagged on output as having an I/O error and the dump processing continues.
3. This track is cleared in a restore operation.

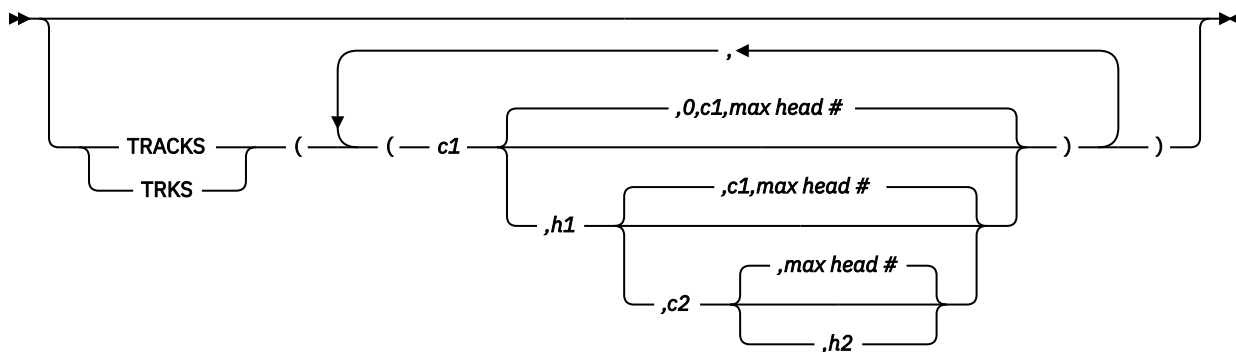
z/OS UNIX files



Writers

Specifies that a backup of a regular file should continue if DFSMSdss found that the file was concurrently open with write intent by some other application.

TRACKS



TRACKS specifies ranges of tracks to be dumped. When you restore the data, this entire range or its subset must be specified with the RESTORE command.

Restriction: You cannot use the TRACKS keyword with the TOL(ENQF) keyword.

c1,h1

Specifies the cylinder and head number of the beginning of the range. Specify hexadecimal numbers as X'c1' or X'h1'.

c2,h2

Specifies the cylinder and head number of the end of the range. Specify hexadecimal numbers as X'c2' or X'h2'.

You can enter X'FFFFFFF' (or 268435455) as the high cylinder value. This causes DFSMSdss to choose the high cylinder value to be the end of the volume.

Note:

1. The *c2* must be greater than or equal to *c1*.
2. If *c2* equals *c1*, *h2* must be greater than or equal to *h1*.

DFSMSdss verifies that the range is within the limits of the device. If you do not specify all four values for a range, DFSMSdss provides the missing values unless the omitted value causes a syntax error. No intervening values can be omitted. For example:

**Specified
Results**
None

Syntax error

c1

c1,0,c1, maximum head number

c1,h1

c1,h1,c1, maximum head number

c1,h1,c2

c1,h1,c2, maximum head number

c1,,c2

Syntax error

,h1

Syntax error

c1,h1,X'FFFFFF'

c1,h1,maximum cylinder number for the volume, maximum head number

For more information about using TRACKS during a physical dump operation, see [Chapter 6, “Managing availability with DFSMSdss,”](#) on page 39.

UNCATALOG



UNCATALOG applies to physical and logical data set dump operations.

For a *physical* data set dump operation, UNCATALOG instructs DFSMSdss to uncatalog any single-volume, non-VSAM, cataloged data sets successfully dumped from the current volume.

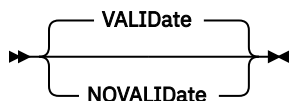
For a *logical* data set dump operation, UNCATALOG instructs DFSMSdss to uncatalog any successfully dumped single or multivolume non-VSAM data sets that are currently cataloged. (For VSAM or SMS-managed data sets, use the DELETE keyword.)

Note:

1. UNCATALOG is ignored for VSAM and SMS-managed data sets.
2. Do not specify UNCATALOG with CONCURRENT, because after the concurrent copy operation has begun, the original data can still be updated.
3. For a logical data set dump operation, the use of the UNCATALOG keyword causes the enqueue in a data set to be held until all data sets are dumped. DFSMSdss does not uncatalog the data set until after all data sets are dumped.
4. Any non-SMS, non-VSAM data set that has a high-level qualifier of SYS1 cannot be uncataloged unless PROCESS(SYS1) is specified.

For information about a patch to modify the UNCATALOG algorithm, see [“Using RESET or UNCATALOG in a logical data set dump \(PN60114\)”](#) on page 222.

VALIDATE

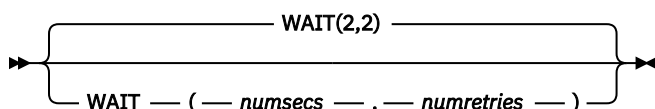


VALIDATE on a logical data set dump, specifies that all indexed VSAM data sets are to be validated as they are dumped. If the NOVALIDATE keyword is specified, the indexed VSAM data sets are dumped without validation. VALIDATE is the default.

Note:

1. If an indexed VSAM data set is dumped using VALIDATE, it must be restored on a system that supports VALIDATE. Otherwise, an error message is issued, and the restore fails.
2. Do not specify the NOVALIDATE keyword when processing VSAM extended-format or extended-addressable data sets.
3. When a data set is restored, the free space in the control areas and control intervals are reset to the values in the catalog entry. You can override the values in the catalog entry by specifying the FREESPACE keyword on the restore.
4. Use the NOVALIDATE keyword on the DUMP command if you wish to restore to a DFDSS Version 2 Release 5 system that does not have the appropriate VALIDATE support, or to any level of DFDSS prior to Release 2 Version 5.
5. DFSMSdss does not support VALIDATE processing when backing up encrypted indexed VSAM data sets. If VALIDATE is specified it will be ignored during processing of a VSAM encrypted data set.

WAIT



For physical data set dump processing

WAIT specifies to DFSMSdss the length of a wait and the number of retries to obtain control of a data set.

numsecs

Specifies a decimal number from 0 to 255 that designates the interval, in seconds, between retries.

numretries

Specifies a decimal number from 0 to 99 that designates the number of times DFSMSdss must retry to gain control of a data set.

For logical data set dump processing

WAIT specifies to DFSMSdss the length of wait and the number of passes to obtain control of a data set.

numsecs

Specifies a decimal number from 0 to 255 that designates the interval, in seconds, to wait before attempting another pass through the list of selected data sets.

numretries

Specifies a decimal number from 0 to 99 that designates the number of passes to make through the list of selected data sets. Each pass is an attempt to obtain control of a data set.

The default for *numsecs,numretries* is (2,2), which specifies two retries at 2-second intervals. If you do not want to wait for a data set, specify 0 for either numsecs or numretries.

For logical data set dump operation, the WAIT keyword has a different meaning when: (1) data sets are being serialized, (2) multiple data sets are being processed, and (3) WAIT(0,0) is not specified. In this case, DFSMSdss makes multiple passes through the list of selected data sets. On each pass, DFSMSdss

processes the data sets that (1) can be serialized without waiting for the resource and (2) were not processed before. At the end of a pass, if none of the data sets could be processed without waiting for a resource, then, in the next pass, at the first occurrence of a data set that was not processed, a WAIT will be issued. That data set and the remainder of the list will be processed if possible. The above procedure will be repeated until all data sets are processed or the WAIT limits are reached. For example, if WAIT(3,10) is specified and 5 data sets are left to be processed, up to 10 passes are made. On each pass, an unprocessed data set is waited upon for 3 seconds. Thus, only a 30-second maximum will ever be waited, not 150 (5 times 3 times 10).

Note: The WAIT keyword does not control wait/retry attempts for system resources (such as the VTOC and the VVDS). For system resources, the default wait time is 3 seconds and the default retry count is 30. This results in a total wait time of 90 seconds.

For more information about controlling the wait/retry attempts for system resources, see [“Controlling the wait/retry time for serialization of system resources \(PN11523\)”](#) on page 215.

For z/OS UNIX file dump processing

WAIT specifies to DFSMSdss the length of a wait and the number of retries to obtain an open context of a regular file. Access to the user data of a regular file requires that DFSMSdss open a file to obtain an open context (open token) so that it can read from the file.

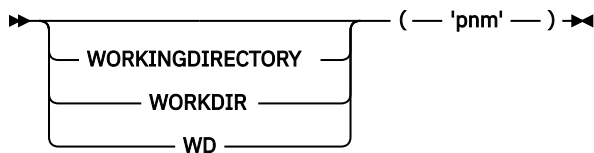
numsecs

Specifies a decimal number from 0 to 255 that designates the interval, in seconds, between retries.

numretries

Specifies a decimal number from 0 to 99 that designates the number of times DFSMSdss must retry the open of the regular file.

WORKINGDIRECTORY

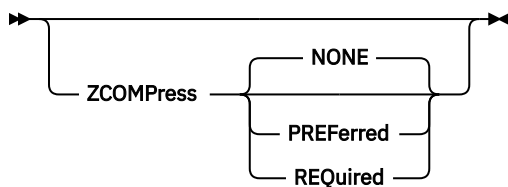


WORKINGDIRECTORY specifies the directory where DFSMSdss is to position itself in the z/OS UNIX hierarchy. It assists in setting the scope of the hierarchy the invoker is interested in. Any path specified in the INCLUDE criteria is concatenated with the WORKINGDIRECTORY to obtain an absolute path to the file(s) being processed. See [Chapter 17, “DFSMSdss filtering—choosing the z/OS UNIX files you want processed,”](#) on page 265.

Note:

1. Must be specified with PATH
2. Must be an absolute path.
3. File attributes for directories are stored in the backup for any paths beginning after the working directory path.

ZCOMPRESS



ZCOMPRESS specifies that DFSMSDss writes the dumped data in compressed format to the output medium using zEDC Services. This decreases the space that is occupied by the dumped data.

NONE requests that DFSMSDss not perform compression with zEDC Services. This is the default behavior and specifying this option is not necessary to bypass zCompression.

PREFERRED specifies that DFSMSDss use zEDC Services, if possible. If zEDC Services are not available or cannot be used, DFSMSDss continues processing normally.

REQUIRED specifies that compression using zEDC Services must be used. If they are not available or cannot be used, DFSMSDss fails the operation.

When encrypted data sets are dumped by data set, or reside within a volume that is dumped, the compression ratio can show a decreased ratio.

Notes:

1. For logical and physical dumps, ZCOMPRESS(PREFERRED) can be specified with either COMPRESS or HWCOMPRESS keywords, but not both. If zEDC Services cannot be used, DFSMSDss attempts to use the alternate compression method specified.
2. For z/OS UNIX file dumps, ZCOMPRESS is the only compression supported. COMPRESS and HWCOMPRESS are not supported. If ZCOMPRESS(PREFERRED) is specified during Path processing, DFSMSDss does not attempt an alternative compression format.
 - a. Files must be at least 4096 bytes in length for DFSMSDss to attempt compression.
3. You cannot use the stand-alone restore program with a dump that has been compressed with the ZCOMPRESS keyword.

Data integrity considerations for full or tracks dump operation

For a full or tracks dump operation, DFSMSDss serializes the VTOC to preclude DADSM functions (such as ALLOCATE, EXTEND, RENAME, and SCRATCH) from changing the contents of the VTOC on the volume during the dump operation. Data sets are *not* serialized on these full or tracks operations. Therefore, some data sets might be opened by other jobs during the dump operation, resulting in copies of partially updated data sets. You can minimize this possibility by performing the dump operation when there is low system activity.

Full data integrity can only be guaranteed by performing dump operations by data set when TOL(ENQF) or SHARE keywords are not specified.

Format of the output data set

For the format of the output data set, see [Chapter 6, “Managing availability with DFSMSDss,” on page 39](#).

Examples of full and tracks dump operations

In the following example, data from DASD volume 111111 is to be dumped to the first data set of standard label tape volumes TAPE01 and TAPE02.

The command input to be substituted for a full and tracks dump are shown below in Example 1A and Example 1B, respectively. To restore the same volume, refer to Examples 1, 1A, and 1B of the RESTORE command.

Example 1: a data set dump

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD      DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//TAPE      DD       UNIT=(3480,2),VOL=SER=(TAPE01,TAPE02),
// LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=USER2.BACKUP
//SYSIN     DD       *
command input
/*
```

See command input in “[Example 1A: a full dump operation](#)” on page 427 and “[Example 1B: a tracks dump operation](#)” on page 427.

Example 1A: a full dump operation

```
DUMP  INDDNAME(DASD) OUTDDNAME(TAPE)
```

Example 1B: a tracks dump operation

```
DUMP  TRACKS(1,0,1,5) INDDNAME(DASD) -
      OUTDDNAME(TAPE)
```

Example 1C: full volume dump operation with CONCURRENT

```
//DSSJOB JOB      accounting information,REGION=nnnnK
//DUMPSTEP EXEC     PGM=ADRDSSU
//SYSPRINT DD       SYSOUT=*
//DASD      DD       UNIT=SYSDA,VOL=SER=(SSDASD),DISP=OLD
//TAPE      DD       UNIT=TAPE,VOL=SER=(TAPE01,TAPE02,TAPE03),LABEL=(1,SL),
// DISP=(NEW,KEEP),DSN=USER.BACKUP
//SYSIN     DD       *
      DUMP FULL INDDNAME(DASD) OUTDDNAME(TAPE) -
      COMPRESS CONCURRENT
/*
```

This JCL does a DFSMSdss full-volume dump using concurrent copy. This job continues (with a warning message) if concurrent copy initialization fails. No special action is required to perform a restore operation after a concurrent copy dump operation.

Examples of physical data set dump operations

Example 2 depicts specified data sets on DASD volumes (numbered 111111 and 222222) that are being dumped to the first data set of standard label tape volume called TAPE02.

Examples 2A through 2G below complement examples 2A through 2D in the restore section, in any combination; for example, the dump tape produced in example 2C can be used as the input tape for example 2A under the RESTORE command.

Example 2: depicting DASD volume DUMP

```
//JOB2      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD1     DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//DASD2     DD       UNIT=3380,VOL=(PRIVATE,SER=222222),DISP=OLD
//TAPE      DD       UNIT=3480,VOL=SER=TAPE02,
// LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=USER2.BACKUP
//SYSIN     DD       *
command input
/*
```

See command input in [“Example 2A: using the INCLUDE subkeyword”](#) on page 428, [“Example 2B: using the INCLUDE and EXCLUDE subkeywords”](#) on page 428, [“Example 2C: using the INCLUDE, EXCLUDE, and BY subkeywords”](#) on page 428, [“Example 2D: with filtering data in a data set”](#) on page 428, [“Example 2E: with passwords in the input stream”](#) on page 428, [“Example 2F: with passwords in a data set”](#) on page 428, and [“Example 2G: wait for data sets if they or other data sets with the same name are in use by other jobs”](#) on page 429.

Example 2A: using the INCLUDE subkeyword

```
DUMP  INDDNAME(DASD1,DASD2) OUTDDNAME(TAPE) -
      DATASET(INCLUDE(USER2.**,USER3.*))
```

Example 2B: using the INCLUDE and EXCLUDE subkeywords

```
DUMP  INDDNAME(DASD1,DASD2) OUTDDNAME(TAPE) -
      DATASET(INCLUDE(USER2.**,USER3.*) -
              EXCLUDE(USER2.**.REP))
```

Example 2C: using the INCLUDE, EXCLUDE, and BY subkeywords

```
DUMP  INDDNAME(DASD1,DASD2) OUTDDNAME(TAPE) -
      DATASET(INCLUDE(USER2.**,USER3.*) -
              EXCLUDE(USER2.**.REP) -
              BY((DSCHA,EQ,1)))
```

Example 2D: with filtering data in a data set

```
DUMP  INDDNAME(DASD1,DASD2) OUTDDNAME(TAPE) -
      DATASET(FILTERDD(A1))
```

Note: The following DD statement must be added to the JCL shown above:

```
//A1 DD DSNAME=USER2.FILTER,DISP=SHR
```

This cataloged data set (USER2.FILTER) contains three card-image records. The information shown is positioned in columns 2 through 72 of each record:

```
INCLUDE(USER2.**,USER3.*) -
EXCLUDE(USER2.**.REP) -
BY((DSCHA,EQ,1))
```

Example 2E: with passwords in the input stream

```
DUMP  INDDNAME(DASD1,DASD2) OUTDDNAME(TAPE) -
      DATASET(INCLUDE(USER2.**,USER3.*)) -
      PASSWORD(USER2.ABC.DEF/PSWD1,USER2.XYZ/PSWD2)
```

Example 2F: with passwords in a data set

```
DUMP  INDDNAME(DASD1,DASD2) OUTDDNAME(TAPE) -
      DATASET(INCLUDE(USER2.**,USER3.*)) -
      PASSWORD(PDD)
```

Note: The following DD statement must be added to the JCL shown above:

```
//PDD DD DSNAME=USER2.PASSWORD,DISP=SHR
```


This cataloged data set (USER2.PASSWORD) contains a single card-image record. The information shown is positioned in columns 2 through 72:

```
USER2.ABC.DEF/PSWD1,USER2.XYZ/PSWD2
```

Example 2G: wait for data sets if they or other data sets with the same name are in use by other jobs

```
DUMP  INDDNAME(DASD1) OUTDDNAME(TAPE) -
      DATASET(INCLUDE(**)) -
      WAIT(1,99)
```

If a data set is in use, DFSMSdss waits for one second, then tries to gain access to the resource again. This is done as many as 99 times for each data set.

Example 2H: clearing volumes of uncataloged data sets

```
DUMP  DATASET(INCLUDE(**) -
             BY((DSORG NE VSAM) -
                (CATLG EQ NO))) -
      INDDNAME(DASD1,DASD2) -
      OUTDDNAME(TAPE) -
      DELETE PURGE
```

If you do not want a dump of the uncataloged data sets, the output ddname TAPE can be a dummy. DASD1 and DASD2 identify the input volumes. A physical data set dump can handle multiple uncataloged single-volume data sets with the same name if multiple volumes are specified. This is because each volume is processed in order, one at a time. The dump cannot handle a multivolume data set even if all the volumes on which it resides are specified as input volumes.

Examples of logical data set dump operations

This section contains examples of logical data set dump operations.

Example 1: dumping data sets constantly in use

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD1     DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//DASD2     DD       UNIT=3380,VOL=(PRIVATE,SER=222222),DISP=OLD
//TAPE      DD       UNIT=3480,VOL=SER=TAPE02,
// LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=USER2.BACKUP
//SYSIN     DD       *
DUMP  LOGINDDNAME(DASD1) OUTDDNAME(TAPE) -
      DATASET(INCLUDE(**)) TOL(ENQF) WAIT(0,0)
/*
```

DFSMSdss does not wait (WAIT(0,0)) if a data set is in use. Instead, it processes the data set without serialization or enqueueing (TOL(ENQF)).

Example 2: dumping a user catalog and its aliases

To dump a user catalog, you perform a logical data set dump with the fully qualified user catalog name as the data set name. No filtering is allowed. If the user catalog has any aliases, the aliases are automatically dumped.

```
//JOB2      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD1     DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//TAPE      DD       UNIT=3480,VOL=SER=TAPE02,
// LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=USER2.BACKUP
//SYSIN     DD       *
DUMP OUTDDNAME(TAPE) -
      DS(INCLUDE(MY.USER.CAT))
/*
```

Example 3: logical data set dump operation with catalog filtering

```
//JOB3      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//TAPE      DD       UNIT=3480,VOL=SER=TAPE04,
// LABEL=(1,SL),DISP=(NEW,CATLG),DSN=USER3.BACKUP
//SYSPRINT  DD       SYSOUT=A
//SYSIN     DD       *
DUMP OUTDD(TAPE) -
      DS(INCL(USER1.**))
/*
```

All data sets cataloged in the standard search order whose first-level qualifier is USER1 are to be dumped. Because some of these data sets are multivolume, source DASD volumes are not specified, resulting in data set selection by catalog.

Example 3 can be modified as follows to dump only data sets changed since the last backup. In addition, data sets that end with a qualifier of LISTING are not to be dumped (EXCL(**.LISTING)).

```
//SYSIN     DD       *
DUMP OUTDD(TAPE) -
      DS(INCL(USER1.**)) -
      EXCL(**.LISTING) -
      BY((DSCHA EQ 1)))
/*
```

Example 4: logical data set dump operation with VTOC filtering

```
//JOB4      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD1     DD       VOL=SER=338001,UNIT=3380,DISP=OLD
//TAPE      DD       UNIT=3480,VOL=SER=TAPE04,
// LABEL=(1,SL),DISP=(NEW,CATLG),DSN=USER3.BACKUP
//SYSIN     DD       *
DUMP DATASET(INCLUDE(USER3.**)) -
      LOGINDDNAME(DASD1) -
      OUTDDNAME(TAPE) -
      DELETE PURGE
/*
```

All data sets on volume 338001 whose first qualifier is USER3 are included in a logical data set DUMP. DFSMSdss filters using the VTOC of volume 338001. Catalogs are also used, as needed, for multivolume and VSAM data sets.

The previous example can be modified as follows to dynamically allocate volume 338001 and to do SELECTMULTI processing. All single volume data sets on volume 338001 are included in a logical data set dump operation. SELECTMULTI(ANY) specifies that a cataloged data set residing on volumes 338001, 338003, and 338005 will be dumped even though 338003 and 338005 are not in the LOGINDYNAM volume list.

```
//JOB4      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//TAPE      DD       UNIT=3480,VOL=SER=TAPE04,
// LABEL=(1,SL),DISP=(NEW,CATLG),DSN=USER3.BACKUP
//SYSIN     DD       *
DUMP DATASET(INCLUDE(**)) -
    SELECTMULTI(ANY) -
    LOGINDYNAM(338001) -
    OUTDDNAME(TAPE) -
    DELETE PURGE
/*
```

Example 5: logical data set dump operation for Storage Management Subsystem (SMS)

```
//JOB5      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//TAPE      DD       DSN=BACKUP(+1),DISP=(,CATLG),
// DCB=(SYS1.DFDSS.DSCB)
//SYSIN     DD       *
DUMP LOGINDYNAM(338001) -
    SELECTMULTI(FIRST) -
    DATASET(INCLUDE(**)) -
    OUTDDNAME(TAPE) -
    DELETE
/*
```

This example backs up the volume to a generation data set. You can use generation data set groups to create and manage multiple backup versions of a volume or data sets.

A volume is backed up for converting to or from SMS management by using a logical data set dump function.

Example 6: logical dump operation with CONCURRENT

```
//JOB6      JOB      accounting information,REGION=nnnnK
//DUMPSTEP  EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=*
//TAPE      DD       UNIT=TAPE,VOL=SER=(TAPE01,TAPE02,TAPE03),LABEL=(1,SL),
// DISP=(NEW,KEEP),DSN=USER.BACKUP
//SYSIN     DD       *
DUMP DATASET(INCLUDE(USER.LOG,USER.TABLE,USER.XREF)) -
    OUTDDNAME(TAPE) OPTIMIZE(4) CONCURRENT
/*
```

This JCL does a DFSMSdss logical data set dump of three fully qualified data sets using concurrent copy. This job continues with a warning message if concurrent copy initialization fails. No special action is required to perform a restore operation after a concurrent copy dump operation.

Example 7: clearing volumes of uncataloged data sets

```
//JOB7      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD1     DD       VOL=SER=MYVOL1,UNIT=SYSDA,DISP=OLD
//DASD2     DD       VOL=SER=MYVOL2,UNIT=SYSDA,DISP=OLD
//TAPE      DD       UNIT=3480,VOL=SER=TAPE04,LABEL=(1,SL),
// DISP=(NEW,CATLG),DSN=USER3.BACKUP
//SYSIN     DD       *
DUMP DATASET(INCLUDE(**)) -
    BY((DSORG NE VSAM) -
        (CATLG EQ NO))) -
    LOGINDDNAME(DASD1,DASD2) -
    OUTDDNAME(TAPE) -
    DELETE PURGE
/*
```

Logical data set dump cannot be used to dump SMS data sets that are not cataloged or are cataloged outside the standard order of search. DFSMSdss physical data set dump or IDCAMS DELETE NVR can be used for such cleanup operations.

If you do not want a dump of the uncataloged data sets, the output ddname TAPE can be a dummy. 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 if 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 no cataloged data set by the same name exists on the system.

Examples of file dump operation

This JCL does a DFSMSdss file dump of three absolute paths.

```
//JOB6      JOB      accounting information,REGION=nnnnK
//DUMPSTEP  EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=*
//TAPE      DD       UNIT=TAPE,VOL=SER=(TAPE01,TAPE02,TAPE03),LABEL=(1,SL),
//  DISP=(NEW,KEEP),DSN=USER.BACKUP
//SYSIN     DD       *
      DUMP PATH(INCLUDE('Documents/istio-1.01.1/License' -
                      'Documents/istio-1.01.1/README.md' -
                      'Documents/istio-1.01.1/istio.VERSION' )) -
      WORKINGDIRECTORY('/u/usr/ernestof') -
      OUTDDNAME(TAPE)
/*
```

PRINT command for DFSMSdss

With the PRINT command, you can print:

- A single-volume non-VSAM data set, as specified by a fully qualified name. You must specify the volume where the data set resides, but you do not need to specify the range of tracks it occupies.
- A single-volume VSAM data set component (not cluster). The component name specified must be the name in the VTOC, not the name in the catalog.
- Ranges of tracks.
- All or part of the VTOC. The VTOC location need not be known.

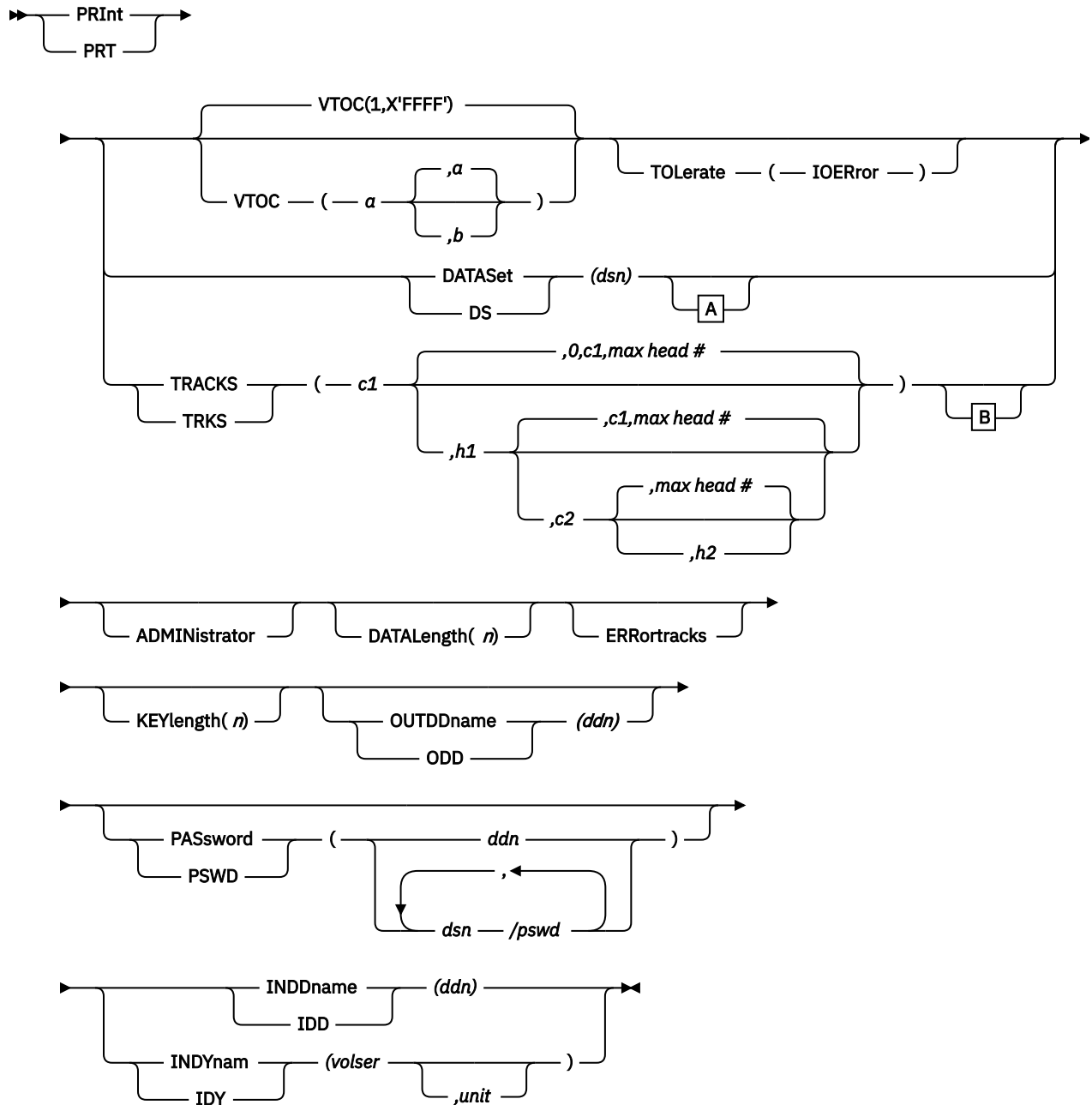
Note: In order to print a multivolume data set, multiple PRINT commands with the appropriate INDD/INDY keywords must be used.

Unless the ALLDATA keyword is specified, only the used space is printed for sequential or partitioned data sets or data sets with data set organizations of null.

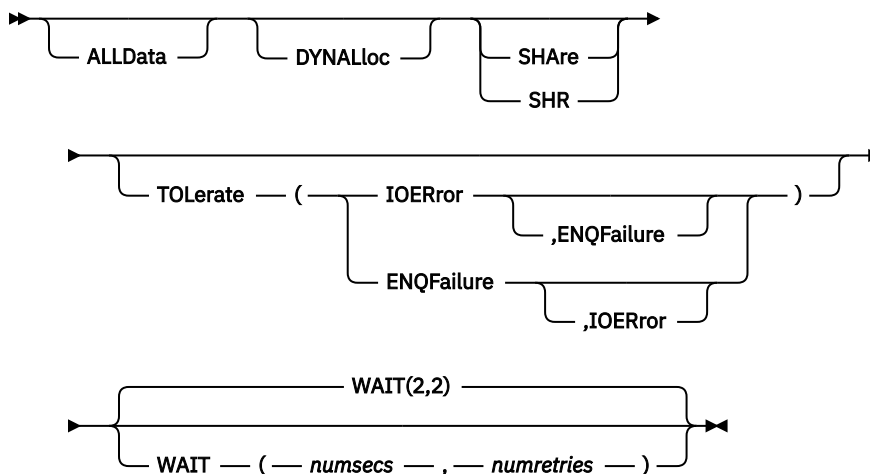
If an error occurs in reading a record, DFSMSdss attempts to print the record in error. You can print all requested tracks or just a subset of the tracks that have data checks.

Related reading: For additional information about authorization checking, see [Chapter 21, “Data security and authorization checking,”](#) on page 533.

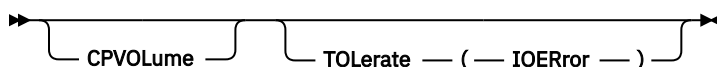
PRINT syntax



A: Optional Keywords with PRINT DATASET



B: Optional Keywords with PRINT TRACKS



Explanation of PRINT command keywords

This section describes the keywords for the PRINT command.

ADMINISTRATOR



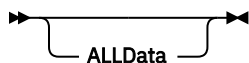
`ADMINISTRATOR` allows you to act as a DFSMSdss-authorized storage administrator for the PRINT command. DFSMSdss-initiated access checking to data sets and catalogs is bypassed. If you are not authorized to use the `ADMINISTRATOR` keyword, the command ends with an error message.

To use the `ADMINISTRATOR` keyword, all of the following conditions must be true:

- FACILITY class is active.
- The applicable FACILITY-class profile is defined.
- You have READ access to that profile.

Related reading: For additional information about the use of the `ADMINISTRATOR` keyword, see [“ADMINISTRATOR keyword” on page 542](#).

ALLDATA



`ALLDATA` specifies, when the `DATASET` keyword is also specified, that *all* allocated space in the data set is to be printed.

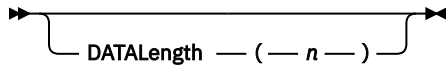
CPVOLUME



`CPVOLUME` specifies that the volume is VM-formatted and that the OS-compatible VTOC must begin on track zero, record five. The OS-compatible VTOC does not describe the extents of any data on the volume.

Therefore, you must specify the track ranges to be printed with the TRACKS keyword. CPVOLUME is only allowed with the ADMINISTRATOR keyword because DFSMSdss cannot check access authorization for VM data.

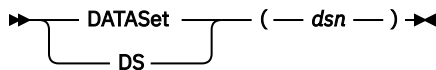
DATALENGTH



n

Specifies the logical length, in decimal format, of the data portion of a record. It is used only if the count field of a record on any track has a data check.

DATASET



dsn

Specifies the fully qualified name of the data set to be printed. The data set is printed in logical sequence.

Note: Data set filtering is not allowed with the PRINT command.

DYNALLOC



DYNALLOC specifies dynamic allocation, instead of enqueue, to serialize the use of data sets. This allows cross-system serialization in a JES3/MVS environment.

Consider:

- The serialization is of value only when the dynamic allocation/JES3 interface is not disabled.
- Run time increases when you use the DYNALLOC keyword to serialize data sets (as opposed to enqueue) because overhead is involved in dynamic allocation and serialization across multiple processors.

ERRORTRACKS



ERRORTRACKS specifies that only tracks on which data checks occur are to be printed.

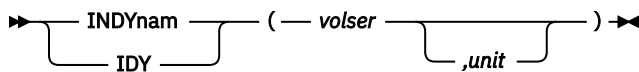
INDDNAME



ddn

Specifies the name of the DD statement that identifies a volume that contains the data set, range of tracks, or the VTOC to be printed. If you want to print a multivolume data set, you must print one volume at a time.

INDYNAM



INDYNAM specifies that the volume that contains the data set, range of tracks, or the VTOC to be printed is to be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*).

Using INDYNAM instead of DD statements to allocate DASD volumes does not noticeably increase run time and permits easier coding of JCL and command input.

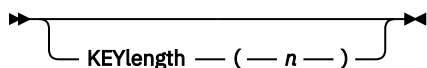
volser

Specifies the volume serial number of a DASD volume to be printed.

unit

Specifies the device type of a DASD volume to be printed. This parameter is optional.

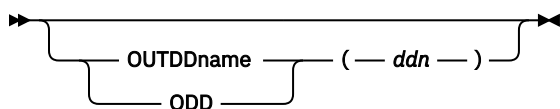
KEYLENGTH



n

Specifies the key length, in decimal format, of a record. It is used only if the count field of a record on any track has a data check.

OUTDDNAME



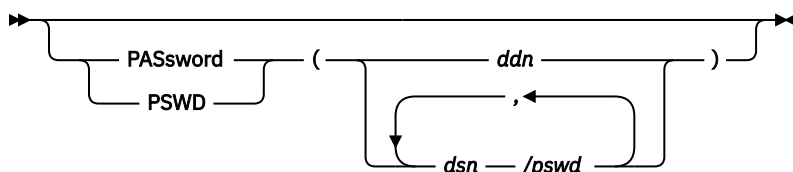
ddn

Specifies the name of the DD statement that identifies the (output) print data set. Each of the DD statements corresponding to a DDNAME (*ddn*) must identify only one volume serial number. If this keyword is not specified, the default is SYSPRINT.

Note:

1. If the DCB keyword LRECL is specified on the DD statement, it must be in the range of 84 to 137 inclusive. If BLKSIZE is specified, it must be at least four greater than the LRECL.
2. If an LRECL less than 84 is chosen, the return code is 8 and an error message is issued.
3. If the specified LRECL is greater than 137, LRECL and BLKSIZE are set to 137 and 141, respectively.

PASSWORD



PASSWORD specifies the passwords DFSMSdss is to use for password-protected data sets. (Password checking is bypassed for RACF-protected data sets.) This is required only if:

- You do not have the required volume-level RACF DASDVOL or RACF DATASET access.

- The installation authorization exit does not bypass the checks.
- You do not want to be prompted for the password for VSAM data sets.

Note: You should specify the passwords for all data sets that do not have RACF protection but do have password protection. During processing, a utility invoked by DFSMSdss may have to prompt the operator for a password. You can control authorization checking by using the installation authorization exit.

For VSAM data sets, passwords are checked at the cluster level only.

Note: The PASSWORD keyword is not valid for a PRINT VTOC command.

ddn

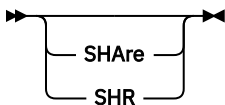
Specifies the name of the DD statement that identifies the sequential data set, or member of a partitioned data set, that contains data set names and their passwords. This data set must contain card-image records in DFSMSdss command syntax format.

dsn/pswd

dsn is a fully qualified data set name. *pswd* is its password. If no password follows the slash (/), *dsn* is treated as though it were *ddn*.

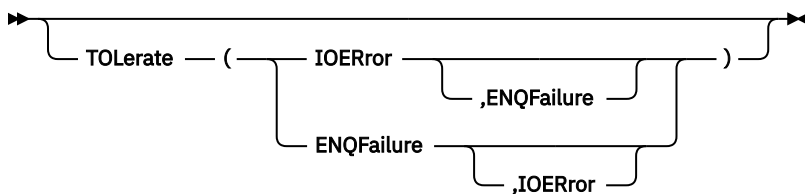
Printing of actual data set passwords specified in your input command stream is suppressed in the SYSPRINT output.

SHARE



SHARE specifies that DFSMSdss is to share, for read access with other programs, the data set that is to be printed.

TOLERATE



ENQFailure

Specifies that data sets are to be processed even though shared or exclusive access fails.

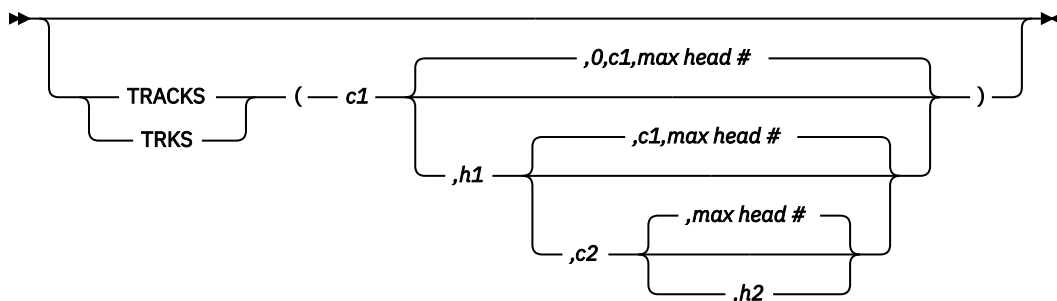
TOLERATE(ENQFAILURE) is ignored if it is specified in a PRINT TRACKS or PRINT VTOC operation.

IOError

Specifies that DFSMSdss is to continue processing even though I/O errors occur, but is to end after 100 errors.

Related reading: For additional information about TOL(ENQF), see Chapter 24, “Examples of the application program with the user interaction module (UIM),” on page 627.

TRACKS



TRACKS specifies ranges of tracks to be printed.

c1,h1

Specifies the cylinder and head number of the beginning of the range. Specify hexadecimal numbers as X'c1' or X'h1'.

c2,h2

Specifies the cylinder and head number of the end of the range. Specify hexadecimal numbers as X'c2' or X'h2'. The c2 must be greater than or equal to c1. If c2 equals c1, h2 must be greater than or equal to h1.

You can enter X'FFFFFFF' (or 268435455) as the high cylinder value. This causes DFSMSdss to choose the high cylinder value to be the end of the volume.

DFSMSdss verifies that the range is within the limits of the device. If you do not specify all four values for a range, DFSMSdss provides the missing values unless the omitted value causes a syntax error. No intervening values can be omitted. For example:

Specified Results

None

Syntax error

c1

c1,0,c1, maximum head number

c1,h1

c1,h1,c1, maximum head number

c1,h1,c2

c1,h1,c2, maximum head number

c1,c2

Syntax error

,h1

Syntax error

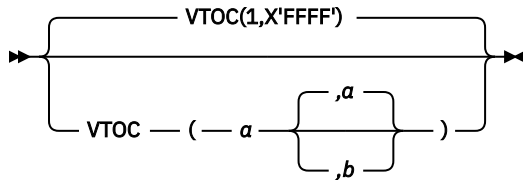
c1,h1,X'FFFFFFF'

c1,h1,maximum cylinder number for the volume, maximum head number

Note:

NOTE: Any reference to a track or range of tracks that reside in the EAS of an EAV will result in the entire 21-cylinder area being printed.

VTOC



VTOC specifies that all or part of the VTOC is to be printed. Omitting the VTOC, DATASET, and TRACKS keywords causes the entire VTOC to be printed. Part of the VTOC can be printed by specifying:

```
VTOC(a,b)
```

where *a* and *b* are the relative track numbers (1 is the first track) of the first and last tracks to be printed. The value of *b* must be equal to or greater than the value of *a* and equal to or less than 65535 (X'FFFF'). Either of these numbers can be specified in decimal or hexadecimal. To specify a hexadecimal number, code X'nn'.

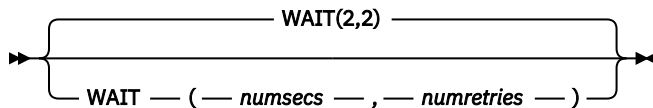
If only the first value (*a*) is specified, only that track is printed.

If the second value (*b*) is greater than the relative track number of the last physical track of the VTOC, DFSMSdss prints up to and including the last track of the VTOC. Therefore, another way of printing the entire VTOC would be to specify:

```
VTOC(1,X'FFFF')
```

Note: The PASSWORD keyword is not valid for a PRINT VTOC command.

WAIT



WAIT specifies to DFSMSdss the length of a wait in seconds and the number of retries to obtain control of a data set.

numsecs

Specifies a decimal number from 1 to 255 that designates the interval, in seconds, between retries.

numretries

Specifies a decimal number from 0 to 99 that designates the number of retries to gain control of a data set.

The default for *numsecs,numretries* is (2,2), which specifies two retries at 2-second intervals. If you do not want to wait for a data set, specify 0 for either *numsecs* or *numretries*.

Note: The WAIT keyword does not control wait/retry attempts for system resources (such as the VTOC and the VVDS). For system resources, the default wait time is 3 seconds and the default retry count is 30. This results in a total wait time of 90 seconds.

Related reading: For more information about controlling the wait/retry attempts for system resources, see the [“Controlling the wait/retry time for serialization of system resources \(PN11523\)”](#) on page 215.

Examples of print operations

The following are examples of the PRINT command.

Example 1: printing a range of tracks

```
//JOB2      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD      DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//SYSIN     DD       *
PRINT TRKS(1,0,1,5) INDDNAME(DASD)
/*
```

Prints a hard copy of tracks 0 through 5 from cylinder 1 on volume 111111.

```
//JOB2      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//DASD      DD       UNIT=3390,VOL=(PRIVATE,SER=222222),DISP=OLD
//SYSIN     DD       *
PRINT TRKS(X'1AFF',X'0',X'1AFF',X'E') INDDNAME(DASD)
/*
```

Prints a hard copy of tracks 0-14 from cylinder 6911 (x'1AFF') specified in hexadecimal on volume 222222.

Example 2: printing a component of a Virtual Storage Access Method (VSAM) cluster

```
//JOB3      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//SYSIN     DD       *
PRINT INDYNAM(338000) /* ALLOC VOL 338000 DYNAMICALLY */ -
DS(PARTS.VSAM1.INDEX) /* DATA SET THAT HAS BAD TRACK */ -
WAIT(0,0)             /* DO NOT WAIT IF ENQ FAILS */ -
TOL(ENQF)             /* IGNORE ENQ FAILURES */ -
PSWD(PARTS.VSAM1/USERPSWD) /* PASSWORD FOR CLUSTER */
/*
```

A component of a VSAM data set is to be printed. The printing proceeds even if the component cannot be serialized (enqueued).

Example 3: printing a data set

```
//STEPT003 EXEC PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=*
//SYSIN     DD       *
PRINT DATASET(PUBSEXMP.SAM.S01) INDYNAM(D9S060)
/*
```

The output shown in [Figure 21 on page 441](#) was produced from the example above.

```

PAGE 0001      5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES      1999.211 14:55
PRINT
  DATASET(PUBSEXP.SAM.S01) -
  INDYNAM(D9S060)
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'PRINT '
ADR109I (R/I)-RI01 (01), 1999.211 14:55:44 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 14:55:44 EXECUTION BEGINS
*** TRACK(CCHH) 0000000E R0 DATA 0000000E00000000
COUNT 0000000E01000190
0000 F0F0F0F0 F0F1D7E4 C2E2C5E7 D4D74BE2 C1D44BE2 F0F1C1C2 C3C4C5C6 C7C8C9D1
*000001PUBSEXP.SAM.S01ABCDEFHIJ*
0020 D2D3D4D5 D6D7D8D9 E2E3E4E5 E6E7E8E9 C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7
*KLMNOPQRSTUVWXYZABCDEFGHIJKLMNOP*
0040 D8D9E2E3 E4E5E6E7 E8E9C1C2 C3C4C5C6 F0F0F0F0 F0F2D7E4 C2E2C5E7 D4D74BE2
*QRSTUVWXYZABCDEFGHI000002PUBSEXP.S*
0060 C1D44BE2 F0F1C1C2 C3C4C5C6 C7C8C9D1 D2D3D4D5 D6D7D8D9 E2E3E4E5 E6E7E8E9
*AM.S01ABCDEFHIJKLMNOPQRSTUVWXYZ*
0080 C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E9C1C2 C3C4C5C6
*ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEF*
00A0 F0F0F0F0 F0F3D7E4 C2E2C5E7 D4D74BE2 C1D44BE2 F0F1C1C2 C3C4C5C6 C7C8C9D1
*000003PUBSEXP.SAM.S01ABCDEFHIJ*
00C0 D2D3D4D5 D6D7D8D9 E2E3E4E5 E6E7E8E9 C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7
*KLMNOPQRSTUVWXYZABCDEFGHIJKLMNOP*
00E0 D8D9E2E3 E4E5E6E7 E8E9C1C2 C3C4C5C6 F0F0F0F0 F0F4D7E4 C2E2C5E7 D4D74BE2
*QRSTUVWXYZABCDEFGHI000004PUBSEXP.S*
0100 C1D44BE2 F0F1C1C2 C3C4C5C6 C7C8C9D1 D2D3D4D5 D6D7D8D9 E2E3E4E5 E6E7E8E9
*AM.S01ABCDEFHIJKLMNOPQRSTUVWXYZ*
0120 C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E9C1C2 C3C4C5C6
*ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEF*
0140 F0F0F0F0 F0F5D7E4 C2E2C5E7 D4D74BE2 C1D44BE2 F0F1C1C2 C3C4C5C6 C7C8C9D1
*000005PUBSEXP.SAM.S01ABCDEFHIJ*
0160 D2D3D4D5 D6D7D8D9 E2E3E4E5 E6E7E8E9 C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7
*KLMNOPQRSTUVWXYZABCDEFGHIJKLMNOP*
0180 D8D9E2E3 E4E5E6E7 E8E9C1C2 C3C4C5C6
*QRSTUVWXYZABCDEF*
ADR006I (001)-STEND(02), 1999.211 14:55:44 EXECUTION ENDS
ADR013I (001)-CLTSK(01), 1999.211 14:55:44 TASK COMPLETED WITH RETURN CODE 0000
ADR012I (SCH)-DSSU (01), 1999.211 14:55:44 DFSMSDSS PROCESSING COMPLETE. HIGHEST
RETURN CODE IS 0000

```

Figure 21. Output Resulting from a PRINT Command

RELEASE command for DFSMSdss

The RELEASE command releases allocated but unused space from all eligible sequential, partitioned, and extended-format VSAM data sets that pass INCLUDE, EXCLUDE, and BY filtering criteria. The RELEASE command does not release space from guaranteed-space VSAM extended-format data sets. DFSMSdss only selects data sets that have space that can be released.

DFSMSdss offers two ways to process RELEASE commands:

- **Logical processing** operates on a single selected data set at a time. The data set can be multivolume. The user has the option to not specify volumes or to specify one or more volumes with the LOGDDNAME, LOGDYNAM, or STORGRP keywords. Releasing unused space from extended-format VSAM data sets requires logical processing.

If no input volumes are specified and the INCAT keyword is not specified, DFSMSdss selects from all data sets cataloged in the catalogs accessible through the standard search order. For INCAT keyword details, see the INCAT keyword description in this section.

- **Physical processing** operates on all selected data sets that reside on a single volume. The user must specify one or more volumes with the DDNAME or DYNAM keywords. You cannot use physical processing to release unused space from extended-format VSAM data sets.

To process multivolume data sets, do one of the following:

- Specify no input volumes.
- Specify LOGDDNAME, LOGDYNAM, or STORGRP with an appropriate SELECTMULTI option or a volume list.

- Specify DDNAME or DYNAM with the volume or volumes on which the data set has space that can be released.

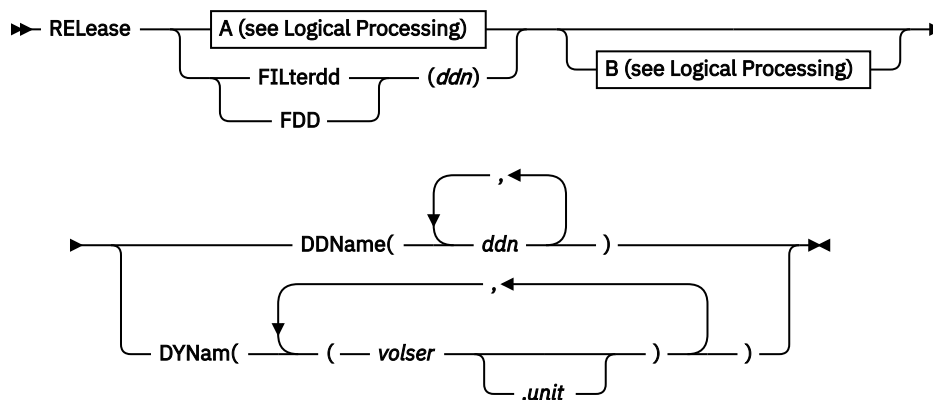
You must exclude, by using the EXCLUDE keyword, data sets whose last used block pointer in the data set's VTOC entry are not properly maintained. This can occur if you use an access method other than BSAM, QSAM, or BPAM. DFSMSdss does not release any space for data sets that are empty (the last used block pointer in the data set's VTOC entry is zero). This restriction does not apply to PDSE data sets; used space in PDSE data sets is maintained internally, rather than by reference to the data set's VTOC entry.

The following apply to the RELEASE command:

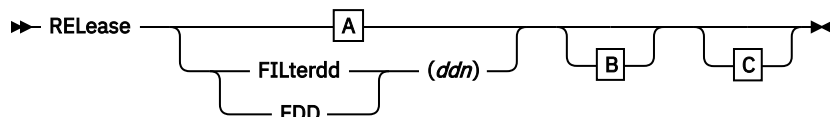
- A data set with the maximum number of extents already in use does not have any space released. For an extended-format VSAM data set, the maximum is 255 extents. For a partitioned data set extended (PDSE) and an extended-format sequential data set, the maximum is 123 extents. For other partitioned and sequential data sets, the maximum is 16 extents.
- For extended-format VSAM data sets, DFSMSdss releases space from only the data component of a base cluster or an alternate index (AIX).
- For striped VSAM data sets, DFSMSdss releases eligible space from each stripe.
- Free tracks in cylinder-allocated extents are not released. Only free cylinders are released.
- DFSMSdss excludes system data sets beginning with SYS1, unless the PROCESS keyword is used.
- DFSMSdss logical processing releases space from each volume on which an extended-format sequential data set contains data.
- DFSMSdss does not support the release of HFS data sets, as the DADSM PARTREL macro no longer supports HFS data sets.
- DFSMSdss does not support the release of zFS data sets.
- Unopened data set sets (DS1LSTAR = 0) are not processed.

For additional information about filtering, see [Chapter 16, “DFSMSdss filtering—choosing the data sets you want processed,”](#) on page 255.

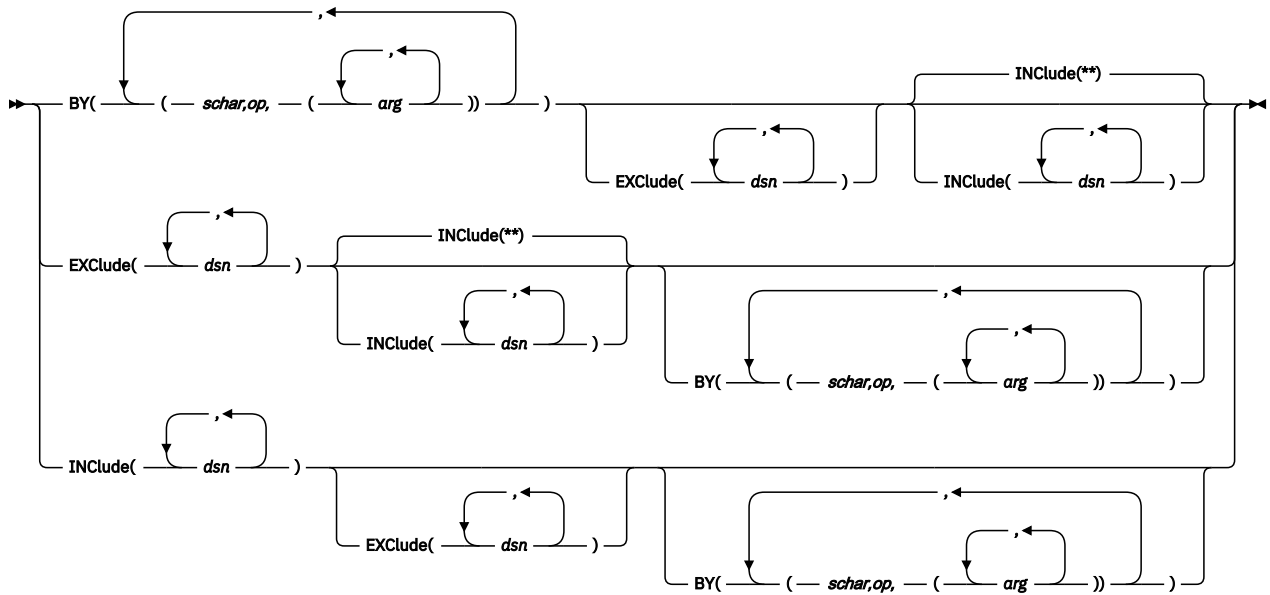
RELEASE syntax for physical processing



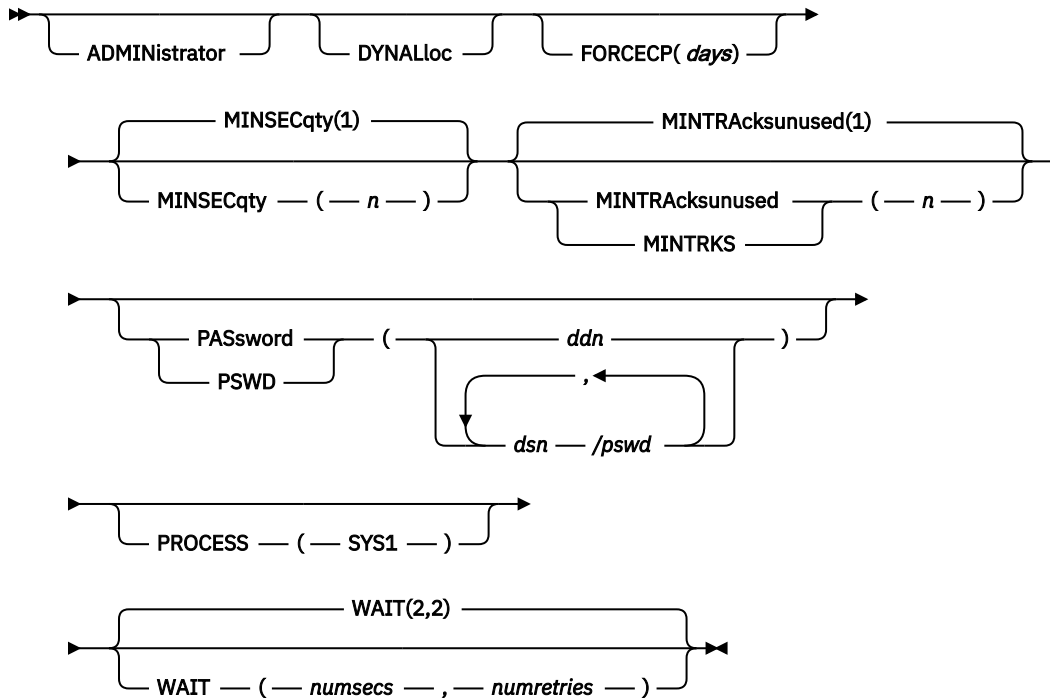
RELEASE syntax for logical processing



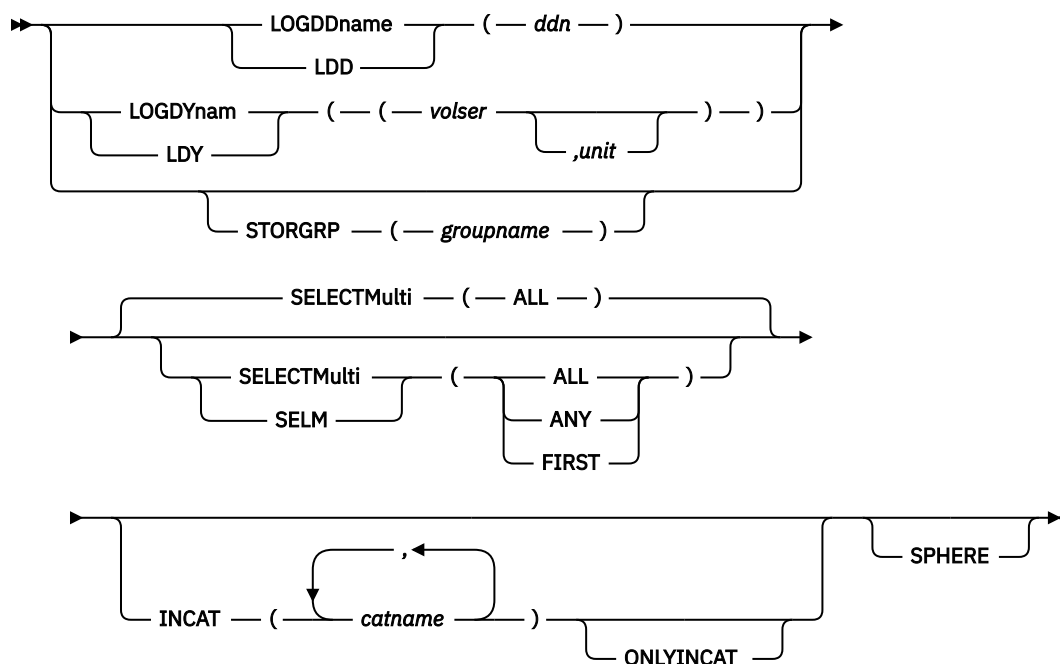
A: Additional Keywords for Physical or Logical Processing



B: Optional Keywords for Physical or Logical Processing



C: Optional Keywords with RELEASE for Logical Processing



Explanation of RELEASE command keywords

This section describes the keywords for the RELEASE command.

ADMINISTRATOR



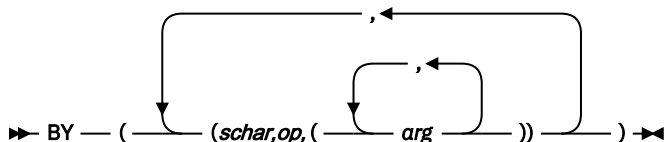
ADMINISTRATOR allows you to act as a DFSMSdss-authorized storage administrator for the RELEASE command. DFSMSdss-initiated access checking to data sets and catalogs is bypassed. If you are not authorized to use the ADMINISTRATOR keyword, the command ends with an error message.

To use the ADMINISTRATOR keyword, all of the following conditions must be true:

- FACILITY class is active.
- Applicable FACILITY-class profile is defined.
- You have READ access to that profile.

For additional information about using the ADMINISTRATOR keyword, see [“ADMINISTRATOR keyword”](#) on page 542.

BY

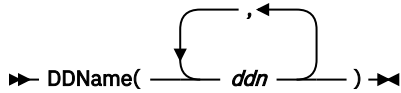


BY specifies that the data sets selected up to this point, by the processing of the INCLUDE and EXCLUDE keywords, are to be further filtered. To select the data set, *all* BY criteria must be met. See the separate discussions of INCLUDE and EXCLUDE for information on how these keywords are specified.

Note: You must use FILTERDD when you have more than 255 entries in INCLUDE, EXCLUDE, or BY list keywords.

For additional information about the BY keyword, see [“INCLUDE” on page 446](#) and [“EXCLUDE” on page 445](#).

DDNAME



DDNAME requests physical processing.

ddn

Specifies the name of the DD statement that identifies a volume whose sequential and partitioned data sets, if selected, are to have their unused space released. To assure correct processing, each of the DD statements corresponding to a DDname (*ddn*) must identify only one volume serial number.

DYNALLOC

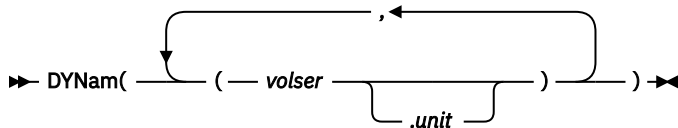


DYNALLOC specifies dynamic allocation, instead of the ENQ macro, to serialize the use of data sets. This allows cross-system serialization in a JES3/MVS environment.

Consider:

- This serialization is of value only when the dynamic allocation/JES3 interface is not disabled.
- Run time increases when you use DYNALLOC to serialize data sets (as opposed to enqueue) because overhead is involved in dynamic allocation and serialization across multiple processors.

DYNAM



DYNAM requests physical processing and specifies that the volume to be processed be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*). Using DYNAM instead of DD statements to allocate DASD volumes will not appreciably increase run time and permits easier coding of JCL and command input.

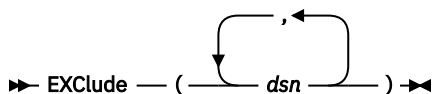
volser

Specifies the volume serial number of a DASD volume to be processed.

unit

Specifies the device type of a DASD volume to be processed. This parameter is optional.

EXCLUDE



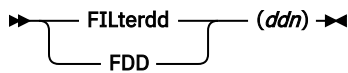
dsn

Specifies the name of a data set to be excluded from the data sets selected by INCLUDE. Either a fully or a partially qualified data set name can be used. See the separate discussions of INCLUDE and BY for information on how these keywords are specified.

Note: You must use FILTERDD when you have more than 255 entries in INCLUDE, EXCLUDE, or BY list keywords.

For more information, see [“BY” on page 444](#) and [“INCLUDE” on page 446](#).

FILTERDD

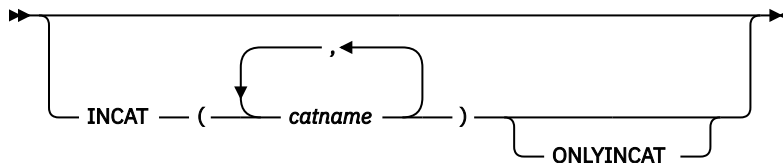


ddn

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 use. This is in the form of card-image records, in DFSMSdss command syntax, that contain the INCLUDE, EXCLUDE, and BY keywords that complete the RELEASE command syntax.

Note: You must use FILTERDD when you have more than 255 entries in the INCLUDE, EXCLUDE, or BY list of subkeywords.

INCAT



INCAT(*catname*) specifies that DFSMSdss search the user catalogs specified by the INCAT(*catname*) keyword, then follow the standard search order to locate data sets. INCAT(*catname*) allows you to identify specific source catalogs. To specify INCAT, RACF authorization might be required.

catname

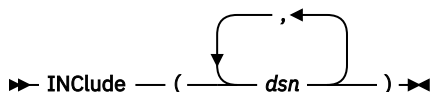
Specifies a fully qualified catalog name.

ONLYINCAT

Specifies that DFSMSdss only searches catalogs that are specified in the INCAT catalog name list.

DFSMSdss does not process an SMS-managed data set that is cataloged outside the standard search order. This is the case even if the data set is cataloged in one of the catalogs that is specified with the INCAT(*catname*) keyword. Ensure that the SMS-managed data sets are cataloged under standard catalog search order.

INCLUDE



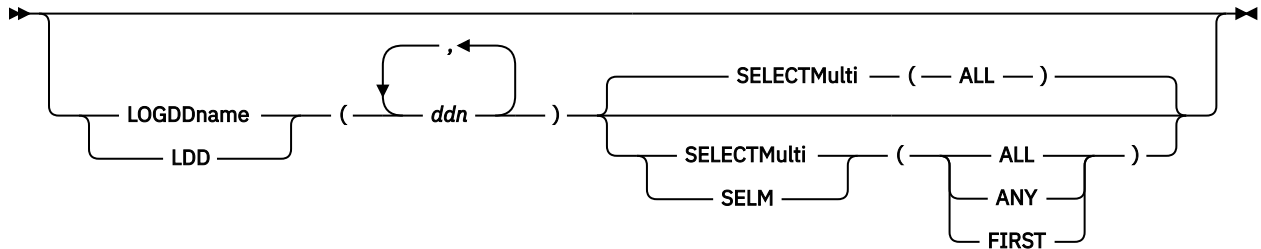
dsn

Specifies the name of a data set whose unused space is eligible to be released. Either a fully or a partially qualified data set name can be used. See [“Filtering by data set names” on page 256](#). If INCLUDE is omitted (but EXCLUDE or BY is specified) or if INCLUDE(**) is specified, *all* data sets are eligible to be selected for releasing.

Note: You must use FILTERDD when you have more than 255 entries in INCLUDE, EXCLUDE, or BY list keywords.

For more information, see [“BY” on page 444](#) and [“EXCLUDE” on page 445](#).

LOGDDNAME



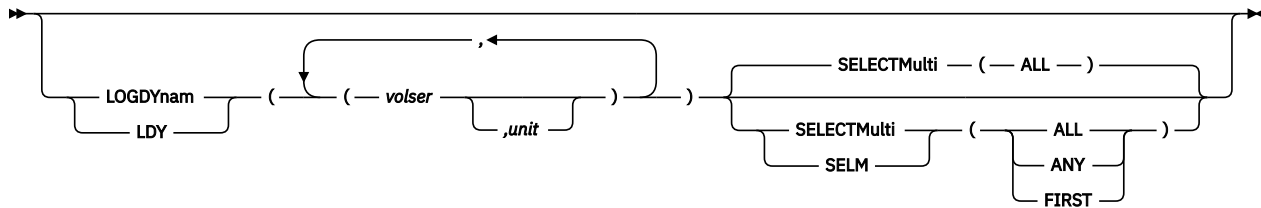
LOGDDName specifies logical processing that is based on a designated volume list.

ddn

Specifies the name of the DD statement that identifies a single volume that contains the data sets to be released by logical processing. You can specify up to 255 DDNAME entries with the LOGDDNAME keyword. Each DD statement can correspond to only one volume serial number.

For more information see [“SELECTMULTI”](#) on page 449 and [“LOGDYNAM”](#) on page 447.

LOGDYNAM



LOGDYNAM requests logical processing and specifies that the volumes that contain the data sets to be processed are dynamically allocated.

volser

Specifies the volume serial number of a DASD volume to be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number by using an asterisk(*). You can specify up to 511 volumes with the LOGDYNAM keyword.

unit

Specifies the device type of a DASD volume to be processed. This parameter is optional.

SELECTMULTI

Specifies how DFSMSdss selects cataloged multivolume data sets. DFSMSdss accepts SELECTMULTI only when you specify logical processing with the LOGDDNAME, LOGDYNAM, or STORGRP keyword. Otherwise, DFSMSdss cannot accept the specification of SELECTMULTI.

ALL

Specifies that DFSMSdss *not process* a multivolume data set unless the following criteria is met:

- The volume list created by the LOGINDDDNAME, LOGINDYNAM, or STORGRP keyword must list all the volumes that contain a part of the data set.
- The volume list created by the LOGINDDDNAME, LOGINDYNAM, or STORGRP keyword must list all the volumes that contain a part of the VSAM cluster.

ALL is the default.

ANY

Specifies that DFSMSdss process a multivolume data set when the following criteria are met:

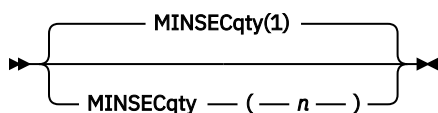
- Any volume specified in the volume list that is created by the LOGINDDDNAME, LOGINDYNAM, or STORGRP keyword must contain a part of the data set.
- Any volume specified in the volume list that is created by the LOGINDDDNAME, LOGINDYNAM, or STORGRP keyword must contain a part of the VSAM cluster.

FIRST

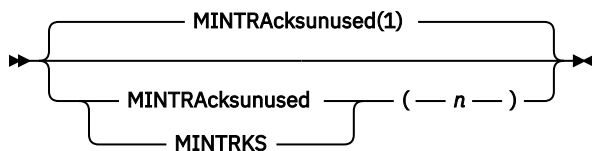
Specifies that DFSMSdss process a multivolume data set only when the volume list includes the volume that contains the first part of the data set. The volume list is created by the LOGINDDDNAME, LOGINDYNAM, or STORGRP keyword. For VSAM data sets, the volume list must include the volume that contains the first extent of the data component for the cluster.

Notes for LOGDDNAME, LOGDYNAM and STORGRP keywords:

1. If the LOGDDNAME, LOGDYNAM, or STORGRP keyword is not specified, DFSMSdss selects from all data sets cataloged in the catalogs accessible through the standard search order.
2. If the LOGDDNAME, LOGDYNAM, or STORGRP keyword is specified, DFSMSdss still uses the standard catalog search order, but it selects data sets only from the specified volumes.
3. A multivolume data set that has extents on volumes is processed when you specify the SELECTMULTI keyword if the following criteria are met:
 - The volumes cannot be designated on the volume list created by the LOGDDNAME, LOGDYNAM, or STORGRP keyword.
 - You must specify the SELECTMULTI keyword option (ANY or FIRST) to release a multivolume data set that has extents on volumes which are not identified with the LOGDDNAME, LOGDYNAM, or STORGRP keyword.

MINSECQTY

Because DFSMSdss releases all the allocated but unused space when you have not specified a secondary allocation quantity, you will not be able to add records to the data set after the release operation. MINSECQTY solves this problem. The release operation is not performed unless the secondary allocation quantity in the VTOC (VVDS for extended-format VSAM) is equal to or greater than *n* tracks and the data set has not reached the maximum number of used extents (16 extents for sequential and partitioned data sets, 123 extents for extended-format sequential data sets and PDSEs, or 255 extents for extended-format VSAM data sets). The letter *n* represents a 1-to-8-digit decimal number with a range from zero to 99999999. The default is one track, if you do not specify MINSECQTY (*n*).

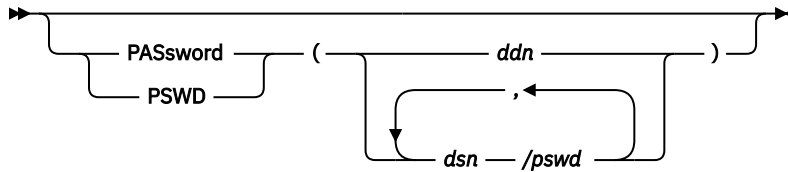
MINTRACKSUNUSED

MINTRACKSUNUSED specifies that the release operation is to be performed only if the number of unused tracks for a selected data set is equal to or greater than *n* (*n* is a 1-to-8-digit decimal number with a range from 0 to 99999999). When you do not specify MINTRKS, DFSMSdss uses a default value of 1. All unused tracks will be released if you perform a release operation.

ONLYINCAT

See [“INCAT” on page 340](#).

PASSWORD



PASSWORD specifies the passwords DFSMSdss uses for selected password-protected sequential and partitioned data sets. (Password checking is bypassed for RACF-protected data sets.) This keyword is required only if:

- You do not have the required volume-level RACF DASDVOL or RACF DATASET access.
- The installation authorization exit does not bypass the checks.

Note: You should specify the passwords for all data sets that do not have RACF protection but do have password protection. During processing, a utility invoked by DFSMSdss may have to prompt the operator for a password. You can control authorization checking by using the installation authorization exit.

ddn

Specifies the name of the DD statement that identifies the sequential data set or member of a partitioned data set that contains data set names and their passwords. This data set must contain card-image records in DFSMSdss command syntax format.

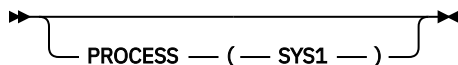
dsn/pswd

dsn is a fully qualified data set name. *pswd* is its password. If no password follows the slash (/), *dsn* is treated as though it were *ddn*.

Printing of actual data set passwords specified in your input command stream is suppressed in the SYSPRINT output.

For additional information about the installation authorization exit, see [z/OS DFSMS Installation Exits](#).

PROCESS



PROCESS specifies that the release operation is to be performed for data sets with a high-level qualifier of SYS1. To specify PROCESS(SYS1), RACF authorization may be required.

For additional information about RACF authorization, see the [Chapter 5, “Protecting DFSMSdss functions,” on page 35](#).

SELECTMULTI

See [“LOGDDNAME” on page 447](#), [“LOGDYNAM” on page 447](#), and [“STORGRP” on page 450](#).

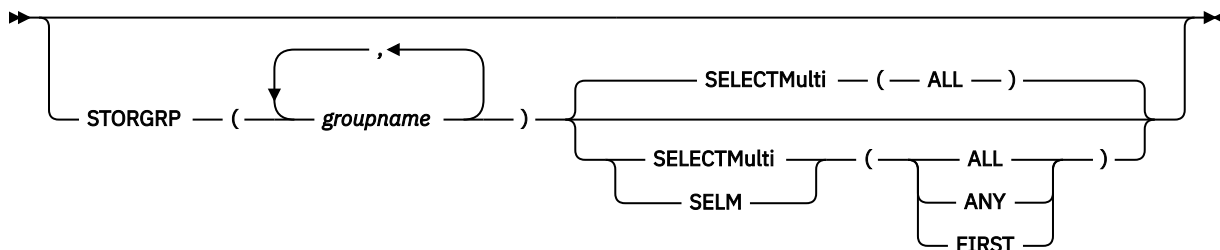
SPHERE



SPHERE specifies that DFSMSdss also select all associated alternate index clusters whenever you select a VSAM base cluster. You do not need to specify individual names of sphere components, only the base cluster name. If you specify a volume list (with LOGDDNAME or LOGDYNAM), you do not need to specify the volumes on which the AIX clusters reside.

To select an entire sphere, specify the base cluster name by using fully or partially qualified data set names. If you specify SPHERE but not the base cluster name, DFSMSdss processes only those data components of the sphere whose names are specified.

STORGRP



STORGRP specifies that all of the online volumes in the storage group be dynamically allocated. If a volume in the storage group is not online, that volume is not used for processing. You can specify up to 255 storage group names. Specifying STORGRP with a storage group name is equivalent to specifying LOGDYNAM or LOGDDNAME with all the online volumes in the storage group included in the list.

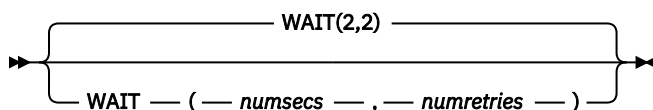
You can specify the STORGRP keyword with the SELECTMULTI keyword, but STORGRP cannot be specified at the same time with the DDNAME, DYNAM, LOGDDNAME, or LOGDYNAM keywords.

Notes for LOGDDNAME, LOGDYNAM, and STORGRP keywords:

1. DFSMSdss selects from all data sets that are cataloged in the catalogs that are accessible through the standard search order if you do not specify the LOGINDDDNAME, LOGINDYNAM, or STORGRP keyword.
2. When you specify the LOGDDNAME, LOGDYNAM, or STORGRP keyword, DFSMSdss still uses the standard catalog search order. However, DFSMSdss selects data sets only from the specified volumes.
3. You can process a multivolume data set that has extents on volumes when you specify the SELECTMULTI keyword if the following criteria are present:
 - The volumes cannot be designated on the volume list that is created by the LOGDDNAME, LOGDYNAM, or STORGRP keyword.
 - You must specify the SELECTMULTI keyword option (ANY or FIRST) to release a multivolume data set that has extents on volumes which are not identified with the LOGDDNAME, LOGDYNAM, or STORGRP keyword.

For more information about the SELECTMULTI keyword, see [“LOGDYNAM” on page 447](#).

WAIT



WAIT specifies to DFSMSdss the length of a wait in seconds and the number of retries to obtain control of a data set.

numsecs

Specifies a decimal number from 0 to 255 that designates the interval, in seconds, between retries.

numretries

Specifies a decimal number from 0 to 99 that designates the number of retries to gain control of a data set.

The default for *numsecs,numretries* is (2,2), which specifies two retries at 2-second intervals. If you do not want to wait for a resource, specify 0 for either numsecs or numretries.

Note: The WAIT keyword does not control wait/retry attempts for system resources (such as the VTOC and the VVDS). For system resources, the default wait time is 3 seconds and the default retry count is 30. This results in a total wait time of 90 seconds.

For information about controlling the wait/retry attempts for system resources, see the [“Controlling the wait/retry time for serialization of system resources \(PN11523\)”](#) on page 215.

Example of a release operation

The following is an example of a release operation on selected sequential and partitioned data sets.

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//SYSIN     DD       *
RELEASE INCLUDE(**) -
  DYNAM(338000)      /* DYNAM ALLOC VOL 338000          */ -
  MINTRKS(10)        /* THERE ARE 10 OR MORE UNUSED TRKS  */ -
/* MINSEC NOT SPEC. IT DEFAULTS TO 1 */
/*
```

Unused tracks of sequential and partitioned data sets on volume 338000 are to be released if both:

- The number of unused tracks in the data set is greater than or equal to 10.
- The data set can be extended later if required (MINSEC(1)). This need not be specified, because it is the default.

The above example can be modified as follows to release unused tracks of all sequential and partitioned data sets, other than system data sets, that have any unused tracks and that can be extended:

```
//SYSIN     DD       *
RELEASE INCLUDE(**) -
  DYNAM(338000)      /* DYNAM ALLOC VOL 338000 */
/*
```

The following is an example of the commands used to release unused space from extended-format VSAM data sets residing wholly or in part on a specific volume, and to release unused space from the data sets' alternate indexes that reside on any volumes:

```
//SYSIN     DD       *
RELEASE INCLUDE(**) -
  BY(DSORG EQ VSAM)  /* RELEASE ONLY VSAM      */ -
  LOGDYNAM(339000)   /* DYN ALLOC VOL 339000   */ -
  SELECTMULTI(ANY)   /* EVEN IF MULTIVOL      */ -
  SPHERE             /* RELEASE AIXES          */
/*
```

To release unused space from all eligible data sets that are cataloged in a particular user catalog, without specifying any volume:

```
//SYSIN     DD       *
RELEASE INCLUDE(**) -
  INCAT(CATALOGA) ONLYINCAT /* ONLY FROM CATALOG */
/*
```

RESTORE command for DFSMSdss

With the RESTORE command, you can restore data to DASD volumes from DFSMSdss-produced dump volumes. You can restore data sets, an entire volume, or ranges of tracks. You can restore to unlike devices from a logical dump tape.

The FULL keyword with the RESTORE command restores an entire DASD volume. The TRACKS keyword for the RESTORE command restores a range of tracks.

DFSMSdss offers several ways to process RESTORE commands:

- *Logical processing* is data set-oriented, which means it operates against data sets independently of physical device format.
- *Physical processing* can operate against data sets, volumes, and tracks, but is oriented toward moving data at the track-image level.
- *File processing* is z/OS UNIX-oriented and can process individual UNIX files that are specified as absolute paths.

The processing method is determined by what type of dump tape is used as input and by the keywords specified on the command.

Target data set allocation differs between a physical data set and logical data set restore of non-VSAM data sets. Logical data set restore allocates target data sets according to the amount of used space in the source data set, thereby freeing unused space. Physical data set restore preserves the original size of the source data set. To force unused space to be kept during logical data set restore, the ALLDATA or ALLEXCP keyword must be specified during a dump. However, the action that restore takes with respect to these keywords depends on data set characteristics and device characteristics. If you require that all of the unused space is restored, then you should be sure 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.

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

Extended-physical-sequential data sets, VSAM extended-format data sets, and SAM compressed extended function data sets cannot be restored to non-SMS-managed target volumes during a physical or logical data set restore. Data sets with DFM attributes can be restored to non-SMS-managed target volumes but the DFM attributes will be lost and a warning message will be issued.

DFSMSdss RESTORE will always preserve data set encryption attributes of the source data set. Therefore, new allocations will be defined with the source encryption attribute. If a preallocated target data set is encountered, it must also be encrypted to be considered a usable target data set. The usable preallocated target will be overwritten with the source encryption attributes (which includes the key label).

For more information about using the RESTORE command, see [Chapter 6, “Managing availability with DFSMSdss,”](#) on page 39.

Special considerations for RESTORE

The following special considerations apply when you perform a restore operation:

- When restoring a partitioned data set that has a secondary allocation of zero, the amount of unused space that is allocated to the target data set might be different from that of the source data set if the ALLDATA and ALLEXCP keywords were not specified for the dump.
- When restoring a VSAM data set with no secondary allocation, a secondary allocation can be added as follows:
 - When the primary allocation is less than one cylinder, a secondary allocation equal to the primary is created.
 - When the primary allocation is greater than or equal to one cylinder, a secondary allocation is created by computing one percent of the primary and rounding up to the next cylinder (in tracks).

The index component can also have secondary space added if it has no secondary allocation.

- The RESTORE FULL or RESTORE TRACK commands might invoke ICKDSF to rebuild the VTOC INDEX data set for a target volume. Therefore, users of these commands require the appropriate authority for ICKDSF.
- When performing a logical or physical data set restore operation of a VSAM extended-format data set, the target data set allocation must be consistent with the source data set allocation as follows:
 - If the source is an extended-format VSAM, then the target must be an extended-format VSAM.
 - If the source is a compressed VSAM KSDS, then the target must be a compressed 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.
- The target control interval size must be equal to the source.
- When performing a logical or physical data set restore operation of a non-VSAM extended-format data set, the target data set allocation must be consistent with the source data set allocation as follows:
 - If the source is an extended-format non-VSAM data set, then the target must be an extended-format non-VSAM data set.
 - If the source is a compressed non-VSAM data set, then the target must be a compressed non-VSAM data set.
 - If the source is an encrypted non-VSAM data set, then the target must be an encrypted non-VSAM data set.
 - If the source is a non-extended-format non-VSAM data set, then the target must be restored to a non-extended-format non-VSAM data set.
- The actions DFSMSdss performs for RENAME, RENAMEUNCONDITIONAL, REPLACE, or REPLACEUNCONDITIONAL depend on the keywords you specify and the configurations of data sets on volumes. This section contains figures that describe specific environments for DFSMSdss restore operations.
- If you restore a data set that has an F8/F9 DSCB pair on the volume from which it is dumped to a volume that does not support F8/F9 DSCBs, the attributes in the F9 DSCB are lost. To retain these extended attributes, the target volumes of the RESTORE must support F8/F9 DSCBs.

Data integrity considerations for full or tracks restore operations

For a full or tracks restore operation, DFSMSdss serializes the VTOC to prevent DADSM functions such as ALLOCATE, EXTEND, RENAME, and SCRATCH from changing the contents of the VTOC on the volume during the restore operation. Data sets are not serialized on these restore operations and thus, some data sets can be opened by other jobs during the restore operations. The result might be that partially updated data sets are restored. Full data integrity can always be guaranteed by performing restore operations by data set only when TOLERATE(ENQFAILURE) or SHARE is not specified.

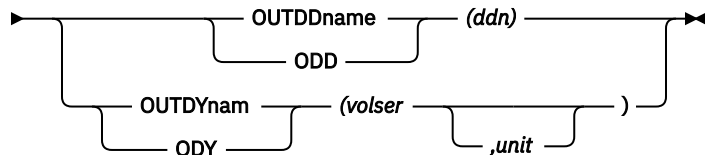
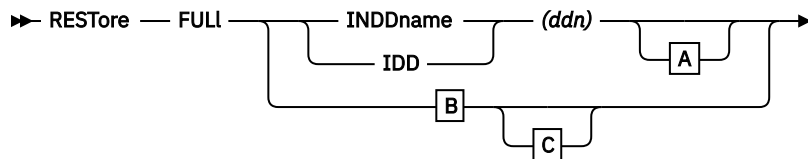
During full or tracks restore operations, and stand-alone restore, it is possible to duplicate a volume serial number in the sysplex. If there is data in the coupling facility for the duplicated volume serial number, data integrity can be compromised. The recommended procedure before restoring a suspected volume is as follows:

1. Use the D SMS,CFVOL(volid) command to determine if there is any data in the coupling facility caches for the volume serial number to be restored.
2. Determine the disposition of any data that is in the caches.
3. Do not restore the suspected volume until the duplicated volume can be varied offline to the sysplex, thus ensuring that there is no data in any coupling facility cache for the duplicated volume.

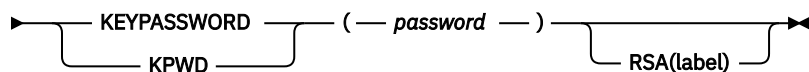
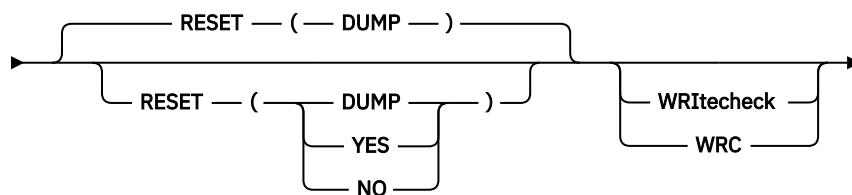
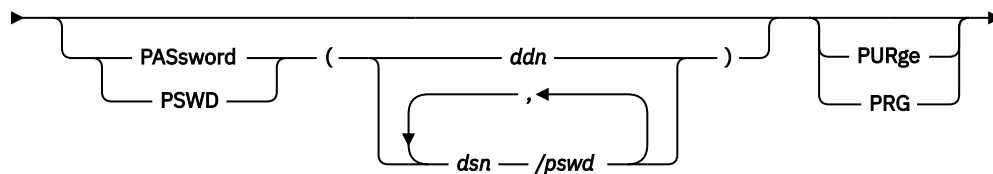
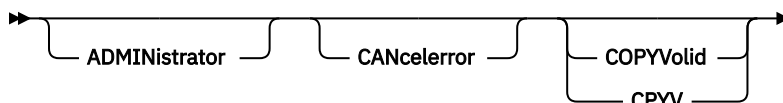
During full or tracks restore operations, DFSMSdss resets the data-set-changed indicator for each data set being restored. Because the data set has not changed since the backup was performed, DFSMSdss resets the data-set-changed indicator.

Attention: Use caution when using DFSMSdss Full Volume Restore in an environment in which other data set applications or backup applications are used. If the applications rely on the data-set-changed indicator to mean that the data set has not changed since earlier processing, a data integrity issue can result when using DFSMSdss to Restore full volumes when other applications are tracking the data sets on that volume.

RESTORE FULL command syntax

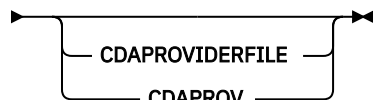
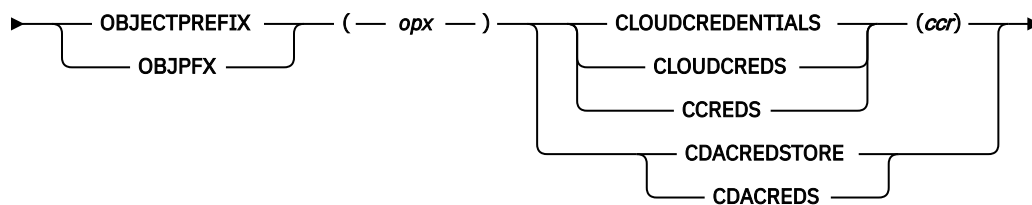


A: Optional keywords for the RESTORE FULL command

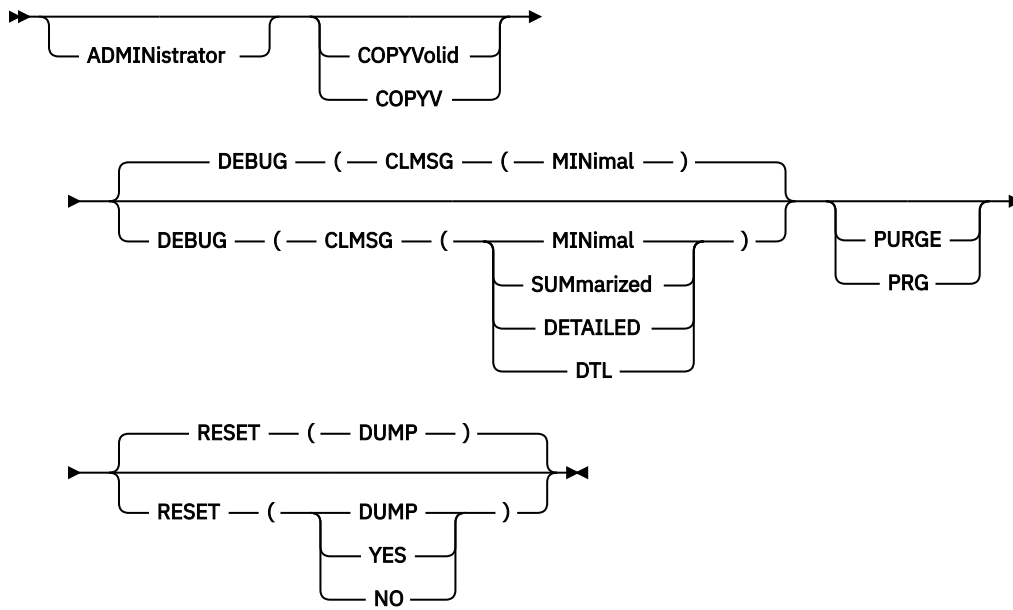


B: Additional keywords for RESTORE FULL

►► CLOUD — (— *cln* —) — CONTAINER — (— *con* —) —►

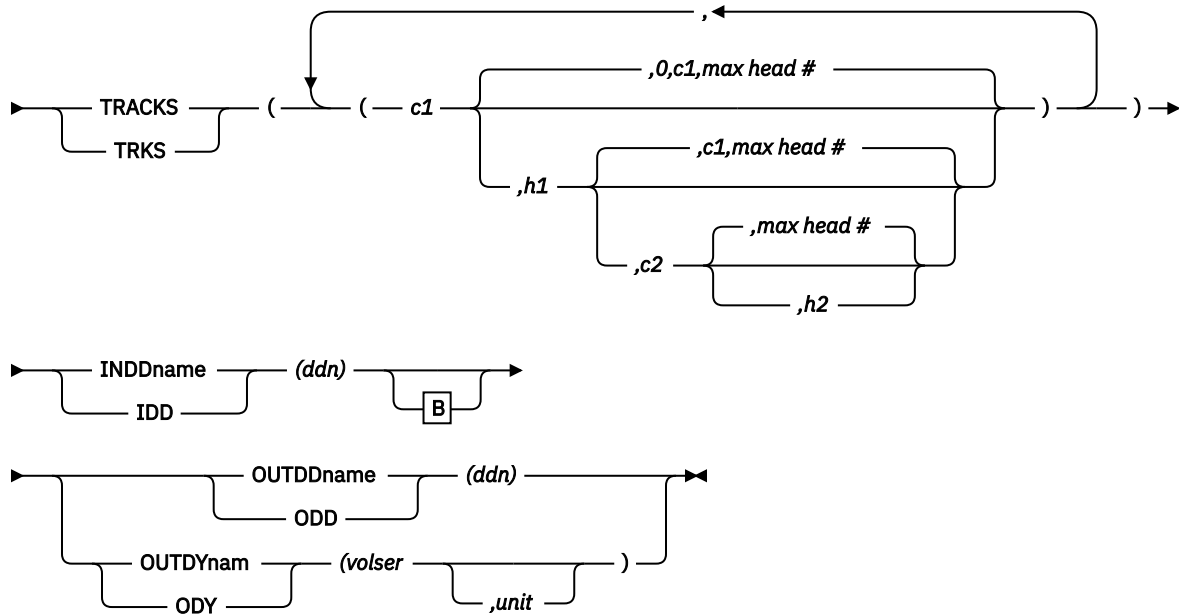


C: Optional keywords with RESTORE FULL and CLOUD specified



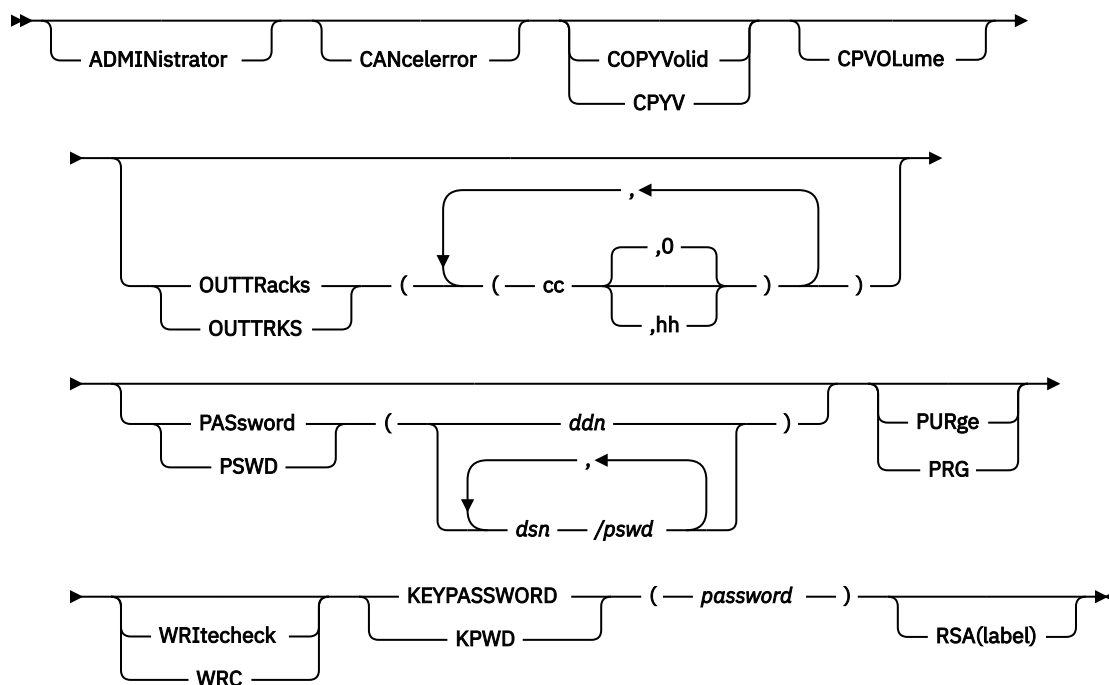
RESTORE TRACKS command syntax

➤ RESTore ➤

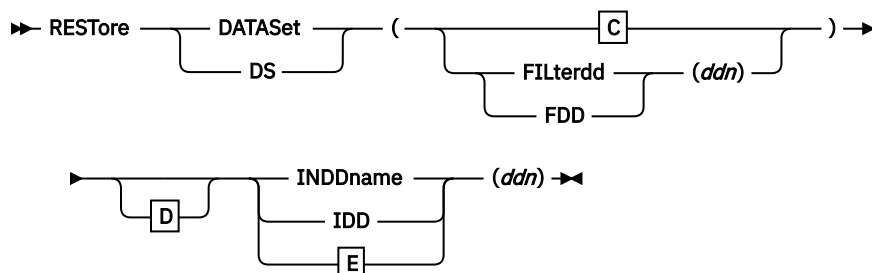


B: Optional keywords for the RESTORE TRACKS command

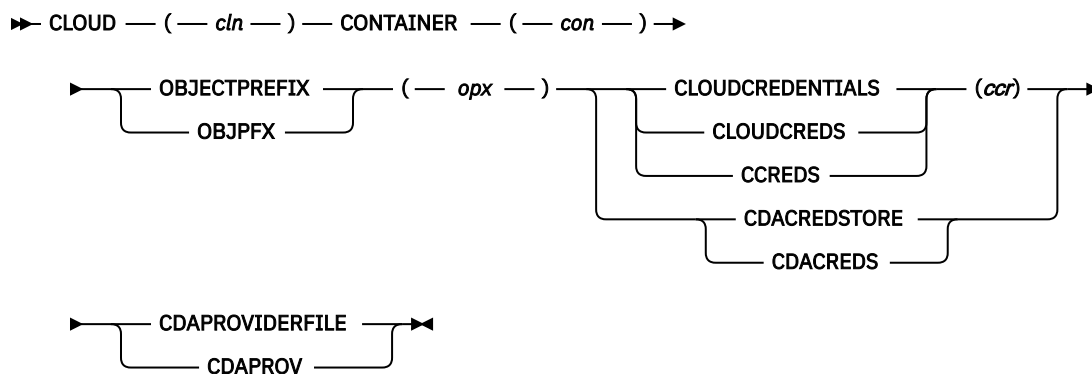
RESTORE Command for DFSMSdss



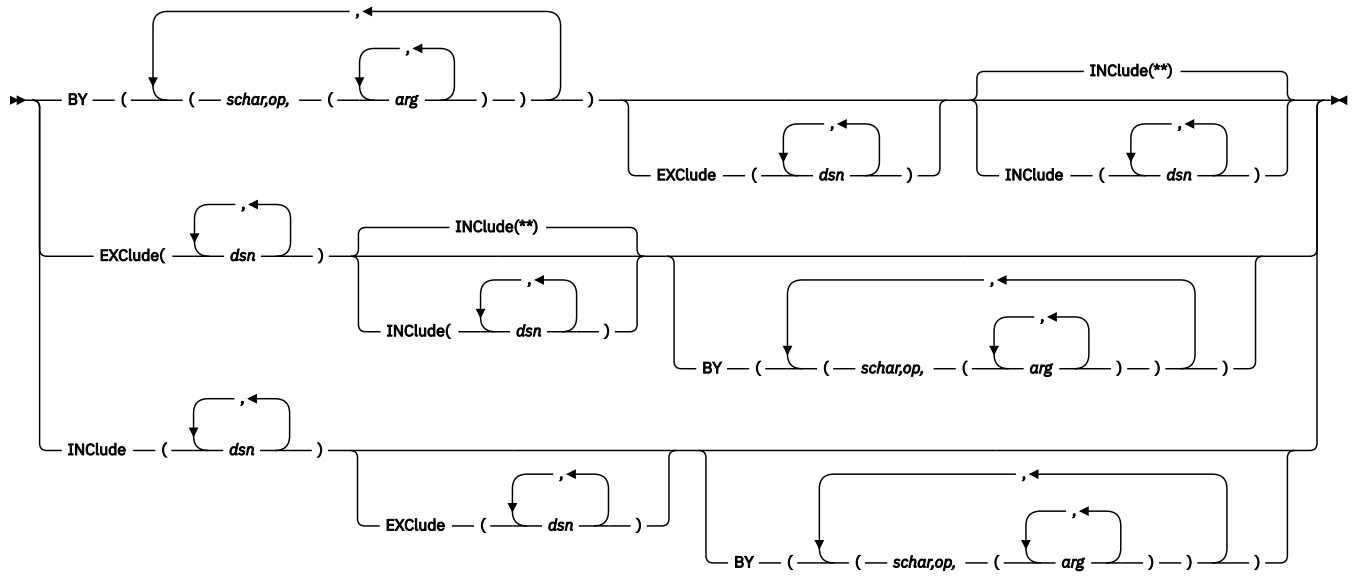
RESTORE DATASET command syntax for logical data set



E: Additional keywords used for logical data sets

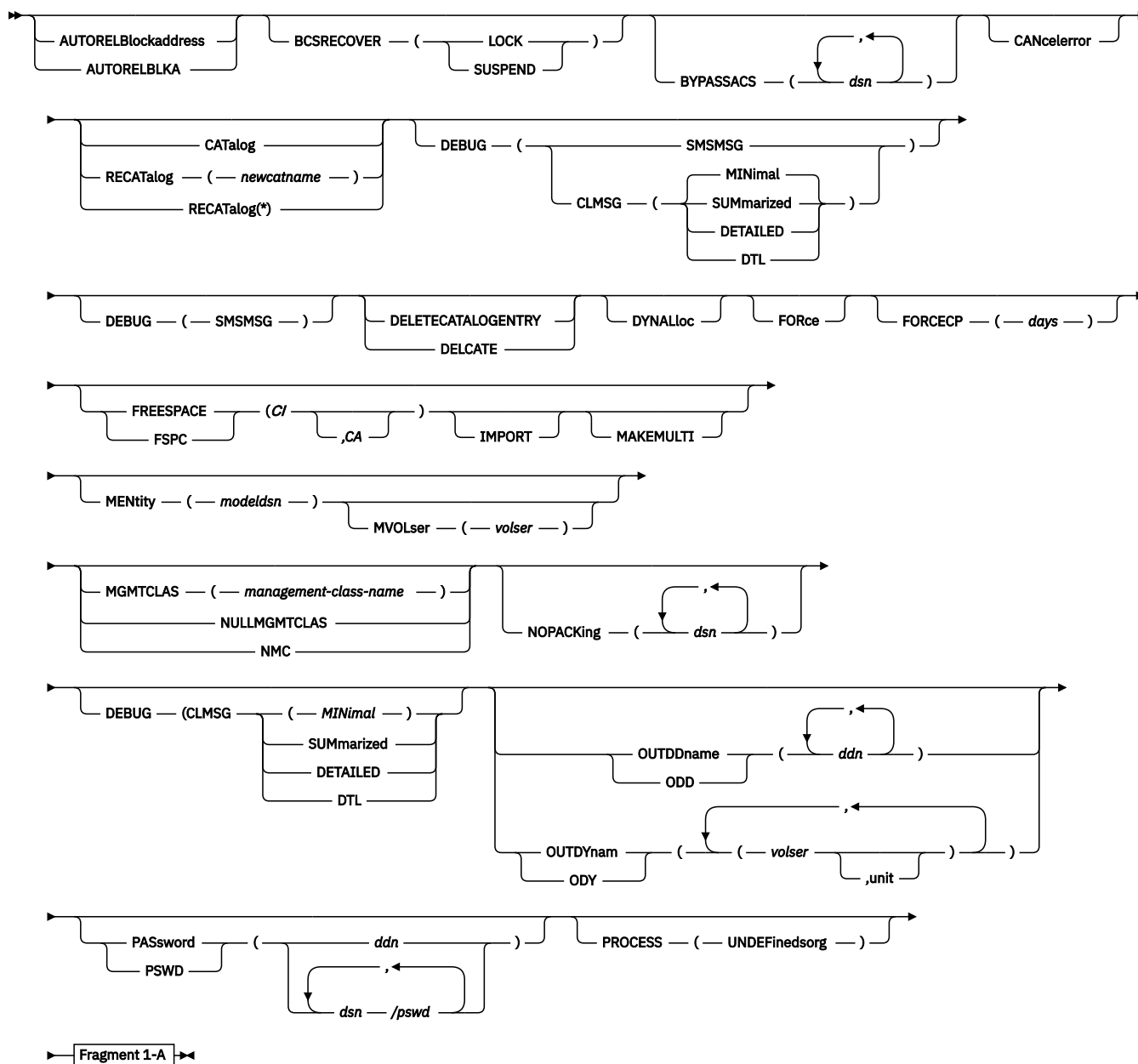


C: Additional keywords used for logical data sets

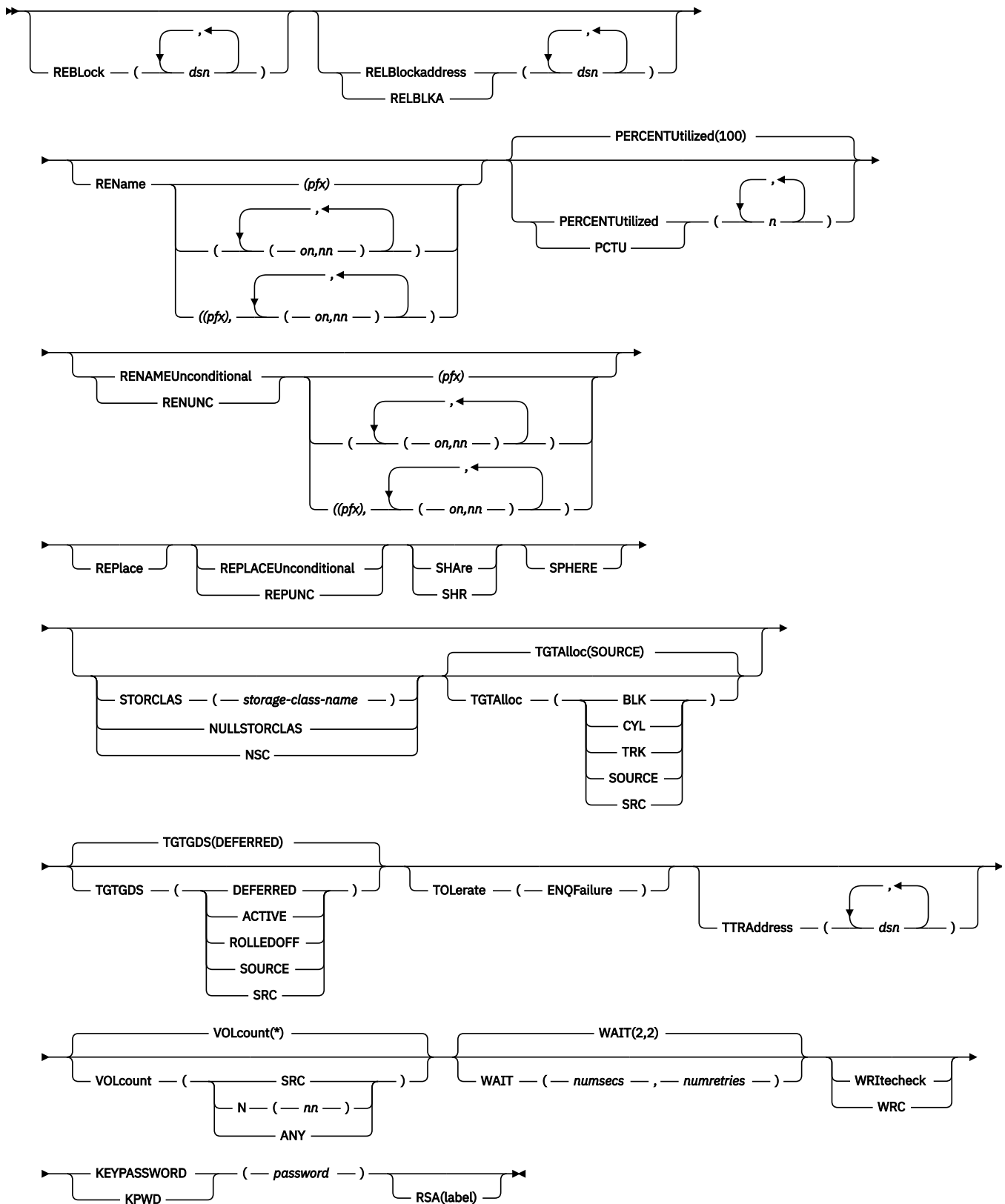


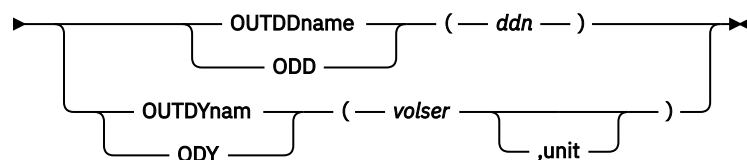
D: Optional keywords used for logical data sets (Part 1 of 2)

RESTORE Command for DFSMSdss

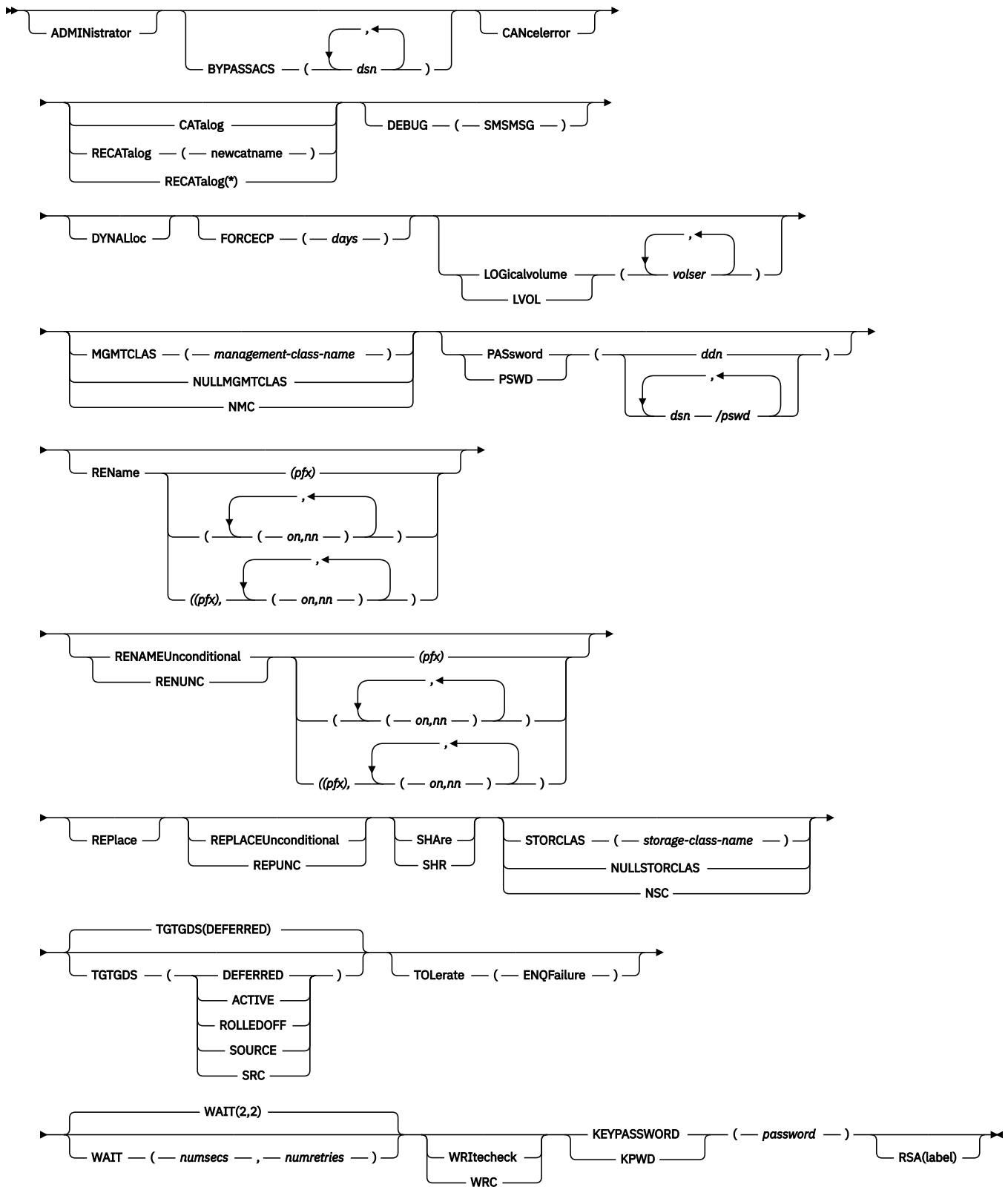


Fragment 1-A





460 z/OS: z/OS DFSMSdss Storage Administration



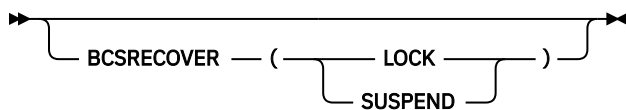
AUTORELBLOCKADDRESS specifies that direct access data sets are to be automatically processed as being organized by relative block address provided that they were accessed with an OPTCD setting indicating relative block addressing. These direct access data sets are to be processed by relative block address rather than by TTR, and without maintaining the relative track and record number for each record. The blocks must fit on the tracks of the target volume.

Note:

1. If any such data set is actually organized by TTR, the data set might become unusable.
2. The TTRADDRESS keyword takes precedence over the AUTORELBLOCKADDRESS keyword. Refer to the RELBLOCKADDRESS and TTRADDRESS keywords for more information.
3. AUTORELBLOCKADDRESS is ignored for direct access data sets with variable records formats or with standard user labels.

For more information about OPTCD, see [z/OS DFSMS Macro Instructions for Data Sets](#).

BCSRECOVER



BCSRECOVER facilitates recovery of pre-allocated logical data sets. The BCSRECOVER parameter applies to both RLS and non-RLS catalogs. You must specify one of the following subparameters:

BCSRECOVER(LOCK)

specifies a serialized close of user catalogs.

BCSRECOVER(SUSPEND)

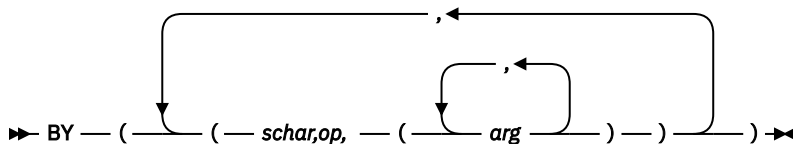
specifies that access to the catalog is suspended from all systems in the sysplex.

As part of BCSRECOVER processing, DFSMSdss automatically unlocks and resumes the catalog.

Note:

1. Programs using BCSRECOVER require READ access to RACF FACILITY CLASS resource IGG.CATLOCK.
2. If a user catalog is locked or suspended using IDCAMS or the MODIFY CATALOG command, the system ignores the BCSRECOVER keyword..

BY

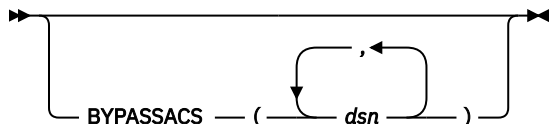


BY specifies that the data sets selected up to this point, by the processing of the INCLUDE and EXCLUDE keywords, are to be further filtered. To select the data set, *all* BY criteria must be met.

For more information, see:

- [“Filtering by data set characteristics” on page 258.](#)
- [“EXCLUDE” on page 471.](#)
- [“INCLUDE” on page 474.](#)

BYPASSACS



BYPASSACS specifies that Automatic Class Selection (ACS) routines are not to be invoked to determine the target data set's storage class or management class names. To specify BYPASSACS, RACF authorization might be required.

dsn

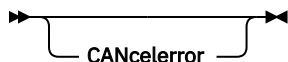
Specifies a fully or partially qualified data set name. To include all selected data sets, specify BYPASSACS(**).

If a data set is being renamed, the old name must be specified.

Note: If BYPASSACS(dsn) is specified, all data sets that pass the BYPASSACS selection criteria are guaranteed the specified storage class. The combination of NULLSTORCLAS and BYPASSACS(dsn) forces the selected data sets to be non-SMS managed.

For more information about RACF authorization, see [Chapter 5, “Protecting DFSMSdss functions,” on page 35](#). For more information about data set names, see [“Filtering by data set names” on page 256](#).

CANCELERROR



CANCELERROR specifies that the restore function be ended for a permanent read error, or that the restore of a data set be ended for a write error.

- Permanent read error (permanent I/O error):

If CANCELERROR is specified, the restore task is ended. If this keyword is not specified, DFSMSdss attempts to recover from the input errors, but the results can be unpredictable as a result of the difficulty in repositioning and assembling the data.

- Write error, such as an invalid track format:

For data set restore, processing of the data set ends and the target data set is deleted. The restore operation continues with the next data set. For full volume and tracks restore, processing for the volume ends. Subsequent tracks are not processed.

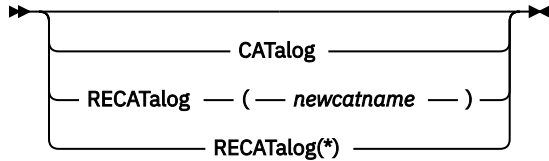
DFSMSdss allows you to change this default operation. A patch byte is provided to allow you to change the default handling of invalid tracks created during RESTORE processing.

CANCELERROR has no effect on the following types of errors on a DASD volume:

- Equipment check
- Command reject
- Intervention required
- Busout parity

For more information about the patch bytes, see [Chapter 14, “DFSMSdss patch area,” on page 213](#).

CATALOG or RECATALOG



For a logical restore operation, CATALOG instructs DFSMSdss to catalog data sets that it allocates. For a physical restore operation, CATALOG is used for non-VSAM single volume data sets; RECATALOG is ignored. DFSMSdss does not catalog VSAM data sets during physical restore. If the CATALOG keyword is specified, it is ignored when processing VSAM data sets. You must use IDCAMS DEFINE RECATALOG to catalog the data sets after the physical restore.

CATALOG

catalogs the target data set in a catalog as determined by the standard catalog search order. This is the default for VSAM data sets, multivolume data sets, and SMS-managed data sets (during logical data set restore). It is also the default for single-volume, non-VSAM, SMS-managed data sets (in physical data set restore).

RECATALOG(*newcatname*)

catalogs the target data set in the *newcatname* catalog.

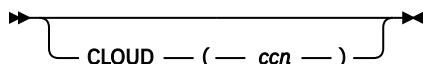
RECATALOG(*)

catalogs the target data set in the same catalog that points to the source data set. If the source data set was not cataloged, the new data set is not cataloged either.

Note:

1. CATALOG or RECATALOG fails if the target data set is already cataloged in the same catalog and RENAME is not specified.
2. CATALOG and RECATALOG are ignored when the target data set is preallocated.
3. RECATALOG is ignored for SMS-managed data sets.
4. Be careful when using RECATALOG(*newcatname*) because the target data set might already be cataloged outside of the standard order of search.
5. If RECATALOG is specified for a physical restore it is ignored. DFSMSdss does not attempt to catalog the data set. If the data set is already cataloged on another volume:
 - The catalog entry is not updated, and
 - The data set is restored, but it is not cataloged.
6. If CATALOG is specified for a physical restore of a single-volume non-VSAM data set, DFSMSdss attempts to catalog the data set. If the data set is already cataloged on another volume:
 - Message ADR385E, reason code 08, is issued,
 - The catalog entry will not be updated, and
 - The data set will be restored, but it will not be cataloged.
7. DFSMSdss physical data set restore does not create catalog entries for VSAM, multivolume non-VSAM, or preallocated data sets. The user must create all catalog entries. CATALOG and RECATALOG are ignored when the volume count of the data set cannot be determined. This can occur when a non-SMS managed sequential data set is defined with a data class, but has never been opened.
8. For multivolume non-VSAM data sets, use IDCAMS DEFINE NONVSAM.

CLOUD



ccn

Specifies the name of either a CDA provider file or an SMS construct that identifies the cloud storage the dump to be written on.

If the keyword **CDAPROVIDERFILE** is present, DFSMSdss will look for a z/OS Cloud Data Access (CDA) provider file with the name *ccn.json*. The provider file will not be used if it does not contain the key-value pair "enableDFSMSdss=YES".

If the keyword **CDAPROVIDERFILE** is absent, DFSMSdss will look for a network connection construct by the name *ccn* in the active SMS configuration.

Note:

1. Do not specify CLOUD during DUMP with OUTDD, CONCURRENT, COMPRESS, HWCOMPRESS, ZCOMPRESS, or RSA.
2. If the cloud solution that is being used is TCT, these keywords will be ignored, if the CLOUD is specified: VALIDATE and OPTIMIZE.
3. You must specify the CONTAINER and OBJECTPREFIX keywords when specifying the CLOUD keyword.
4. The name can be up to 30 characters in length.
5. To specify CLOUD, RACF authorization may be required.
6. DFSMSdss will look for a z/OS Cloud Data Access (CDA) provider file with the specified CLOUD name if CDAPROVIDERFILE keyword is specified.

CDACREDSTORE

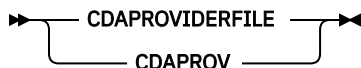


Specifies that the cloud credentials have been stored using the z/OS Cloud Data Access (CDA) Authorization Facility. When specified, DFSMSdss will request the cloud provider credentials from CDA using the cloud name specified in the CLOUD keyword.

Note:

1. Do not specify CLOUDCREDENTIALS if you are specifying CDACREDSSTORE.

CDAPROVIDERFILE

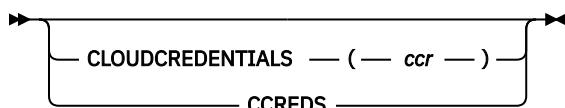


Specifies that DFSMSdss should look for a z/OS Cloud Data Access (CDA) provider file with the specified CLOUD name. If a valid CDA provider file does not exist or is empty, DFSMSdss will search for a network connection construct with the same name in the active SMS configuration.

Note:

- This keyword may be set to a default specification using ADPATCH offset X'62'. For more information on DFSMSdss patch bytes, see [Chapter 14, "DFSMSdss patch area," on page 213](#)
- For more information about the CDA Provider File, see ["Provider file" on page 12.](#)

CLOUDCREDENTIALS

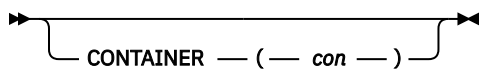


ccr

Specifies an up to 64 character credential (in EBCDIC) that is used when authenticating with a cloud. Valid characters are uppercase and lowercase letters A through Z, numerals 0-9, and the following characters: @#\$%&*-_=:<>?|{ }. You cannot use embedded spaces, commas (,), forward slash (/), parentheses (()), or semi-colons. DFSMSdss removes leading and trailing blanks.

Note:

1. Do not specify CDACREDSTORE when you specify CLOUDCREDENTIALS.
2. The credentials that are specified in your input command stream are not printed to the SYSPRINT output.
3. The credentials must be kept secure. Keep batch JCL jobs that specify this keyword in a data set or library that has limited access and is controlled by a security product. If the credentials cannot be kept secure, then do not use this keyword.

CONTAINER**con**

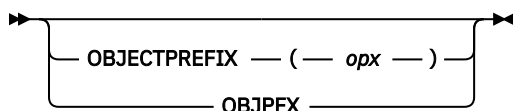
Specifies the container in which objects are stored. The container specified as input does not have to exist at the time of the backup. If necessary, DFSMSdss will create it.

A DFSMSdss specific prefix may be prepended to the specified container name, according to the cloud storage solution being used:

- When using a TCT cloud storage solution and no prefix is specified, the prefix prepended will be 'SYSZADR.'
- When using the Direct to Cloud storage solution, the prefix prepended will be 'SYSZADR-'.
- If either 'SYSZADR.' or 'SYSZADR-' is specified as part of the container name, no prefix will be prepended regardless of the cloud storage solution used.

Note:

1. The name can be up to 128 characters in length. If the CDACREDSTORE keyword is used, the container name may be folded to lowercase.
2. The allowable characters are uppercase letters A-Z, numbers 0-9, special characters \$ @ # - _ , and . in the non-first position.
3. Specify the CLOUD and OBJECTPREFIX keywords when you specify the CONTAINER keyword.

OBJECTPREFIX**opx**

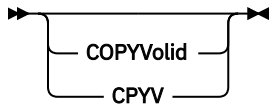
Specifies a unique prefix that DFSMSdss is to use for all of the objects that make up a particular backup. The prefix provides uniqueness amongst multiple backups in the same container. At the beginning of the backup process, DFSMSdss will determine if a backup using the same *opx* exists in the specified cloud and container. If a backup already exists with the same *opx*, then DFSMSdss will fail the backup. To overwrite a backup using the same *opx*, you can set a patch at offset x'5D'.

Note:

1. The name can be up to 44 characters in length, and the name cannot start with a number.

2. You must specify the CLOUD and CONTAINER keywords keywords when specifying the OBJECTPREFIX keyword.

COPYVOLID



COPYVOLID specifies that the volume ID from the dumped DASD volume is to be copied to the output DASD volume. This applies to a full restore operation; it applies to a tracks restore only if track 0 (zero) is to be restored.

When the volume serial number on a DASD volume is changed, the operator is notified. The operating system then initiates either:

- A demount if there is another volume with the same serial number
- A mount to get the volume with the new serial number mounted

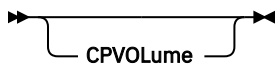
This might change the mount attributes of the volume. You should exercise operating precautions if there are two or more processors sharing the same DASD volume.

When the volume serial number is changed by using a COPYVOLID keyword or when both the dumped volume and the restored volume have different serial numbers, profiles are not built for the RACF-protected data sets on the restored volume or for the RACF DASDVOL for RACF-protected DASD volumes.

Note:

1. If you are doing a full restore operation and the input has VSAM data sets, the VOLID must be copied.
2. The COPYVOLID keyword is required for a full-volume restore operation of SMS-managed source volumes.
3. Exercise caution using the COPYVOLID keyword in a multiple task job step when two or more of the tasks are using the same output volume. If the output volume is made unavailable by the first task, all succeeding tasks that use the same output volume will fail.

CPVOLUME



CPVOLUME specifies that the output volume is a VM-format volume and that the OS-compatible VTOCs must begin on track zero, record five. You must specify the track range to be copied with the TRACKS keyword, as the OS-compatible VTOCs do not describe the extents of any data on the volume. You must also specify the ADMINISTRATOR keyword with CPVOLUME because DFSMSdss cannot check access authorization for VM data.

DATASET

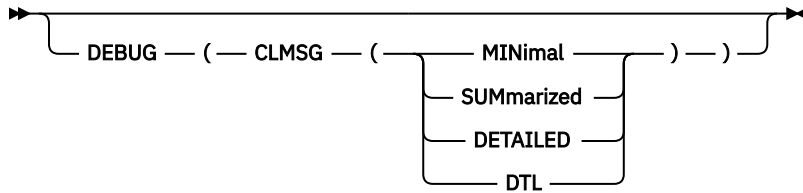


DATASET specifies a data set restore operation, using filtering.

Note: Either the FILTERDD, INCLUDE, EXCLUDE, or BY keyword must be used when DATASET is selected.

For more information about filtering, see [Chapter 16, “DFSMSdss filtering—choosing the data sets you want processed,”](#) on page 255.

DEBUG



You can use DEBUG as a diagnostic tool. When you specify the CLMSG subkeyword, DFSMSdss issues messages that provide details on the progress of a restore from an object storage cloud. When DEBUG(CLMSG) is not specified, MINimal is the default. Specify DEBUG(CLMSG) with one of the following sub-keywords:

CLMSG(MINimal)

Specifies that DFSMSdss is not to issue any messages that provide detail on the progress of a restore from an object storage cloud.

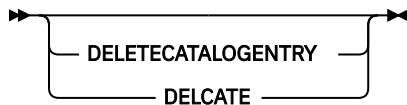
CLMSG(SUMmarized)

Specifies that DFSMSdss is to issue an informational message for each object that is restored from an object storage cloud.

CLMSG(DETAILED)

Specifies that DFSMSdss is to provide detailed information about each HTTP request that is made to an object storage cloud. If the CDAPROVIDERFILE keyword is used, debug information from Cloud Data Access (CDA) will be printed.

DELETECATALOGENTRY



DELETECATALOGENTRY specifies that the user wants to perform a disaster recovery and that the existing catalog entries for the target data sets might no longer be valid. If a target data set is cataloged but not available, then DFSMSdss performs a DELETE NOSCRATCH operation for the data set.

Specifying the DELETECATALOGENTRY command requires proper RACF facility class authorization.

For more information about RACF, see *z/OS DFSMSdss Storage Administration*. For more information about protecting keywords with RACF, see *z/OS Security Server RACF Security Administrator's Guide*.

Caution: Use extreme care with the DELETECATALOGENTRY keyword. If any of the following conditions exist, a DELETE NOSCRATCH operation is performed:

Condition 1

The target data set is cataloged but it is not found on the volume indicated by the catalog.

Condition 2

The target data set is cataloged but the volume that the catalog indicated does not exist on the restoring system and the restoring system is not sharing catalogs with another system.

Condition 3

The target data set is cataloged but the volume indicated by the catalog is offline on the restoring system and the restoring system is not sharing catalogs with another system.

Condition 4

The target data set is cataloged but the volume indicated by the catalog does not exist on the restoring system and the restoring system is sharing catalogs with another system.

Condition 5

The target data set is cataloged but the volume indicated by the catalog is offline on the restoring system and the restoring system is sharing catalogs with another system.

Condition 6

The target data set is a multivolume data set and some, but not all of the data, is no longer available. For this purpose, a multivolume data set is one in which any part of the data set resides on more than one volume. Examples include the following:

- A VSAM KSDS with the data- and index-components on one volume and an alternate index (AIX) on another volume.
- A VSAM KSDS with the index-component on a different volume than the data-component.
- A key-range VSAM data set with the key-ranges residing on more than one volume.

For conditions 1 and 2, it is appropriate to use DELETECATALOGENTRY to have DFSMSdss perform a DELETE NOSCRATCH operation for the catalog entries.

Condition 1 typically occurs during a disaster recovery when DFSMSdss restores or imports the catalogs before recovering the data sets. The target data set does not exist on the volumes indicated by the catalog. The restore will not be successful unless you or DFSMSdss delete the data set entries in the catalogs.

Condition 2 typically occurs during a disaster recovery when DFSMSdss restores the data sets to different volumes on a different operating system. DFSMSdss catalogs the target data sets, but points to volumes that do not exist on the target system. The restore will not be successful unless you or DFSMSdss delete the data set entries in the catalogs.

If you specify DELETECATALOGENTRY for conditions 3 through 6, the following damage is most likely to occur:

- Condition 3 typically occurs during a disaster recovery when DFSMSdss restores the data sets to different volumes on a different operating system. DFSMSdss cataloged the target data sets, but points to offline volumes on the target system. The restore will not be successful unless you or DFSMSdss delete the data set entries in the catalogs.
- Condition 3 can also occur in any environment where volumes are varied on and offline.

If you specify DELETECATALOGENTRY and then vary the volume back online, you might find that there are *two* copies of a data set: the original data set on the volume that was varied offline and the restored data set. DFSMSdss will no longer catalog the original data set.

- Conditions 4 and 5 typically occur in a shared system environment with a nonsymmetric system configuration. The following examples apply:
 - There are shared catalogs between two systems.
 - A data set is dumped from system A, but restored into system B.
 - The data set is cataloged in system A in a catalog that is shared by system B.
 - The volume containing the data set and, if applicable, its VVDS is on system A and is unavailable to system B.

Restriction: If you specify DELETECATALOGENTRY and then vary the volume back online, you might find that there are *two* copies of a data set: the original data set on the volume that was varied offline and the restored data set. If this happens, DFSMSdss can no longer catalog the original data set.

- DFSMSdss attempts to detect Condition 6 and issues an error message instead of performing the DELETE NOSCRATCH. However, it will not always be possible to do so. In that event, the DELETE NOSCRATCH is performed and DFSMSdss attempts to restore the data set with one of the following results:
 - DFSMSdss successfully restores the data set, and there is no residual data from the original data set. In this case, there is nothing further for you to do.
 - DFSMSdss successfully restores the data set, but there is residual data from the original data set. Although DFSMSdss restored the data set, you might find, for example, that there is an uncataloged AIX from the original data set. In this case, you will have to perform other corrective actions. These are the same actions that you would have had to perform even if you had not used the DELETECATALOGENTRY keyword.

- An error occurs during the restore that prevents the data set from being restored. The error results from the fact that the data set was only partially missing. For example, residual data and the VSAM Volume Record (VVR) might exist on a second volume. When the data set that is being restored is extended to that volume, a duplicate entry condition is created. Again, if this occurs, you will have to perform other corrective actions before you can successfully restore the data set.

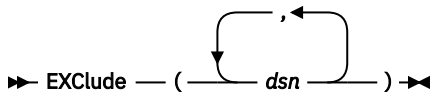
Note:

1. Use DELETECATALOGENTRY only for logical data set restore.
2. Do not use DELETECATALOGENTRY to restore partially damaged volumes or data sets.
3. You cannot use DELETECATALOGENTRY to delete catalog entries for any of the following:
 - User catalogs.
 - Migrated data sets (VOLSER=MIGRAT).
 - The SYSRES volume (VOLSER=*****).
 - A data set for which the volsers in the catalog do not match either the source volsers from the dump tape or the target volsers specified with OUTDD/OUTDYNAM.
4. If you are performing a disaster recovery and the original volumes are not available on the restoring system, you should also specify the IMPORT parameter. If you do not specify the IMPORT keyword, the system might prompt you to mount the volumes on which the target data resides (as indicated by the catalog). Even if you reply 'CANCEL', DFSMSdss attempts to perform a DELETE NOSCRATCH on the data set because it does not recognize the 'CANCEL' request. Even if the DELETE NOSCRATCH is successful, DFSMSdss might cause the following results:
 - DFSMSdss might fail to allocate the volume
 - DFSMSdss might issue the message ADR405E
 - DFSMSdss might not restore the data set
5. DELETECATALOGENTRY will not delete any catalog entry when the phantom entry indicates a different type of data set than the source data set.

DYNALLOC

DYNALLOC specifies that DFSMSdss is to use dynamic allocation, instead of enqueue, to serialize the use of data sets. This allows cross-system serialization in a JES3/MVS environment. The following conditions apply to DYNALLOC:

- The serialization is of value only when the dynamic allocation/JES3 interface is *not* disabled.
- Run time increases when you use DYNALLOC to serialize data sets (as opposed to enqueue). This is because overhead is involved in a dynamic allocation and serialization across multiple processors.

EXCLUDE**dsn**

Specifies the name of a data set to be excluded from the data sets selected by INCLUDE. Either a fully or a partially qualified data set name can be used.

For more information, see:

- [“BY” on page 463](#),
- [“INCLUDE” on page 474](#).

Guideline: When data sets have location-dependent data, list their names in the EXCLUDE list. This prevents DFSMSdss from restoring them even if you use the FORCE keyword.

FILTERDD

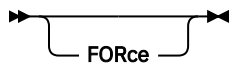


ddn

Specifies the name of the DD statement that identifies the sequential data set or member of a partitioned data set containing the filtering criteria to use. This is in the form of card-image records in DFSMSdss command syntax. It contains the keywords INCLUDE, EXCLUDE, and BY, completing the RESTORE command syntax.

Note: You must use FILTERDD when you have more than 255 entries in the INCLUDE, EXCLUDE, or BY list of subkeywords.

FORCE

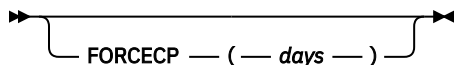


For a logical data set restore operation, FORCE specifies that DFSMSdss allows an unmovable data set or a data set allocated by absolute track allocation to be moved.

Note:

1. Use the EXCLUDE keyword with the FORCE keyword when you are restoring data sets that have location-dependent data. Naming these data sets in the EXCLUDE list prevents DFSMSdss from restoring them even when you specify the FORCE keyword.
2. Specify the FORCE keyword with the REPLACE or REPLACEUNCONDITIONAL keyword if you want to restore the data from unmovable data sets whose extents do not match.

FORCECP



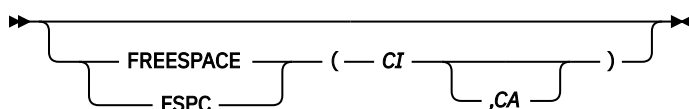
FORCECP specifies that any checkpointed data sets resident on the SMS volume or volumes can be logically restored or that MVS checkpointed data sets can be physically restored. Checkpoint indications are removed from the data sets.

days

Specifies a one-to-three digit number in the range of zero to 255. It also specifies the number of days that must have elapsed since the last referenced date before the data set can be restored.

Note: For IMS GSAM checkpointed data sets that are physically restored, FORCECP is not required, and checkpoint indications are not removed from the data sets regardless of whether or not FORCECP is specified.

FREESPACE



FREESPACE specifies free space values for DFSMSdss-allocated target VSAM data sets. If this keyword is omitted, the control interval and control area free space are the same as the source data set.

CI

Specifies the percentage of freespace to be kept in each control interval during allocation of the data set.

CA

Specifies the percentage of freespace to be kept in each control area during allocation of the data set. When omitted, the control area free space is the same as the source data set.

FULL**➡ FULL ➡**

FULL specifies that an entire DASD volume is to be restored. This is the default for the RESTORE command.

Note:

1. You cannot specify SHARE or TOL(ENQF) for FULL operations.
2. It is possible to duplicate a volume serial number during a restore FULL operation.

For more information about how to maintain data integrity when restoring a volume with FULL restore, see [“Data integrity considerations for full or tracks restore operations” on page 453.](#)

IMPORT

IMPORT specifies that the data sets that are being restored were dumped from a different system and they should be considered new data sets.

Because the restored data sets are new to the system, DFSMSdss modifies certain source data set processing. The following examples apply:

- DFSMSdss bypasses checking to see if you are authorized to read a data set with the same name as the one that was dumped.

If you are authorized to read the input dump data set that contains the data sets that are being restored, DFSMSdss considers that you have the authority to read any data set that is being restored. DFSMSdss continues to check to see that you are authorized to create a new target data set or replace an existing target data set.

If you are restoring the data set without renaming it, the restore might be unsuccessful. There are several common reasons for such a failure:

- You do not have sufficient authority to create a target data set with the same name as the source data set. You must obtain the required access authority to do so before you can restore the data set.
- A data set already exists with the same name as the source data set and you did not specify the REPLACE or REPLACEUNCONDITIONAL keyword. You must also specify the REPLACE or REPLACEUNCONDITIONAL keyword if you want the restore to replace an existing data set.
- A data set with the same name as the source data set is already cataloged, but is not available to the restoring system. You must make the volumes that contain the data set, and, as applicable, its VVDS, available to the restoring system before you can restore the data set.
- A catalog contains a phantom entry for a data set with the same name as the source data set. In this case, the data set does not exist on any volume. You can perform a separate DELETE NOSCRATCH operation for that data set name. Or you can specify the DELETEDCATALOGENTRY parameter to request that DFSMSdss perform a DELETE NOSCRATCH operation for you.



Attention: Do not use the DELETEDCATALOGENTRY keyword if the restoring system is sharing catalogs, but not the data set volumes, with another system.

- DFSMSdss can try to access a VVDS for a source data set in order to obtain information such as the resource owner. If you specify IMPORT, DFSMSdss suppresses a VVDS not available error.

To specify IMPORT, you must have the proper RACF facility class authorization.

Note:

1. IMPORT should be RACF-protected.
2. IMPORT is only supported for logical data set restore.
3. DELETEDCATALOGENTRY might have to be specified to successfully restore a data set with the old data set name.
4. When IMPORT is specified, DFSMSdss does not create a discrete data set profile unless the source data set is RACF-protected when it is dumped and you specify the MENTITY keyword.

For more information about RACF authorization, see [Chapter 5, “Protecting DFSMSdss functions,”](#) on page 35.

For more information about using the IMPORT keyword, see [“Protecting the usage of DFSMSdss”](#) on page 535.

For more information about using the DELETEDCATALOGENTRY keyword to successfully restore a data set with the old data set name, see [Chapter 6, “Managing availability with DFSMSdss,”](#) on page 39.

INCLUDE

Data set

➤ INCLUDE — (— *dsn* —) ➤

dsn

Specifies the name of a data set eligible to be restored. Either a fully or a partially qualified data set name can be used. See [“Filtering by data set names”](#) on page 256. If INCLUDE is omitted (but EXCLUDE or BY is specified) or if INCLUDE(**) is specified, *all* data sets are eligible to be selected for restoring.

For more information, see:

- [“BY”](#) on page 463,
- [“EXCLUDE”](#) on page 471.

z/OS UNIX files

➤ INCLUDE — (— 'pnm' —) ➤

The INCLUDE keyword is used to specify the path name ('pnm') of the Unix file(s) that are to be restored. 'pnm' is concatenated to the specified WORKINGDIRECTORY path, resulting in an absolute path of the file or directory to be included for processing. The desired file to be restored is identified by being the last file name in the specified path.

See [Chapter 17, “DFSMSdss filtering—choosing the z/OS UNIX files you want processed,”](#) on page 265.

Note:

1. Any parent directories that do not exist are created using the attributes stored in the backup. If existing parent directories are found with names matching those in the backup, they are not modified.
2. The specification of an asterisk wildcard is permitted to determine what files reside in the backup. See [“Determining what files are in a backup”](#) on page 90.

INDDNAME



ddn

Specifies the name of the DD statement that identifies the (input) dump data set. This data set can be on a tape or DASD volume.

Note: Concatenating multiple DFSMSdss dump data sets for the input for RESTORE is not supported and the results are unpredictable. Data sets on any dump data set after the first one in the concatenation will not be restored.

KEYPASSWORD

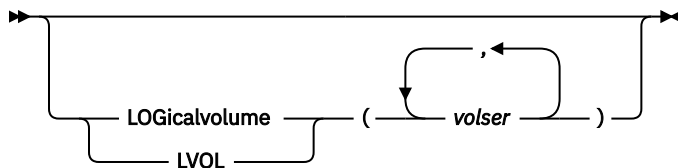


KEYPASSWORD specifies an 8 to 32 character password (in EBCDIC) that is used to generate a clear TDES triple-length key or a clear 128-bit AES key.

If you specified the **KEYPASSWORD** keyword on a previous **DUMP** command, you must also specify this value when restoring the dump through the **RESTORE** command.

Note: The KEYPASSWORD password that is specified in your input command stream is not printed in the SYSPRINT output.

LOGICAL VOLUME



LOGICALVOLUME specifies, for a *physical* data set restore operation, the volume serial numbers of the source DASD volumes that are to be processed. For example, if you have taken a data set dump from volumes 111111, 222222, and so forth, but you want to restore only some data sets from source volume 222222, specify LOGICALVOLUME (222222). LOGICALVOLUME is useful for restoring multivolume data sets.

MAKEMULTI



MAKEMULTI allows DFSMSdss to convert single volume data sets into multivolume data sets. The default is not to convert single volume data sets into multivolume data sets.

This keyword applies only to SMS-managed target data sets. Only single volume, non-VSAM data sets are eligible to be changed into multivolume data sets.

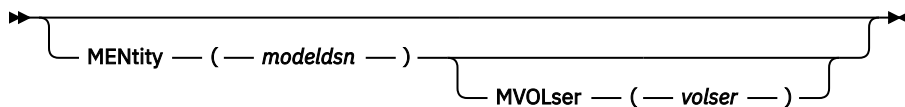
SMS-managed target data sets are given a volume count (VOLCOUNT) that is either:

- The number of SMS output volumes specified in the RESTORE command, if output volumes are specified through OUTDDNAME or OUTDYNAM.
- The number of volumes in the target storage group or 59, whichever is less.

A data set's volume count is the maximum number of volumes to which the data set can extend. At any one time, there might be a mixture of primary volumes (volumes on which space is allocated for the data set) and candidate volumes (volumes on which space can be allocated at a future time). The total sum of the primary volumes and candidate volumes is the data set's volume count.

Note: When MAKEMULTI is specified and VOLCOUNT is also specified with an option other than VOLCOUNT(*), the VOLCOUNT option overrides MAKEMULTI.

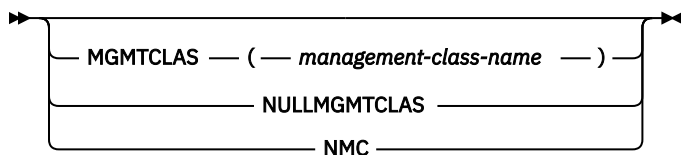
MENTITY



MENTITY specifies a model entity and, optionally, the serial number of the volume containing that entity (*volser*) to be used when DFSMSdss defines discrete profiles. These keywords are used to define the data sets to RACF. Specification of MVOLSER is optional when the model entity (MENTITY) is either (1) cataloged in an integrated catalog facility catalog or (2) a non-VSAM data set cataloged in the standard catalog search order. When MVOLSER is specified for a VSAM model entity, the *volser* specified must be the volume serial number of the catalog in which the model entity is cataloged. If these keywords are not specified, DFSMSdss defines the data set to RACF without using a model.

Restriction: You cannot specify the MVOLSER(*volser*) keyword by itself. It can only be specified with the MENTITY(*modeldsn*) keyword.

MGMTCLAS



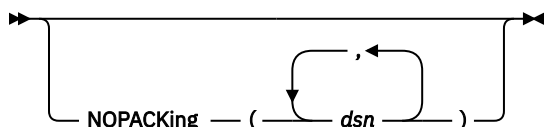
MGMTCLAS specifies the user-desired management class that is to replace the source management class as input to the ACS routines. You must have the proper RACF authority for the management class specified. The keyword itself does not require RACF authorization.

NULLMGMTCLAS/NMC specifies that the input to the ACS routines is to be a null management class rather than the source data set's management class.

MGMTCLAS and NULLMGMTCLAS are mutually exclusive.

Note: All SMS-managed data sets specified in the BYPASSACS keyword will be assigned the specified management class because the ACS routines will not be invoked. Non-SMS-managed data sets do not have a management class.

NOPACKING



NOPACKING specifies that DFSMSdss is to allocate the target partitioned data set only to devices that are the same or like device type as the source, and that DFSMSdss is to use track level I/O to perform data movement. This results in an exact track-for-track image of the source data set on the target volume.

dsn

Specifies the fully or partially qualified names of a PDS to be processed.

NOPACKing is only valid with a PDS. If specified, REBLOCK is ignored for the data set.

A PDS restored using NOPACKing is not compressed during data movement. NOPACKing can be used for a damaged PDS that is currently usable by an application but would be made unusable by compression or other rearrangement of the physical layout of the data.

Note: NOPACKing only applies to logical restore operations. Physical restore uses only track-level I/O. Therefore, no compression will take place against the PDS.

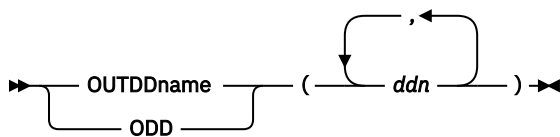
NULLMGMTCLAS

See “MGMTCLAS” on page 344.

NULLSTORCLAS

See “STORCLAS” on page 353.

OUTDDNAME

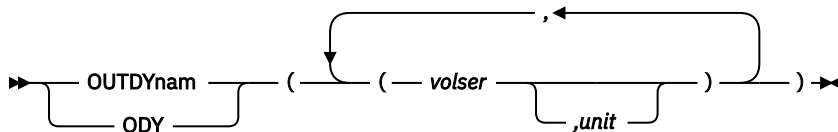


ddn

Specifies the name of the DD statement that identifies a volume to be restored to. To assure correct processing, each of the DD statements corresponding to a DDNAME (*ddn*) must identify only one volser. For a logical data set restore or when you specify spill files, you can specify multiple names, separated by commas. For any other type of restore, you can specify only one name.

For more information, see “OUTDYNAM” on page 477.

OUTDYNAM



OUTDYNAM specifies that the volume to be restored to is to be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*). For a logical data set restore or when you specify spill files, one or more volumes are allowed. For any other type of restore, only one volume is allowed.

volser

Specifies the volume serial number of a DASD volume to be restored.

unit

Specifies the device type of a DASD volume to be restored. This parameter is optional.

Notes for OUTDDNAME and OUTDYNAM Keywords

- OUTDDNAME or OUTDYNAM is required for a physical restore, even for SMS-managed data. They are optional for a logical restore operation, except:
 - For multivolume data sets that are preallocated on volumes that are different from the original source volumes; or
 - When the original source volume is not available, and the restored data set is not going to be SMS-managed.

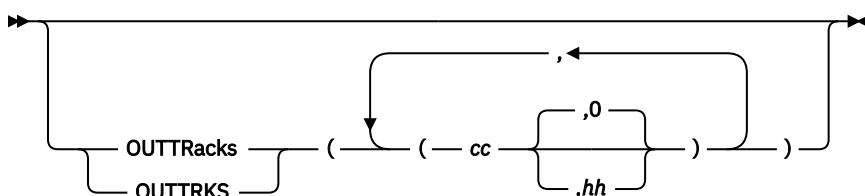
- DFSMSdss now distinguishes between non-SMS and SMS volumes specified in the OUTDDNAME or OUTDYNAM keywords. For non-SMS allocations, only the volumes that are non-SMS are considered for allocation. Similarly, only SMS volumes are considered for SMS allocations.

This distinction 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 being done are counted. If there are no volumes that match the type of allocation (SMS volumes for SMS allocation, or non-SMS volumes for non-SMS allocation), processing proceeds with a null volume list.

- If a non-VSAM data set is migrated by DFSMSHsm after a logical dump operation is performed on the data set, it should be recalled before a restore is attempted for the data set. If it is not recalled and if either OUTDDNAME or OUTDYNAM is specified indicating an output volume with REPLACE or REPLACEUNCONDITIONAL and RECATALOG(*), DFSMSdss issues a message indicating that an error occurred while trying to catalog the restored data set because it was already cataloged as a migrated data set. The data set will be restored on the volume specified by the OUTDDNAME or the OUTDYNAM, but the data set will not be cataloged. If neither OUTDDNAME nor OUTDYNAM is specified, DFSMSdss issues a message indicating that data sets with a volume serial of MIGRAT cannot be restored.

If the DD statement corresponding to the *ddn* of the OUTDDNAME contains the data set name and disposition but not the *volser*, DFSMSHsm recalls the data set automatically and DFSMSdss restores the data set.

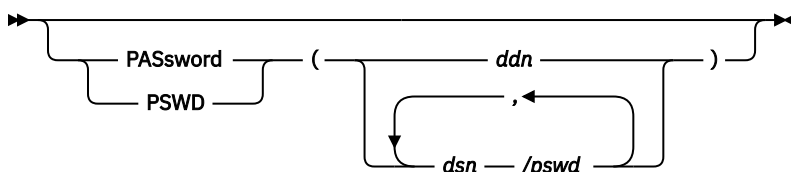
OUTTRACKS



OUTTRACKS specifies, for a track restore operation, the beginning location of the cylinder (*cc*) and head (*hh*) number of the target volume to which the track is to be restored. The number of (*cc*,*hh*) combinations specified in the OUTTRACKS keyword must be the same as the number of (*c1*,*h1*,*c2*,*h2*) combinations specified in the TRACKS keyword.

If OUTTRKS is not specified, the track is restored to its original cylinder and head number.

PASSWORD



PASSWORD specifies the passwords DFSMSdss is to use for password-protected data sets for all restore operations. (Password checking is bypassed for RACF-protected data sets.) The PASSWORD keyword is required only if:

- You do not have the required volume-level RACF DASDVOL or RACF DATASET access.
- The installation authorization exit does not bypass the checks.
- You do not want to be prompted for the password for VSAM data sets.

Note: Specify the passwords for all data sets that do not have RACF protection but do have password protection. During processing, a utility invoked by DFSMSdss might prompt the operator for a password. You can control authorization checking by using the installation authorization exit.

Catalog passwords are not supported to facilitate disaster recovery operations, application data transfers, and data set migration. Catalog protection via an access control facility, such as RACF, is the preferred method of protection.

ddn

Specifies the name of the DD statement that identifies the sequential data set, or member of a partitioned data set, that contains data set names and their passwords in the format *dsn/pswd*[,...]. This data set must contain card-image records in DFSMSdss command syntax format.

dsn/pswd

dsn is a fully or partially qualified data set name. *pswd* is its password. If no password follows the slash (/), *dsn* is treated as though it were *ddn*.

Note: Printing of actual data set passwords specified in your input command stream is suppressed in the SYSPRINT output.

PATH

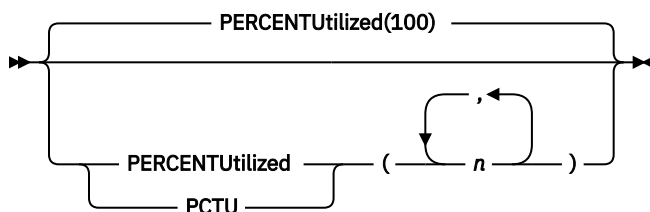
➡ PATH ➡

PATH specifies a path restore operation using filtering. See [Chapter 17, “DFSMSdss filtering—choosing the z/OS UNIX files you want processed,” on page 265](#) for an explanation of how to specify what z/OS UNIX files are to be processed.

PATH is mutually exclusive with FULL, TRACKS and DATASET.

Note:

1. INCLUDE and WORKINGDIRECTORY must be specified when PATH is selected.
2. Relative path naming is not supported.

PERCENTUTILIZED

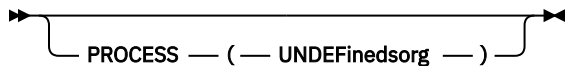
PERCENTUTILIZED specifies that DFSMSdss must stop allocating data sets to the target volumes when the allocated space reaches *n* percent of the total space on the target volume. The default is 100. Specify more than one *n* if you have more than one target volume (for instance, a volume for overflow). If there are more target volumes than you have values in this keyword, the last value is used for the remaining target volumes.

If the output volume is an extended address volume and DFSMSdss is allocating non-VSAM or unsupported VSAM data sets, the PERCENTUTILIZED value specifies that DFSMSdss is to stop allocating data sets to the target volumes when the allocated space reaches *n* percent of the track-managed space on the target volume.

Note:

1. PERCENTUTILIZED is ignored if the target data set is preallocated or if you do not specify an output volume.
2. PERCENTUTILIZED is not supported in an SMS environment.
3. PERCENTUTILIZED is valid only for logical data set restore operations.

PROCESS



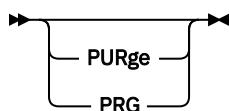
UNDEFInedsorg

specifies that the logical data set restore operation is to be allowed for data sets with undefined data set organization going to an unlike target device of a larger capacity and that DFSMSdss is to use track level I/O to perform data movement. This results in an exact track-for-track image of the source data set on target volume. To specify PROCESS, RACF authorization might be required.

Note: Even though the data is being copied to a device with a larger track capacity, the data might not fit on the output device. For example, if the source device is a 3380, and 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 the message ADR366W (Invalid Track Format) is issued.

For more information about RACF authorization, see [Chapter 5, “Protecting DFSMSdss functions,” on page 35.](#)

PURGE

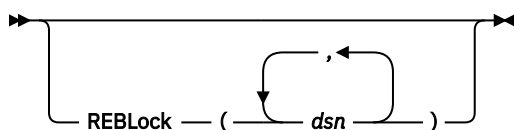


PURGE prevents a FULL (or TRACK) restore operation from ending while unexpired data sets remain on the target volume. For more information, refer to the REPLACE keyword for data set restore discussion.

Note: You must specify PURGE for a FULL volume RESTORE operation if the VVDS name on the target volume does not match the target volume serial number. This type of volume can be created using full volume COPY in conjunction with one of the following conditions:

- DUMPCONDITIONING is specified
- Neither COPYVOLID nor DUMPCONDITIONING are specified

REBLOCK



REBLOCK specifies that DFSMSdss is to reblock one or more of the selected sequential or partitioned data sets.

dsn

Specifies the fully or partially qualified names of a sequential or partitioned data set to be restored and reblocked.

The REBLOCK keyword is ignored for:

- Unmovable data sets
- Data sets with record format of U (except for partitioned load modules)
- Data sets with a record format of V, VS, VBS, or F
- Partitioned data sets with note lists (except for partitioned load modules)
- partitioned data sets that are also specified in the NOPACKING keyword
- Compressed data sets in zEDC format.
- Encrypted format data sets

Additionally, both the installation options exit and the installation reblock exit can override the specification of the REBLOCK keyword. The installation options exit can specify that no data set is to be reblocked. The installation reblock exit can specify whether a given data set is to be reblocked.

Some sequential and partitioned data sets have an attribute of being 'reblockable'. These data sets might be automatically reblocked by DFSMSdss independent of the REBLOCK keyword.

DFSMSdss uses DASD calculation services to determine the optimal block size for the target. The reblocking method used, DFSMSdss or DASD calculation services, is presented to the installation reblock exit.

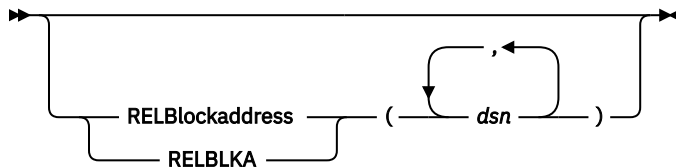
For more information about DFSMSdss processing of reblockable data sets, see [“Maximizing track utilization by reblocking data sets”](#) on page 153.

Note: For source data sets that are defined with RECFM=VB, DFSMSdss may not be able to accurately calculate the required amount of space when REBLOCK is coded. You can remove REBLOCK(**) from your parameters to allow the COPY to proceed without reblocking the data set. If this is not an option you can refer to [“Setting the percentage to overallocate target data set space \(OW27837\)”](#) on page 228 and use the ADRPATCH command to request that DFSMSdss overallocate the VB data set target so the REBLOCK is successful.

RECATALOG

See [“CATALOG or RECATALOG”](#) on page 465.

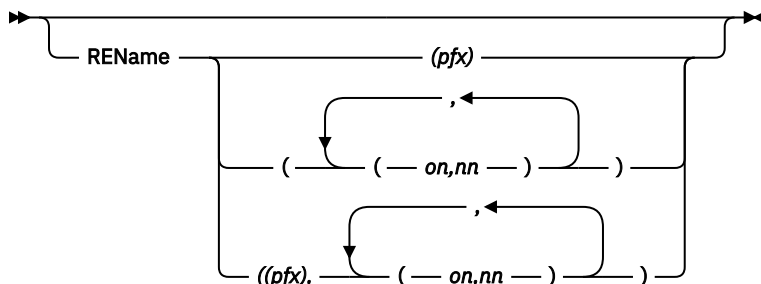
REBLOCKADDRESS



REBLOCKADDRESS identifies the direct data sets (BDAM) whose names match the fully or partially qualified names specified (*dsn*). These direct data sets are organized by relative block address instead of TTR and are to be restored block by block. DFSMSdss updates the block reference count (the relative position of the physical record as stored on its track) of dummy records. This keyword applies only to direct data sets with fixed record formats and without standard user labels.

Note: If the data set is actually organized by TTR, the data set might become unusable.

RENAME



RENAME specifies that, if a data set with the old name exists on the output DASD volume, DFSMSdss is to allocate a new data set with the new name and restore the data set. If the data set with the old name does not exist on the volume, the data set is restored with the old name. For a VSAM data set that already exists on another DASD volume and is cataloged, the VSAM data set is restored with the new name unless the new name also exists and is cataloged.

VSAM data sets can only be renamed during a physical data set restore by using the RENAMEUNCONDITIONAL keyword. If a data set is preallocated with the new name, it is not restored. If a data set is not preallocated, it is restored using the old name. This keyword only applies to *movable* data sets; therefore, unmovable data sets will not be renamed. RENAME and RENAMEUNCONDITIONAL are mutually exclusive; you cannot specify these keywords together.

Note: If the RENAME keyword is specified in conjunction with the REPLACE keyword, only one of the keywords take effect for any particular data set. The RENAME keyword takes precedence over the REPLACE keyword. If a source data set name matches the RENAME criteria, then rename processing is performed and replace processing is not performed. If a preallocated target data set exists with the new name as chosen by the rename criteria, then the restore fails even if the REPLACE keyword was specified. If you want to replace a preallocated target with the new name, specify the REPLACEUNCONDITIONAL keyword. If a source data set name does not match the rename criteria, and a preallocated target data set with the source name exists, the preallocated target data set is replaced.

px

Specifies the prefix to be used to replace the first-level qualifier of the data set name. It is optional, but if specified, must be the first parameter in the list of subkeywords. The prefix is used only if the (*on,nn*) parameters are not specified or the old name filters do not match the data set name.

on

Specifies the old name to be used as filtering criteria for matching data set names on the target volume.

nn

Specifies the new name to be used to derive the new data set name if the data set name selected by the corresponding old name filtering criteria matches the name of a data set that already exists on the target volume.

If none of the old name filters match the data set name and the prefix is specified, the prefix is used to derive the new name. If old name filters do not match and the prefix is not specified, the data set is not renamed. If the old name filter matches and there is an error in the new name filter, the data set is not renamed.

The syntax for the **prefix** is as follows:

- Single-level, fully qualified, unquoted DSNAME.
- 8 characters or less.
- The first character must be alphabetic or national.
- The remaining characters can be alphanumeric or national.

The syntax for the **old name** filter is exactly like that of the INCLUDE filter, and their rules match.

Examples of valid syntax for the **new name** filter are:

Restore the data set with the old name. This provides a powerful tool whereby some data sets can be restored with the old name and others can be restored with the new name.

If DSNAME has one level, then restore with old name.

A.**

First level of DSNAME replaced by A.

A.B.**

First two levels of DSNAME replaced by "A.B".

.A.*

Second level of DSNAME replaced by A.

****..BCD**

Last level of DSNAME replaced by BCD.

DATE..LIST**

First and last levels are replaced by DATE and LIST.

Q.*

If DSNAME has two levels, replace the first by Q.

Q.*.B

If DSNAME has three levels, replace the first and last by Q and B.

****.SYSLIST**

If DSNAME has three levels, replace the last by SYSLIST.

ABC.DEF

No asterisk in substring; replace the entire name with “ABC.DEF”.

Examples of invalid syntax for the **new name** filter are:

****.*.DATA.****

Invalid (level to be replaced is ambiguous).

SYS

Invalid (a qualifier is not completely replaced).

SYS*

Invalid (a qualifier is not completely replaced).

***SYS**

Invalid (a qualifier is not completely replaced).

SYS*TEM

Invalid (a qualifier is not completely replaced).

SYS.DAT%

Invalid (a qualifier is not completely replaced).

Restriction: The use of the wildcard character (%) is not supported for the new name filter of the RENAME, RENAMEU, or RENUNC keywords for the COPY or RESTORE operations.

You cannot change the number of qualifiers unless you use fully-qualified names, for example, RENUNC((A.B.C,A.B.C.D)).

If the new name filter has errors, the data set is not restored. The new name that is derived is truncated to fit 44 characters. If it ends with a period, that period is also truncated.

If the new name is not fully qualified, then it must contain the same number of qualifiers as the old name. For example, given the old name filter DATE . ** and the new name filter DATE . * . * . LIST, DATE . MARCH . TODAY . OLDLIST would be renamed, but DATE . MARCH . OLDLIST would not.

If two or more rules match the old data set name, the resulting new name is the first match.

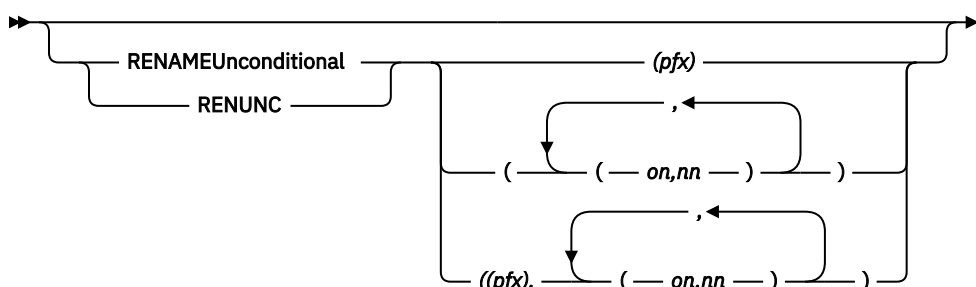
GDG relative generation filtering cannot be used for old or new names.

For more information about the use of the RENAME keyword, see [“Special considerations for RESTORE” on page 452.](#)

For more information about filtering, see [“Filtering by data set names” on page 256.](#)

[“INCLUDE” on page 474.](#)

RENAMEUNCONDITIONAL



RENAMEUNCONDITIONAL specifies that the data set should be restored with the new name, whether or not the data set exists on DASD with the old name. If the data set exists on the volume with the new name, it is not restored. RENAMEUNCONDITIONAL is not supported for VSAM Alternate Index (AIX) data sets during a physical data set restore. This keyword only applies to movable data sets; therefore, unmovable data sets will not be renamed. If the old name filter matches and there is an error in the new name filter, the data set is not restored.

Note:

1. RENAME and RENAMEUNCONDITIONAL are mutually exclusive; you cannot specify these keywords together.
2. RENAMEUNCONDITIONAL specifies that the data set must be restored with the new name, regardless of whether the data set exists on DASD with the old name. If the data set exists on the target volume with the new name and REPLACEUNCONDITIONAL is not specified, an error message is issued and the data set is not restored.
3. If the RENAMEU keyword is specified in conjunction with the REPLACE keyword, only one of the keywords take effect for any particular data set. The RENAMEU keyword takes precedence over the REPLACE keyword. If a source data set name matches the RENAMEU criteria, then rename processing is performed and replace processing is not performed. If a preallocated target data set exists with the new name as chosen by the rename criteria, then the restore fails even if the REPLACE keyword was specified. If you want to replace a preallocated target with the new name, specify the REPLACEUNCONDITIONAL keyword. If a source data set name does not match the rename criteria, and a preallocated target data set with the source name exists, the preallocated target data set is replaced.

px

Specifies the prefix used to replace the first-level qualifier of the data set name. It is optional but, if specified, must be the first parameter in the list of subkeywords. The prefix is used only if the (*on,nn*) parameters are not specified or the old name filters do not match the data set name.

on

Specifies the old name to be used as a filtering criterion to check if it matches the data set name.

nn

Specifies the new name to be used to derive the new data set name if the data set name matches the corresponding old name filtering criterion.

For more information about the use of the RENAME keyword, see [“Special considerations for RESTORE” on page 452](#).

For syntax rules, see the discussion of *px*, *on*, and *nn* under [“RENAME” on page 481](#).

REPLACE

REPLACE specifies that DFSMSdss is to search the target volumes for usable preallocated data sets. If a usable preallocated data set is found, it will be replaced with the data set from the source volume. If no preallocated target is found, DFSMSdss attempts to allocate a data set.

DFSMSdss searches for preallocated data sets as follows:

- For SMS-managed data sets, DFSMSdss first searches in the standard order of search for a catalog entry for the data set.
- For VSAM or Non-VSAM data sets that are not SMS-managed, DFSMSdss searches for a preallocated data set, cataloged or uncataloged, in the following order:
 1. In any output volumes specified.
 2. If DFSMSdss finds no preallocated data set, it then searches the catalogs in the standard order of search.

3. If no catalog entry is found for the data set, DFSMSdss searches the volume that the data set was dumped from.
- If no preallocated target is found, DFSMSdss attempts to allocate a data set. DFSMSdss invokes the ACS routines to determine if the data set should be SMS managed. If it should be SMS managed, then allocation is done according to the SMS constructs. If the data set should not be SMS managed, the output volumes specified are used. If no output volumes were specified, the volume the data set resided on at dump time is used. If allocation is successful, the data set is restored.
- PURGE is accepted and is treated the same as REPLACE. REPLACE only applies if the data set is not being renamed. REPLACEUNCONDITIONAL should be used when renaming a data set and a preallocated target data set should be replaced.
- The REPLACE and REPLACEUNCONDITIONAL keywords can not be specified together.

The volume that the data set was dumped from on system A is shared with system B, where the data set was being restored. If the data set was NONSMS when it was dumped, and the REPLACE keyword is specified on the RESTORE, the order of search, for a pre-allocated data set that is cataloged or uncataloged, will first search any output volumes specified. If no pre-allocated data set is found, then it will search the catalogs in the standard order of search. If no catalog entry is found for the data set, then it will search the volume the data set was dumped from.

Note:

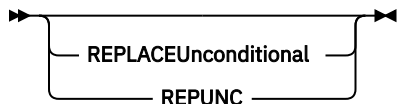
1. REPLACE or REPLACEUnconditional must be specified to restore to preallocated data sets.
2. If the source data set is an extended-addressable VSAM data set, then the target must also be an extended-addressable VSAM data set.
3. If REPLACE is specified with the RESTORE command, the SMS constructs already associated with the preallocated data set remain the same.
4. CATALOG and RECATALOG are ignored for preallocated data sets.
5. The target data set name must match the source data set name. REPLACEUnconditional must be specified to replace a target data set that has a name matching the rename criteria.
6. If you delete and reallocate a striped VSAM data set that is being replaced, DFSMSdss uses the same rules that apply to a new allocation. In general, it is best to not specify output volumes, or the VOLCOUNT keyword, when you are replacing striped VSAM data sets.
7. If the target data set is smaller than the source data set, DFSMSdss scratches the target data set and reallocates it with the size of the source data set.
8. Specify the FORCE keyword with the REPLACE keyword if you want to restore the data from unmovable data sets whose extents do not match.
9. If the RENAME or RENAMEU keywords are specified in conjunction with the REPLACE keyword, only one of the keywords take effect for any particular data set. The RENAME or RENAMEU keyword takes precedence over the REPLACE keyword. If a source data set name matches the RENAME or RENAMEU criteria, then rename processing will be performed and replace processing will not be performed. If a preallocated target data set exists with the new name as chosen by the rename criteria, then the restore fails even if the REPLACE keyword was specified. If you want to replace a preallocated target with the new name, specify the REPLACEUNCONDITIONAL keyword. If a source data set name does not match the rename criteria and a preallocated target data set with the source name exists, the preallocated target data set is replaced.
10. If the data set being restored is a large format sequential data set, but the preallocated target is not a large format sequential data set, the preallocated target will be used and turned into a large format sequential data set as long as it has enough allocated space for the data set being restored. If the preallocated new name target does not have enough allocated space, it will be scratched and reallocated as a large format sequential data set with enough space for the data set being restored.
11. If REPLACE is specified with the RESTORE command, the CA Reclaim attribute already associated with the preallocated data set remains the same.
12. 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.
- If VSAM, the cluster types match
- The DSORG matches.
- If the data set being processed is multivolume or single volume and the non-SMS preallocated data set matches the multivolume or single volume.

For more information about the REPLACE keyword, see [“Special considerations for RESTORE” on page 452.](#)

REPLACEUNCONDITIONAL

Data set



REPLACEUNCONDITIONAL specifies that DFSMSdss is to search the target volumes for usable preallocated data sets. If a usable preallocated data set is found, it is replaced with the data set from the source volume. When used with the RENAME or RENAMEUnconditional keywords, usable preallocated data sets with the new name are replaced. When used without the RENAME or RENAMEUnconditional keywords, usable preallocated data sets with the same name as the source data set are replaced. If no preallocated target is found, DFSMSdss attempts to allocate a data set. The REPLACE and REPLACEUnconditional keywords cannot be specified together.

Note:

1. REPLACEUnconditional must be specified to restore to preallocated data sets that match the rename criteria specified by the RENAME or RENAMEUnconditional keywords.
2. If the source data set is an extended-addressable VSAM data set, then the target must also be an extended-addressable VSAM data set.
3. If REPLACEUnconditional is specified with the RESTORE command, the SMS constructs already associated with the preallocated data set remain the same.
4. CATALOG and RECATALOG are ignored for preallocated data sets.
5. If DFSMSdss deletes and reallocates a striped VSAM data set that is being replaced, the same rules that apply to a new allocation are used to reallocate the data set. In general, it is best to not specify output volumes, or the VOLCOUNT keyword, when you are replacing striped VSAM data sets.
6. If the target data set is smaller than the source data set, DFSMSdss scratches the target data set and reallocates it with the size of the source data set.
7. If the data set being restored is a large format sequential data set, but the preallocated target is not a large format sequential data set, the preallocated target will be used and turned into a large format sequential data set as long as it has enough allocated space for the data set being restored. If the preallocated new name target does not have enough allocated space, it will be scratched and reallocated as a large format sequential data set with enough space for the data set being restored.
8. If REPLACEUnconditional is specified with the RESTORE command, the CA Reclaim attribute that is already associated with the preallocated data set remains the same.

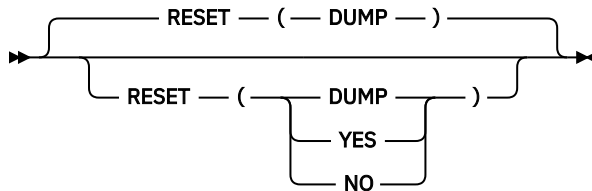
z/OS UNIX files

REPLACEUNCONDITIONAL specifies that if the regular file exists in the directory (preexisting) specified by the path name, it can be overwritten with the regular file from the source location (the backup version). This does not affect any parent directory files that are specified in the path.

For information about DFSMSdss and the way that it determines individual UNIX file attributes, see [Chapter 26, “z/OS UNIX file attributes,” on page 645.](#)

Note:

1. Applies to regular files only.
2. z/OS UNIX file processing ignores use of the REPLACE keyword. Only REPLACEUNCONDITIONAL is supported.

RESET

The RESET keyword directs how DFSMSdss determines if the data-set-changed indicator (DS1DSCHA) must be reset for the data sets on the volume being restored.

YES

DFSMSdss resets the data-set-changed indicator (DS1DSCHA=OFF) for all the data sets on the volume being restored

NO

DFSMSdss does not alter the DS1DSCHA values. Each DS1DSCHA represents the value the data sets had at the time the dump was taken.

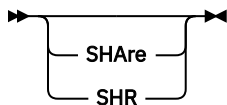
DUMP

DFSMSdss resets the data-set-changed indicator (DS1DSCHA=OFF) only if the RESET keyword was specified on the DUMP command. This is the default.

When RESET is not specified, DFSMSdss treats the RESTORE as though RESET(DUMP) were specified. It only resets DS1DSCHA if the RESET keyword was specified at DUMP time.

Note:

1. If the dump was created with a release of z/OS prior to z/OS V2R1, and RESET(DUMP) is specified, DFSMSdss does not alter the DS1DSCHA values, as if RESET(NO) were specified.
2. If you use a storage management program such as DFSMSHsm that performs incremental backups, you should be careful in coding the RESET keyword.

SHARE

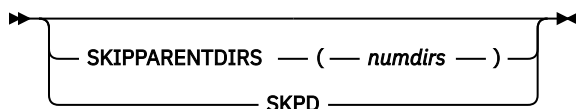
SHARE specifies that DFSMSdss is to share, for read access with other programs, the data sets that are to be restored. The resetting of the data set change indicator is bypassed if SHARE is specified on a data set restore operation.

SHARE and FULL are mutually exclusive; you cannot specify these keywords together.

The SHARE keyword is not honored for VSAM data sets. Exclusive control is obtained for VSAM data sets even if the SHARE keyword is specified. Neither read access nor write access by other programs is allowed while the VSAM data set is being restored, regardless of the share options defined for the data set.

The SHARE keyword is not honored for PDSEs that are targets of a RESTORE operation. This guarantees that while DSS is writing to a target PDSE, no other application can be updating that PDSE or its members.

SKIPPARENTDIRS



SKIPPARENTDIRS specifies that DFSMSdss is to skip restoring the first *numdirs* of the selected path.

numdirs

Specifies a decimal number from 0 to 512 that designates the number of directories to skip from the selected path.

If SKIPPARNTDIRS is not specified, SKIPPARNTDIRS(0) is the default. DFSMSdss will attempt to restore all files along the selected path.

If *numdirs* is greater than or equal to the number of parent directories in the specified path, then all parent directories are omitted from the restored path.

Restriction: The use of SKIPARENTDIRS with *numdirs* greater than 0 is limited to a single wildcard-free path name entry in the INCLUDE keyword. Otherwise, ADR129E is issued and the task is terminated.

SPHERE

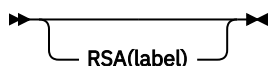


SPHERE specifies that for any VSAM cluster dumped with the SPHERE keyword, DFSMSdss must also restore all associated AIX clusters and paths. Individual sphere component names need not be specified; only the base cluster name is required.

Note:

1. If an AIX is dumped without the SPHERE keyword, during a restore DFSMSdss treats the AIX as a normal VSAM KSDS.
2. The base cluster name must be specified to process the entire sphere. If the SPHERE keyword is specified but the base cluster name is not, none of the associations will be processed.

RSA



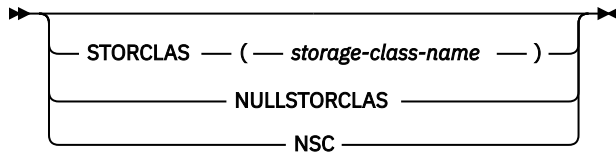
RSA

The RSA keyword allows you to specify a different label of an existing RSA private key that is present in the ICSF PKDS than what was specified on the RSA keyword during the DUMP command. The RSA public key is used to encrypt a randomly generated data key, so that the encrypted data key can be stored on the output medium.

Note:

1. If the RSA keyword was specified during the DUMP command, it doesn't need to be specified on the RESTORE command when the same label for the same RSA private key exists in the ICSF PKDS.

STORCLAS



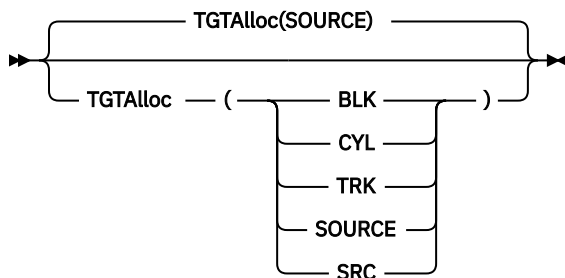
STORCLAS specifies the user-desired storage class that is to replace the source storage class as input to the ACS routines. The user must have the proper RACF authorization for the storage class specified. The keyword itself does not require authorization.

NULLSTORCLAS/NSC specifies that the input to the ACS routines is to be a null storage class rather than the source data set's storage class.

STORCLAS and NULLSTORCLAS are mutually exclusive.

Note: If BYPASSACS(dsn) is specified, all data sets that pass the BYPASSACS selection criteria are guaranteed the specified storage class. The combination of NULLSTORCLAS and BYPASSACS(dsn) forces the selected data sets to be non-SMS managed.

TGTALLOC



TGTALLOC specifies, during a logical data set restore function, how DFSMSdss will allocate the target data set.

BLK

Specifies to allocate by blocks.

CYL

Specifies to allocate by cylinders.

TRK

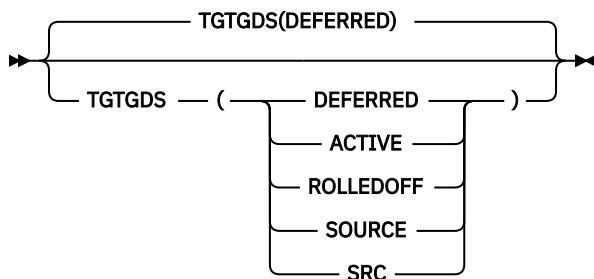
Specifies to allocate by tracks.

SOURCE/SRC

Specifies to allocate with the same space allocation type as that of the source data set.

Note:

1. If the TGTALLOC keyword is omitted, the target allocation defaults to source.
2. If SRC is specified and if the source data set is allocated by track or if TRK is specified, then the final VSAM allocation might be different from the requested one because of VSAM allocation rules.
3. If BLK is specified for VSAM data sets, TRK is used instead. The final VSAM allocation can be different from the requested one because of VSAM allocation rules.

TGTGDS

TGTGDS specifies in what status, during a data set operation, that DFSMSdss is to place nonpreallocated SMS-managed GDG data sets.

DEFERRED

Specifies that the target data set is to be assigned the DEFERRED status.

ACTIVE

Specifies that the target data set is to be assigned the ACTIVE status, for example, rolled into the GDG base.

ROLLEDOFF

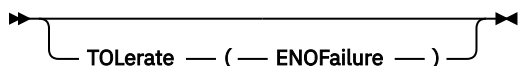
Specifies that the target data set is to be assigned the rolled-off status.

SOURCE/SRC

Specifies that the target data set is to be assigned the same status as that of the source data set.

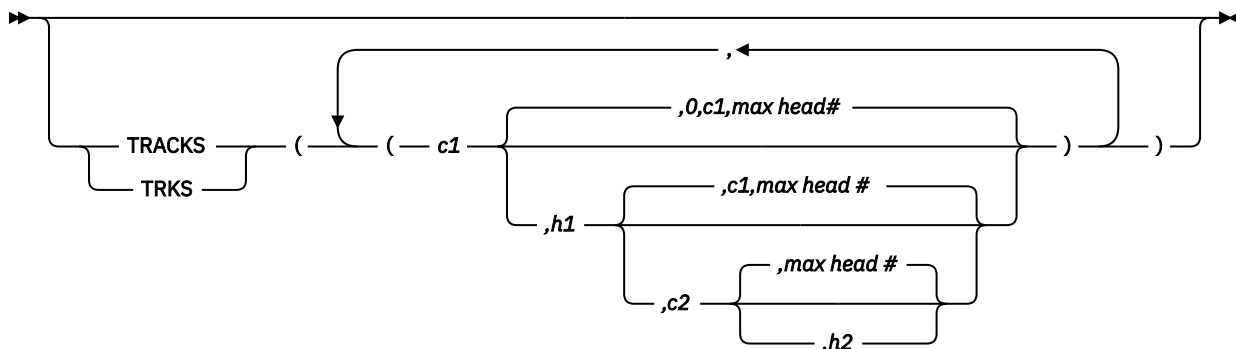
Note:

1. If the TGTGDS keyword is omitted, the target data set is assigned the DEFERRED status.
2. The requested target status of generation data sets must not violate generation data group rules.

TOLERATE**ENQFailure**

Specifies that target data sets are to be processed even though shared or exclusive access fails. TOL(ENQF) and FULL or TRACKS are mutually exclusive; you cannot specify these keywords together.

For more information on TOL(ENQF), see [Chapter 22, “Data integrity—serialization,” on page 561](#).

TRACKS

TRACKS specifies ranges of tracks to be restored (that is, a tracks restore operation). If any of the requested tracks are not in the input file, the restore operation is stopped.

c1,h1

Specifies the cylinder and head number of the beginning of the range. Specify hexadecimal numbers as X'c1' or X'h1'.

c2,h2

Specifies the cylinder and head number of the end of the range. Specify hexadecimal numbers as X'c2' or X'h2'.

Note:

1. The c2 must be greater than or equal to c1.
2. If c2 equals c1, h2 must be greater than or equal to h1.

You can enter X'FFFFFF' (or 268435455) as the high cylinder value. This causes DFSMSdss to choose the high cylinder value to be the end of the volume.

DFSMSdss verifies that the range is within the limits of the device. If you do not specify all four values for a range, DFSMSdss provides the missing values unless the omitted value causes a syntax error. No intervening values can be omitted. For example:

Specified Results**None**

Syntax error

c1

c1,0,c1, maximum head number

c1,h1

c1,h1,c1, maximum head number

c1,h1,c2

c1,h1,c2, maximum head number

c1,c2

Syntax error

,h1

Syntax error

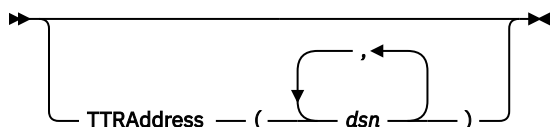
c1,h1,X'FFFFFF'

c1,h1,maximum cylinder number for the volume, maximum head number

You cannot specify TOL(ENQF) with the TRACKS keyword.

Data Integrity Note: It is possible to duplicate a volume serial number during a restore TRACKS operation. Follow the procedure in [“Data integrity considerations for full or tracks restore operations” on page 453](#) to maintain data integrity when restoring a volume with TRACKS restore.

For more information about using the TRACKS keyword during physical processing, see [“Physical processing” on page 26](#).

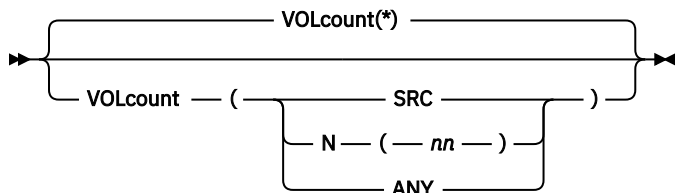
TTRADDRESS

TTRADDRESS identifies the direct access data sets whose names match the fully or partially qualified names specified (dsn). These direct access data sets are organized by TTR rather than by relative block addressing and are to be processed track by track. The target device track capacity must be equal to or greater than the source.

Note:

1. If a direct access data set is specified by both the RELBLOCKADDRESS and the TTRADDRESS keywords, the data set is not processed. Refer to the RELBLOCKADDRESS keyword for more information.
2. The TTRADDRESS keyword takes precedence over the AUTORELBLOCKADDRESS keyword processing for the specified data sets (dsn).

VOLCOUNT



VOLCOUNT specifies the method DFSMSdss uses to determine the number of volumes (volume count) for allocating the SMS target data set for a logical data set restore of VSAM or non-VSAM data sets.

* (asterisk)

Specifies that DFSMSdss determine the volume count for allocation according to the following conditions:

- If the source data set is a single-volume data set, one volume is allocated.
- If the source data set is a multivolume data set and either a list of volumes is not specified to DFSMSdss through OUTDDNAME or OUTDYNAM or there are no SMS volumes in the list, DFSMSdss allocates the same number of volumes that were in the multivolume source data set.
- If the source data set is a multivolume data set and a volume list is specified through OUTDDNAME or OUTDYNAM, the volume count is the number of SMS volumes in the list.

DFSMSdss does not adjust the final number of candidate volumes after the allocation is complete.

The * (asterisk) is the default for this keyword.

SRC

Specifies that DFSMSdss *rely on the source volume count* to determine the number of volumes to allocate for the target data set as follows:

- If no output volume list is specified, DFSMSdss allocates the same number of volumes that the source data set had.
- If a volume list is specified through OUTDDNAME or OUTDYNAM, the volumes in the list that are SMS-managed must be in the same storage group, and the allocation must be directed to that storage group.

DFSMSdss does not adjust the final number of candidate volumes after the allocation is complete.

N(nn)

nn represents the number of volumes to be used for SMS data set allocation. Any value between 0 and 59 can be specified with the following conditions:

- If *nn* is not zero and a volume list is specified through OUTDDNAME or OUTDYNAM, DFSMSdss allocates either the number of SMS volumes in the volume list or *nn*, whichever is less.
- If *nn* is zero and a volume list is specified through OUTDDNAME or OUTDYNAM, DFSMSdss allocates either the number of SMS volumes in the volume list or the number of volumes that were allocated for the source data set, whichever is less.
- If a volume list is specified through OUTDDNAME or OUTDYNAM and there are no SMS volumes in the list, or there is no volume list, DFSMSdss allocates either the number of volumes used by the source data set or *nn*, whichever is *more*.

DFSMSdss does not adjust the final number of candidate volumes after the allocation is complete.

ANY

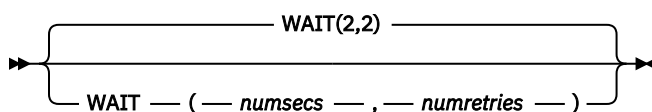
Specifies that DFSMSdss *use a maximum volume count* to allocate the SMS target data set as follows:

- DFSMSdss initially sets a volume count of 59 for the allocation.
- If the data set is allocated on more volumes than were used to allocate the source data set, DFSMSdss reduces the number of volumes used to the number of primary volumes needed to satisfy the allocation.
- If the data set is allocated on the same number or fewer volumes than were used to allocate the source data set, DFSMSdss reduces the number of volumes used to the number of volumes used for allocation of the source data set.

Note:

1. VOLCOUNT does not convert any of the following data sets to multivolume: PDS or PDSE data sets, single-volume data sets whose organization is undefined, or empty non-VSAM, single-volume data sets.
2. VOLCOUNT does not change the number of volumes for key-range KSDS data sets.
3. Guaranteed space is not honored when VOLCOUNT(ANY) is used.
4. VOLCOUNT(ANY) does not support keyed VSAM data sets that have an imbedded index. If VOLCOUNT(ANY) is specified and a data set has an imbedded index, the data set is processed as if VOLCOUNT(*) were specified.
5. VOLCOUNT(ANY) does not support any type of striped data set (physical, sequential, extended, or VSAM). If VOLCOUNT(ANY) is specified and a data set is striped, the data set is processed as if VOLCOUNT(*) were specified.
6. When you specify VOLCOUNT(ANY), the &ANYVOL and &ALLVOL read-only variables are not available to the storage group ACS routine.
7. For nonguaranteed-space, striped VSAM data sets: The minimum number of volumes that DFSMSdss allocates is determined by the number of stripes, which is based on the STORCLAS sustained data rate (SDR). DFSMSdss does not consider the number of volumes in the output volume list, or any of the VOLCOUNT specifications. If there are not enough enabled volumes in the STORGRP to support the SDR, DFSMSdss reduces the number of stripes. If there are excess volumes specified, those volumes become nonspecific (*) candidates.
8. For guaranteed-space, striped VSAM data sets: DFSMSdss allocates the number of volumes that are specified in the output list, regardless of the SDR. (To be striped, the SDR must be greater than zero.) The VOLCOUNT rules described above apply.

You can override VOLCOUNT keyword settings with the options installation exit routine, as described in [z/OS DFSMS Installation Exits](#).

WAIT

WAIT specifies to DFSMSdss the length of a wait in seconds and the number of retries to obtain control of a data set.

numsecs

Specifies a decimal number from 0 to 255 that designates the time interval, in seconds, between retries.

numretries

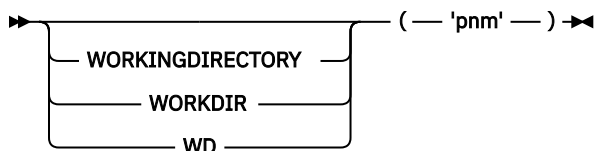
Specifies a decimal number from 0 to 99 that designates the number of retries to gain control of a data set.

The default for *numsecs,numretries* is (2,2), which specifies two retries at 2-second intervals. If you do not want to wait for a data set, specify 0 for either numsecs or numretries.

Note: The WAIT keyword does not control wait/retry attempts for system resources (such as the VTOC and the VVDS). For system resources, the default wait time is 3 seconds and the default retry count is 30. This results in a total wait time of 90 seconds.

For information about controlling the wait/retry attempts for system resources, see [“Controlling the wait/retry time for serialization of system resources \(PN11523\)”](#) on page 215.

WORKINGDIRECTORY



WORKINGDIRECTORY specifies the directory where DFSMSdss is to position itself in the z/OS UNIX hierarchy to begin restoring files. Any path specified in the INCLUDE criteria is restored beginning at the WORKINGDIRECTORY. See [Chapter 17, “DFSMSdss filtering—choosing the z/OS UNIX files you want processed,”](#) on page 265.

Note:

1. Must be specified with PATH
2. Must be an absolute path.

WRITECHECK



WRITECHECK specifies that the data restored is to be verified for successful completion. This keyword increases the overall elapsed time. The default is no WRITECHECK.

Note: The WRITECHECK keyword is not supported for extended-format sequential data sets.

DFSMSdss RESTORE process

[Table 26 on page 494](#) describes, in decision table format, DFSMSdss restore actions for physical processing of SMS-managed data sets. The specified RESTORE command keywords and the existence of the data set are shown in the upper half. The actions taken are shown in the lower half.

Table 26. Physical Data Set Restore Actions on SMS-Managed Data Sets																											
Action	Non-VSAM Physical Restore																	VSAM Physical Restore									
RENAME	N	N	N	N	Y	Y	Y	Y	Y	Y	–	–	–	N	N	Y	–	Y	–	–	–	–	–	N	N	N	N
RENUNC	N	N	N	N	–	–	–	–	–	–	Y	Y	Y	N	N	–	Y	–	Y	Y	Y	Y	Y	N	N	N	N
REPLACE	Y	Y	–	–	N	N	Y	Y	–	–	N	Y	–	N	N	N	X	N	N	N	Y	Y	–	Y	–	N	–
REPLACEU	–	–	Y	Y	N	N	–	–	Y	Y	N	–	Y	N	N	N	X	N	N	N	–	–	Y	–	Y	N	–
OLD DATA SET EXISTS	Y	N	Y	N	Y	N	Y	N	Y	N	X	X	X	Y	N	Y	X	X	X	X	X	X	X	Y	Y	Y	N
NEW NAME ON VOLUME	–	–	–	–	N	X	N	–	N	–	–	–	–	–	–	Y	Y	–	N	Y	N	Y	X	–	–	–	–

Table 26. Physical Data Set Restore Actions on SMS-Managed Data Sets (continued)

Action	Non-VSAM Physical Restore																VSAM Physical Restore										
Overlay old data set	T	-	T	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	T	T	-	-	
ALLOC with new name on USERVOL and restore	-	-	-	-	T	-	T	-	T	-	T	T	T	-	-	-	-	-	T	-	T	-	T	-	-	-	
ALLOC with old name	-	T	-	T	-	T	-	T	-	T	-	-	-	-	T	-	-	-	-	-	-	-	-	-	-	T	
Do not restore	-	-	-	-	-	-	-	-	-	-	-	-	-	T	-	T	T	T	-	T	-	T	-	-	-	T	-

Legend:

- Y = Yes
- T = Action taken
- N = No
- X = Doesn't matter
- – = Not Applicable
-

Figure 22 on page 495 describes general DFSMSdss actions for both physical and logical restore on non-VSAM data sets. It is *not* a program flowchart. Use it to clarify the restore actions on non-VSAM data sets under varying conditions.

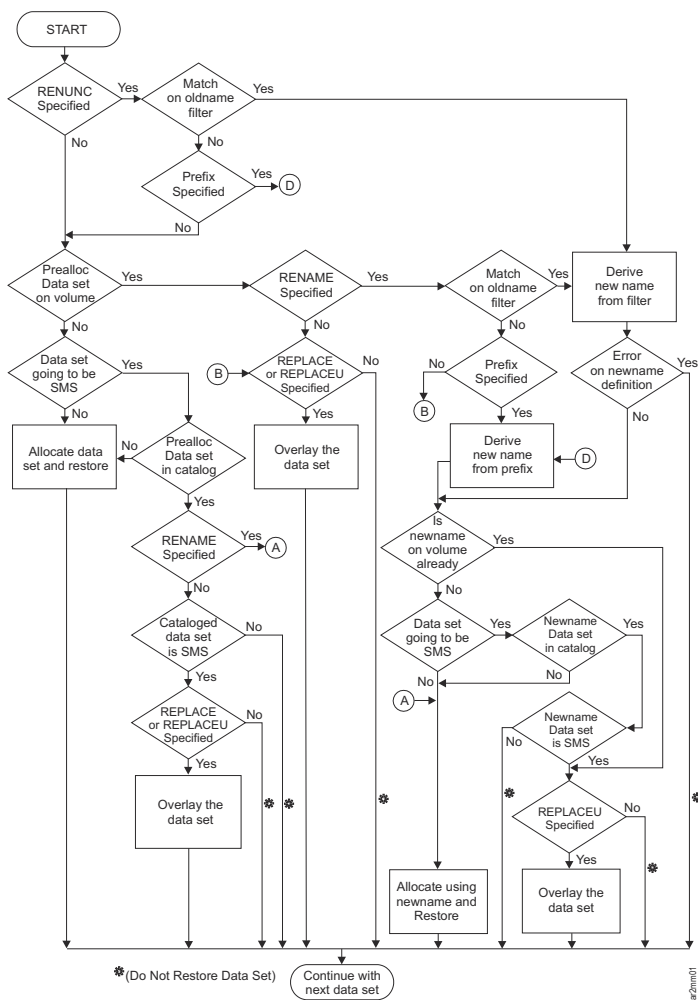


Figure 22. Restore Actions on Non-VSAM Data Sets

Assignment of class names by using the RESTORE and COPY commands

In an SMS environment, you can use STORCLAS, MGMTCLAS, NULLSTORCLAS, NULLMGMTCLAS, and BYPASSACS keywords with the RESTORE and COPY commands to influence the class names assigned to a data set. [Figure 23 on page 496](#) shows how these keywords can influence the storage and management class names of the target data set in a restore or copy operation. However, [Figure 23 on page 496](#) only addresses how the management and storage class names are assigned, not how the storage group name is assigned or how volumes are selected.

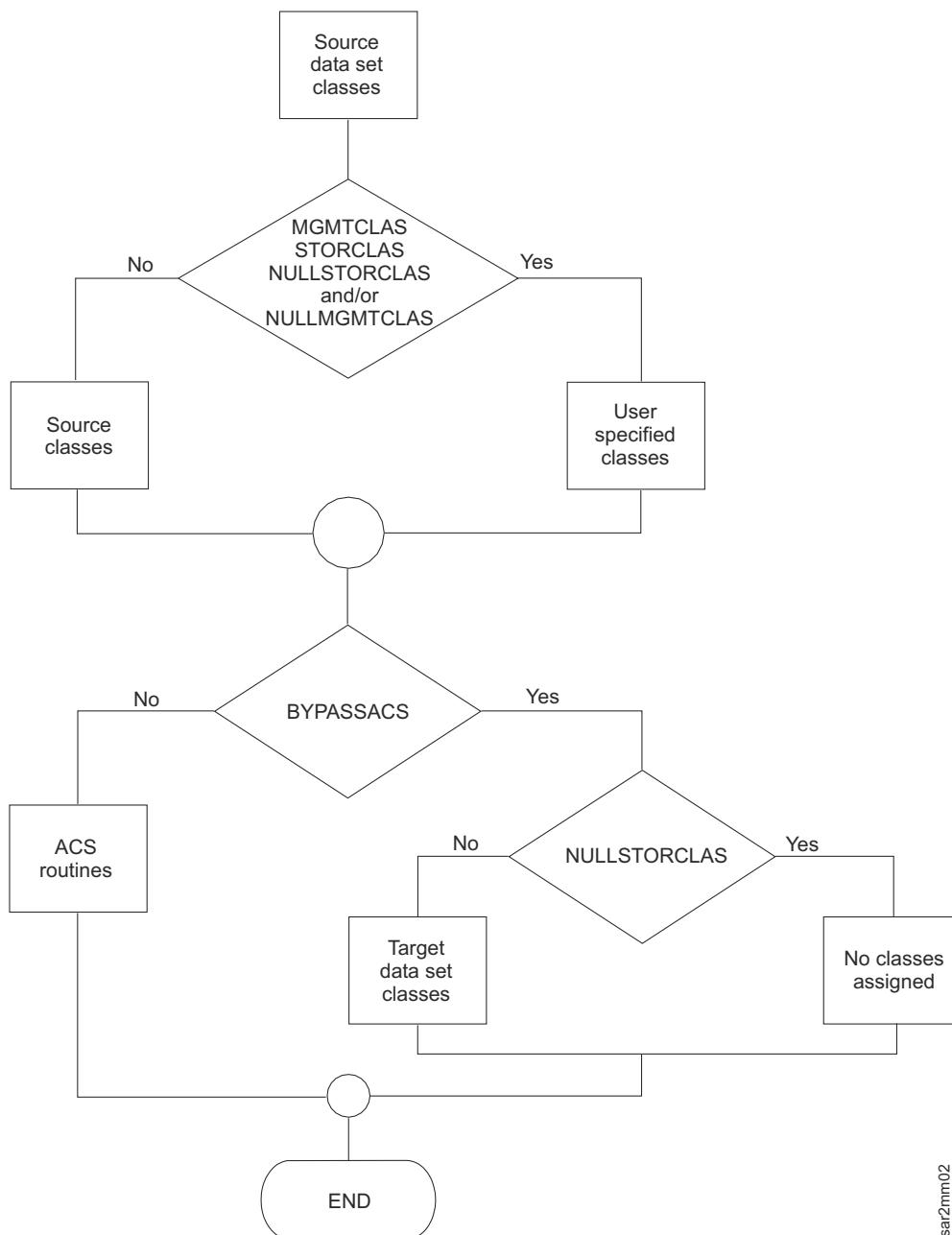


Figure 23. DFSMSdss Target Class Selection

Notes to Figure 23 on page 496:

- If you specify BYPASSACS, the source data set's class names or the class names specified with STORCLAS and MGMTCLAS are assigned to the target data set. If you do not specify BYPASSACS, ACS uses the source data set's class names, or the class names specified with STORCLAS and MGMTCLAS as input to assign the target data set's class names.

- If you specify NULLSTORCLAS, DFSMSdss passes a null storage class to ACS, which selects a storage class for the data set. If you specify NULLSTORCLAS and BYPASSACS together, the data set becomes non-SMS managed.
- NULLMGMTCLAS can only be used with SMS-managed data sets. Specifying NULLMGMTCLAS and BYPASSACS together causes the removal of the original management class of the data set.

For information about how target volumes are selected during restore and copy functions, see [Chapter 6, “Managing availability with DFSMSdss,”](#) on page 39.

Examples of full and tracks restore operations

Example 1 shows that DASD volume numbered 111111 will be restored from the first data set of standard label tape volumes called TAPE01 and TAPE02.

The command input to be substituted for a full and tracks restore operation are shown below in Example 1A and 1B respectively. To dump the same volume, see Examples 1, 1A, and 1B of the DUMP command.

Example 1: DASD volume-restore operation

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//TAPE      DD       UNIT=3480,VOL=SER=(TAPE01,TAPE02),
// LABEL=(1,SL),DISP=(OLD,KEEP),DSNAME=USER2.BACKUP
//DASD      DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//SYSIN     DD       *
command input
/*
```

For the command input, see

- [“Example 1A: full restore operation”](#) on page 497,
- [“Example 1B: tracks restore operation”](#) on page 497,
- [“Example 1C: tracks RESTORE—restore to different tracks”](#) on page 497.

Example 1A: full restore operation

```
RESTORE INDDNAME(TAPE) OUTDDNAME(DASD) PURGE
```

Example 1B: tracks restore operation

```
RESTORE TRACKS(1,0,1,5) INDDNAME(TAPE) -
OUTDDNAME(DASD) PURGE
```

Example 1C: tracks RESTORE—restore to different tracks

```
//JOBTRKS   JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//TAPE      DD       UNIT=3480,VOL=SER=TAPE04,
// LABEL=(1,SL),DISP=(OLD,KEEP),DSN=USER2.BACKUP
//SYSPRINT  DD       SYSOUT=A
//SYSIN     DD       *
RESTORE INDD(TAPE) OUTDYNAM(338000) -
  TRKS(200,10,200,10) /* INPUT TRK 200,10 ON SOURCE VOL*/ -
  OUTTRKS(100,0) /*RESTORE TO 100,0 ON TARGET VOL */ -
  PURGE /* OK TO OVERLAY THE TRK */ -
  /* EVEN IF UNEXPIRED DATA SET AT THE LOCATION */ -
  PSWD(ABC/WRITPSWD) /* PASSWORD FOR THE DATA SET */
/*
```

A track of dump data is restored to a cylinder and head number other than that from which it was dumped. The dump tape (which might have resulted from a full, tracks, or data set dump operation) contains a track dumped from cylinder 200 head 10 that is restored to cylinder 100 head 0.

Examples of physical data set restore operations

Example 2 specifies that data sets on standard label tape volume TAPE02 are to be restored to DASD volume numbered 111111.

Examples 2A through 2D below complement Examples 2A through 2G in [“Examples of physical data set dump operations”](#) on page 427 in any combination. For example, the dump tape produced in Example 2C in [“DUMP Command”](#) can be used as the input tape for Example 2A below.

Example 2: label tape volume RESTORED to DASD

```
//JOB2      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=ADRDSSU
//SYSPRINT  DD       SYSOUT=A
//TAPE      DD       UNIT=3480,VOL=SER=TAPE02,
// LABEL=(1,SL),DISP=(OLD,KEEP),DSNAME=USER2.BACKUP
//DASD1     DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//SYSIN     DD       *
command input (see Examples 2A, 2B, ... below)
/*
```

For command input, see:

- [“Example 2A: using the INCLUDE subkeyword to restore all data sets on a dump tape”](#) on page 498,
- [“Example 2B: using the INCLUDE and EXCLUDE subkeywords”](#) on page 498,
- [“Example 2C: using the INCLUDE, EXCLUDE, and BY subkeywords”](#) on page 498,
- [“Example 2D: with filtering data in a data set”](#) on page 499,
- [“Example 2E: using the LOGICALVOLUME and REPLACE keywords”](#) on page 499,
- [“Example 2F: using the REPLACE and RENAME keywords”](#) on page 499,
- [“Example 2G: using the REPLACE and RENAMEUNCONDITIONAL keywords”](#) on page 499,
- [“Example 2H: restore all data sets”](#) on page 499,
- [“Example 2I: restore data sets”](#) on page 500.

Example 2A: using the INCLUDE subkeyword to restore all data sets on a dump tape

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

Example 2B: using the INCLUDE and EXCLUDE subkeywords

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

All data sets are restored, except those ending with a qualifier of LIST.

Example 2C: using the INCLUDE, EXCLUDE, and BY subkeywords

```
RESTORE INDDNAME(TAPE) OUTDDNAME(DASD1) -
        DATASET(INCLUDE(**) -
                EXCLUDE(**.LIST) -
                BY((EXPDT,LE,80045)))
```

All data sets that satisfy the BY subkeyword except those specified in the EXCLUDE subkeyword are restored.

Example 2D: with filtering data in a data set

```
RESTORE INDDNAME(TAPE) OUTDDNAME(DASD1) -
        DATASET(FILTERDD(A1))
```

Note to Example 2D: The following DD statement must be added to the JCL shown in Example 2:

```
//A1 DD DSNAME=USER2.FILTER,DISP=SHR
```

This cataloged data set (USER2.FILTER) contains three card-image records. The information shown below is positioned in columns 2 through 72 of each record:

```
INCLUDE(**) -
  EXCLUDE(**.LIST) -
  BY((DSCHA,EQ,1))
```

Example 2E: using the LOGICALVOLUME and REPLACE keywords

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

Although the dump tape contains data sets from source volumes 111111 and 222222, only the data sets from source volume 111111 are restored. If preallocated data sets exist on the volume, DFSMSdss replaces them. Unmovable data sets that are not preallocated are not restored.

Example 2F: using the REPLACE and RENAME keywords

```
RESTORE INDDNAME(TAPE) OUTDDNAME(DASD1) -
        DATASET(INCLUDE(**)) LOGICALVOLUME(111111) -
        REPLACE -
        RENAME((USER2),(USER4.**,USER3.**))
```

In the above example, renaming takes place only for data sets that exist on DASD with the old name. Data sets with a first-level qualifier of USER4 are renamed to a first-level qualifier of USER3. The first-level qualifiers of all other data sets are replaced by USER2. Unmovable data sets are not renamed.

Example 2G: using the REPLACE and RENAMEUNCONDITIONAL keywords

```
RESTORE INDDNAME(TAPE) OUTDDNAME(DASD1) -
        DATASET(INCLUDE(**)) LOGICALVOLUME(111111) -
        REPLACE -
        RENAMEUNCONDITIONAL((USER2),(*.PEAR.**,*PLUM.**), -
                             (MY.SPECIFIC.DS,YOUR.ANY))
```

RENAMEUNCONDITIONAL is used for movable data sets; REPLACE is used for unmovable data sets. With the RENAMEUNCONDITIONAL keyword, movable data sets are renamed whether or not they exist on DASD with the old name. In the example, data sets with a second-level qualifier of PEAR are renamed by using a second-level qualifier of PLUM. MY.SPECIFIC.DS is renamed as YOUR.ANY. The first-level qualifier of all other movable data sets is changed to USER2. Unmovable data sets are not renamed.

Example 2H: restore all data sets

```
RESTORE INDDNAME(TAPE) OUTDDNAME(DASD1) -
        WAIT (1,99) DATASET(INCLUDE(**))
```

Example 2H shows you what to do if the data sets are in use for a short time interval during a restore operation. DFSMSdss waits for a second at a time and retries as many as 99 times if the data set is in use by another job.

Example 2I: restore data sets

```
RESTORE INDDNAME(TAPE) OUTDDNAME(DASD1) -  
        DATASET(INCLUDE(**)) TOL(ENQF) WAIT(0,0)
```

In Example 2I, DFSMSdss tries to serialize (ENQ) each data set. If the ENQ fails, DFSMSdss does not wait (WAIT(0,0)), and the data set is processed without serialization or enqueueing (TOL(ENQF)).

Examples of logical data set restore operations

The following are examples of logical data set RESTORE operations.

Example 1: logical data set RESTORE—output volumes not specified

```
//JOB5      JOB      accounting information,REGION=nnnnK  
//STEP1     EXEC     PGM=ADRDSSU  
//TAPE      DD       UNIT=3480,VOL=SER=TAPE04,  
// LABEL=(1,SL),DISP=(OLD,KEEP),DSN=USER3.BACKUP  
//SYSPRINT  DD       SYSOUT=A  
//SYSIN     DD       *  
RESTORE INDD(TAPE) -  
        DS(INCL(USER1.MULTVOL)) -  
        REPLACE  
/*
```

In Example 1, data set USER1.MULTVOL is restored. The location to which it is to be restored is not given. This RESTORE statement also applies when the data set has been scratched inadvertently after the dump operation. A multivolume data set is restored to the volumes from which it was dumped, provided the data set is preallocated on the output volumes. If it is not preallocated, the data set is restored to the first volume from which it was dumped that has adequate space to restore the data set as a single-volume data set.

The RESTORE statement can be modified as follows to support multiple restores of both single and multivolume data sets from dump tapes:

```
//SYSIN     DD       *  
RESTORE INDD(TAPE)          /* RESTORE          */ -  
        DS(INCL(USER1.CNTL.**)) /* USER'S CONTROL DATA SETS */ -  
        REPLACE             /* OVERLAY DATA SETS IF THEY EXIST */  
/*
```

Example 2: logical restore of an unmovable data set

```
//JOB6      JOB      accounting information,REGION=nnnnK  
//STEP1     EXEC     PGM=ADRDSSU  
//TAPE      DD       UNIT=3480,VOL=SER=TAPE04,  
// LABEL=(1,SL),DISP=(OLD,KEEP),DSN=USER4.BACKUP  
//SYSPRINT  DD       SYSOUT=A  
//SYSIN     DD       *  
RESTORE INDD(TAPE) OUTDYNAM(338000) -  
        DS(INCL(HIGH.PERF)) -  
        FORCE          /* TO FORCE RESTORE OF UNMOVABLE DATA SET */  
/*
```

In Example 2, the unmovable data set HIGH.PERF does not currently exist on volume 338000. However, this data set did exist on volume 338000 at the time of the dump. You want to restore it to volume 338000 and would prefer it be restored to the location it originally occupied. However, the location where this data set would normally be restored is occupied by other data sets. So, you restore the data set

to another place on the volume because you specify the FORCE keyword. The data set is marked as unmovable on volume 338000 because of one of the following situations:

- You do not want DFSMSHsm to move it to an unlike device type.
- You do not want the data set to be relocated by a DEFRAg operation for performance reasons.
- It was allocated as an ABSTR data set for performance reasons.

Example 3: logical data set dump, followed by a restore to an unlike device

```
//JOB1      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//TAPE      DD    UNIT=3480,VOL=TAPE01,
// LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=USER2.BACKUP
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
DUMP DATASET(INCL(USER2.OLDDS)) -
      OUTDD(TAPE)
/*

//JOB2      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=A
//TAPE      DD    UNIT=3480,VOL=TAPE01,
// LABEL=(1,SL),DISP=(OLD,KEEP),DSNAME=USER2.BACKUP
//DASD      DD    UNIT=3380,VOL=(PRIVATE,SER=222222),DISP=OLD
//SYSIN     DD    *
RESTORE DATASET( -
      INCLUDE(USER2.OLDDS) ) -
      INDDNAME(TAPE) -
      OUTDDNAME(DASD) -
      RENAME(*.OLDDS,*.NEWDS)
/*
```

In the first part of example 3, DFSMSdss dumps a cataloged data set (USER2.OLDDS) from the source volume to an IBM standard label dump tape (TAPE01). Next, DFSMSdss restores USER2.OLDDS from TAPE01 to a 3380 target volume (DASD volume 222222). The RENAME keyword is used to change the name of the data set to USER2.NEWDS.

Example 4: dump and restore for storage management subsystem (SMS) conversion

```
//JOB1      JOB   accounting information,REGION=nnnnK
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=*
//DTAPE01   DD    DISP=(,CATLG),DSN=V338001.USER3.BACKUP,
// LABEL=(1,SL),UNIT=3480,VOL=SER=TAPE01
//SYSIN     DD    *
DUMP -
      DS( INC(**) ) -
      LOGINDYNAM ( -
      (338001) -
      ) -
      DELETE PURGE COMPRESS -
      OUTDDNAME (DTAPE01)
/*
```

This initial part of [“Example 4: dump and restore for storage management subsystem \(SMS\) conversion”](#) on page 501 dumps all the single-volume data sets on the non-SMS volume 338001 to TAPE01 with the DELETE option. The DELETE and PURGE keywords are required to avoid duplications in the restore operation.

```
//SYSPRINT DD SYSOUT=*
//DTAPE01 DD DISP=(OLD,KEEP),DSN=V338001.USER3.BACKUP,
// LABEL=(1,SL),UNIT=3480,VOL=SER=TAPE01
//SYSIN DD *
RESTORE DS(INC(**)) -
STORCLAS(SC01MJA1) -
INDDNAME (DTAPE01)
/*
```

This second part of “[Example 4: dump and restore for storage management subsystem \(SMS\) conversion](#)” on page 501 restores all of the data sets that were dumped in the first half of this example. Because no output volume is specified, most of the data sets will be allocated on SMS volumes throughout the system. The STORCLAS keyword indicates that the storage administrator wants the data sets to have a storage class of SC01MJA1. The ACS routines might or might not assign the target data sets that the storage class specified. All data sets that are not converted to SMS (ACS STORCLAS routine returns a null storage class) will be restored to the original volume.

Example 5: using the RENAME keyword to restore a VSAM data set

```
//JOB3 JOB accounting information,REGION=nnnnK
//STEP1 EXEC PGM=ADRDSSU
//TAPE DD UNIT=3480,VOL=SER=TAPE04,
// LABEL=(1,SL),DISP=(OLD,KEEP),DSN=USER3.BACKUP
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
RESTORE INDD(TAPE) OUTDYNAM(338000) -
DS(INCL(PARTS.VSAM1)) -
RENAME(*.VSAM1,*.VSAM2) -
CATALOG
/*
```

A VSAM key-sequenced data set, PARTS.VSAM1, is restored from a *logical* dump tape in this example. It is renamed as PARTS.VSAM2 and cataloged in the standard order of search. The cluster’s components, PARTS.VSAM1.DATA and PARTS.VSAM1.INDEX, are also renamed.

Example 6: using the RECATALOG keyword

```
//JOB4 JOB accounting information,REGION=nnnnK
//STEP1 EXEC PGM=ADRDSSU
//TAPE DD UNIT=3480,VOL=SER=TAPE04,
// LABEL=(1,SL),DISP=(OLD,KEEP),DSN=USER3.BACKUP
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
RESTORE INDD(TAPE) OUTDYNAM((338001),(338002)) -
DS(INC(PARTS.**)) /* OR DS(INC(**)) */ -
PCTU(80) -
RECATALOG(USERCAT2) -
TGTTALLOCC(SOURCE)
/*
```

In “[Example 6: using the RECATALOG keyword](#)” on page 502, data sets with a first-level qualifier of *PARTS* were dumped logically. All are restored to volume 338001 and cataloged in catalog USERCAT2. These data sets were on volume 338000 at the time of dump. If the data sets do not fit on volume 338001, a spill volume 338002 is specified. To ensure that the data sets on volume 338001 can be extended, 20% of the total space on volume 338001 is to be left as free space (PCTU(80)). TGTTALLOCC(SOURCE) specifies that the data sets are to be restored with the same allocation type they had when they were dumped.

SPACEREL command for DFSMSdss

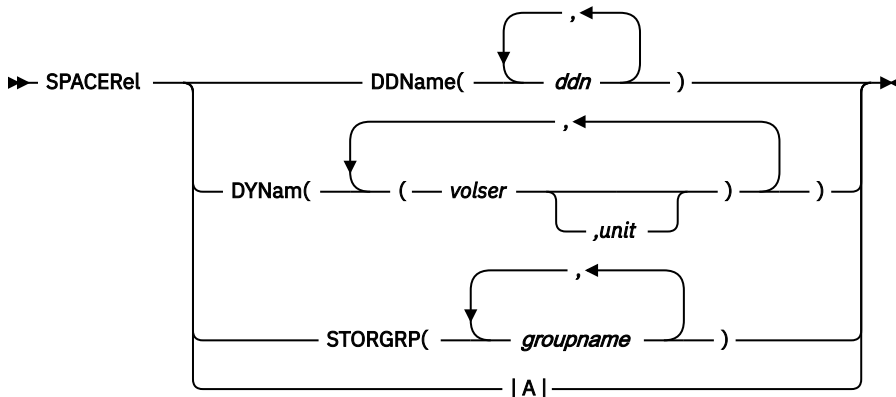
The SPACEREL command determines the free space extents on the extent space efficient (ESE) volumes designated in the DDNAME, DYNAM, or STORGRP parameter and attempts to release the associated physical space to the extent pool.

Using the SPACEREL command might require RACF authorization. If your installation has defined the RACF FACILITY class profile, STGADMIN.ADR.SPACEREL, your user ID requires READ access to the profile. For more information, see [“Protecting the usage of DFSMSdss” on page 535](#).

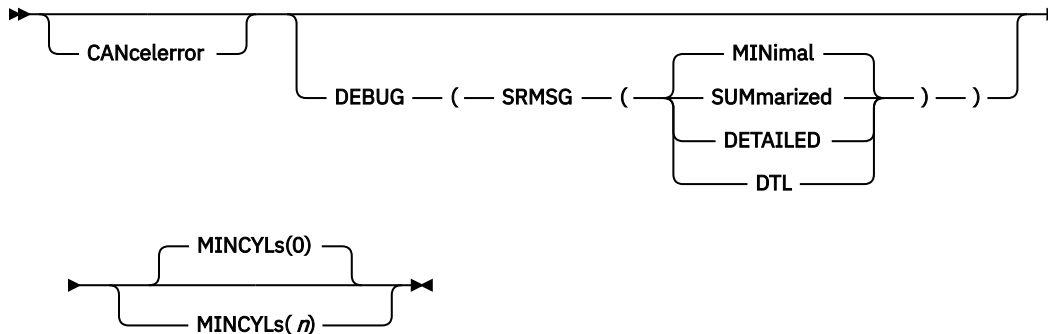
Note: The SPACEREL command supports SMS and non-SMS managed ESE volumes with an indexed VTOC at the volume and storage group levels. Volumes with other VTOC formats are not supported.

DFSMSdss will issue the RESERVE macro to prevent updates to the VTOC to ensure integrity of the VTOC Pack Space Map (VPSM) while processing. If the volume is unavailable for SPACEREL processing, DFSMSdss will wait and retry. For system resources, the default wait time is three seconds and the default retry count is 30. This results in a total wait time of 90 seconds. For information about controlling the wait/retry attempts for system resources, see [“Controlling the wait/retry time for serialization of system resources \(PN11523\)” on page 215](#).

SPACEREL syntax



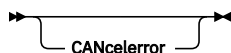
A: Optional keywords for SPACEREL processing



Explanation of SPACEREL command keywords

This section describes the keywords for the SPACEREL command.

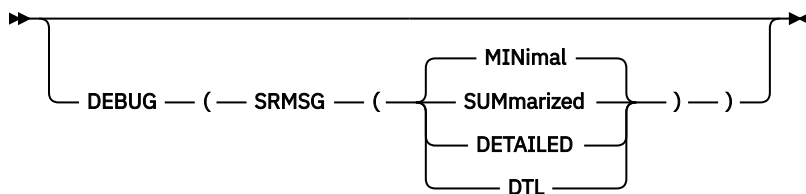
CANCELERROR



CANCELERROR specifies that the SPACEREL operation is to be ended for the volume if a space release I/O related error occurs. If this keyword is not specified and a space release I/O error occurs, the SPACEREL operation for the volume continues with the rest of the extents.

Note: CANCELERROR has no effect on the other types of errors for SPACEREL operation.

DEBUG



You can use DEBUG as a diagnostic tool. The SRMSG subkeyword option designates the level of the progress information that DFSMSdss should provide for SPACEREL operation.

SRMSG(MINIMAL)

Specifies that DFSMSdss is to issue a completion message and no additional information message will be provided. This is the default. The following is an example of messages that are issued when you use this keyword:

Example:

```

ADR006I (001)-STEND(01), 2016.299 15:12:30 EXECUTION BEGINS
ADR523I (001)-SRFP (01), SPACEREL FOR VOLUME SRE00A HAS COMPLETED

```

SRMSG(SUMMARIZED)

Specifies that DFSMSdss is to issue the ADR522I message when SPACEREL will begin to process a volume, and a completion message when the operation has completed. No free space extent information will be provided. The following is an example of messages that are issued when you use this keyword:

Example:

```

ADR006I (001)-STEND(01), 2016.299 15:12:30 EXECUTION BEGINS
ADR522I (001)-SRFP (01), THE FREE SPACE EXTENTS ON VOLUME SRE00A WILL BE PROCESSED BY
SPACEREL
ADR523I (001)-SRFP (01), SPACEREL FOR VOLUME SRE00A HAS COMPLETED

```

SRMSG(DETAILED)

Specifies that DFSMSdss is to issue the ADR522I message when SPACEREL will begin to process a volume, and a completion message when the operation has completed. Free space extent ranges will be listed under ADR522I. The following is an example of messages issued when you use this keyword:

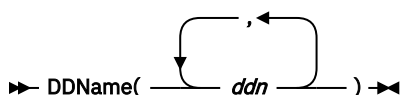
Example:

```

ADR006I (001)-STEND(01), 2016.299 15:12:30 EXECUTION BEGINS
ADR522I (001)-SRFP (01), THE FREE SPACE EXTENTS ON VOLUME SRE00A WILL BE PROCESSED BY
SPACEREL
                SEQUENCE BEGIN C:H - END C:H
                00000001 00000001:1 00000001:5
                00000002 00000001:B 00000040:E
                00000003 00000056:0 00000063:E
                00000004 00000094:0 000000C7:E
                00000005 000000CE:0 00000458:E
ADR523I (001)-SRFP (01), SPACEREL FOR VOLUME SRE00A HAS COMPLETED

```

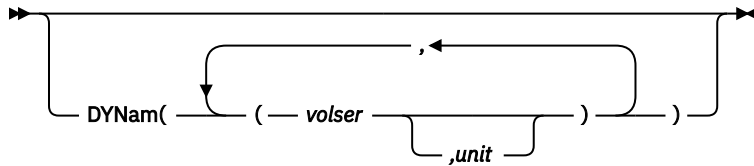
DDNAME



DDNAME requests physical processing.

ddn

Specifies the name of the DD statement that identifies an extent space efficient (ESE) volume whose free space extents are to have its associated physical space released back to the extent pool. To assure correct processing, each of the DD statements corresponding to a DDNAME (*ddn*) must identify only one volume serial number.

DYNAM

DYNAM requests physical process and specifies that the volume to be processed be dynamically allocated. The volume must be mounted and online. You cannot specify a nonspecific volume serial number using an asterisk (*). Using DYNAM instead of DD statements to allocate DASD volumes will not appreciably increase run time and permits easier coding of JCL and command input.

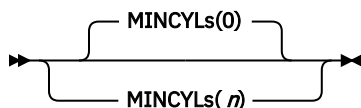
volser

Specifies the volume serial number of an extent space efficient (ESE) volume whose free space extents are to have its associated physical space released back to the extent pool.

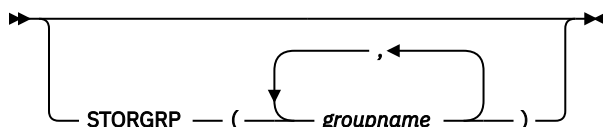
unit

Specifies the device type of a DASD volume to be processed. This parameter is optional.

For additional information regarding storage requirements when processing multiple volumes, see the [“Storage requirements” on page 19](#).

MINCYLs

The MINCYLs keyword specifies that SPACEREL will only be performed on free space extents that are greater than or equal to *n* cylinders. *n* is a 1-to-8 digit decimal number with a range in the range 0 - 99999999. When MINCYLs is not specified, the default is 0 and SPACEREL will be issued for all free space extents.

STORGRP

STORGRP specifies that all of the online extent space efficient (ESE) volumes in the storage group be dynamically allocated. If a volume in the storage group is not online, that volume is not used for processing. You can specify up to 255 storage group names. Specifying STORGRP with a storage group name is equivalent to specifying DDNAME or DYNAM with all the online volumes in the storage group included in the list.

STORGRP cannot be specified at the same time with the DDNAME or DYNAM keyword.

When STORGRP is specified, the SPACEREL command will perform physical volume processing. DFSMSdss will determine the free space extents on each volume designated in the storage groups and attempt to release the physical space associated with the ESE volume back to the extent pool.

Chapter 19. Syntax—auxiliary commands

This section describes the following **auxiliary** commands that you can use to further refine DFSMSdss processing:

- Writing to the Operator:
 - write-to-operator (WTO) command
 - write-to-operator with reply (WTOR) command
- Scheduling Tasks:
 - SERIAL command
 - PARALLEL command
- Controlling Task Processing:
 - SET command
 - IF-THEN-ELSE command sequence
 - EOJ command

Writing to the operator for DFSMSdss

Use either the WTO or the WTOR command to direct a message to the system console. DFSMSdss prefixes the WTO and WTOR messages with a message ID of ADR111I and ADR112A, respectively.

When DFSMSdss encounters either a WTO or WTOR command, it waits until the last-requested function command completes before issuing the WTO or WTOR command.

WTO command

The WTO command lets you write an ADR111I message to the system console that does not request a reply from the operator. The message cannot be greater than 247 characters. You must enclose the message within quotation marks. The command syntax is:

➤ WTO — '*message*' ➤

DFSMSdss assigns the following routing codes to the WTO message:

2

Master console information

11

Programmer information

DFSMSdss assigns the following descriptor code to the WTO message:

6

Job status

WTOR command for DFSMSdss

The WTOR command lets you write an ADR112A message to the system console. The ADR112A message requests that the operator perform some action, and then issue a reply. You can use WTOR, for example, to request that the operator mount a required volume or quiesce a data base before your DFSMSdss job continues to process. The WTOR message cannot be greater than 114 characters. You must enclose the message within quotation marks. The command syntax is:

➤ WTOR — '*message*' ➤

DFSMSDss assigns the following routing code to the WTOR message:

1

Master console action

DFSMSDss assigns the following descriptor code to the WTOR message:

2

Action that is required

Scheduling tasks

You can use the SERIAL or PARALLEL commands to schedule tasks. The SERIAL or PARALLEL command must precede the commands to be executed in the SERIAL or PARALLEL mode.

SERIAL command for DFSMSDss

The SERIAL command lets you re-initiate serial task scheduling (only one task at a time) after you have used parallel task scheduling. Tasks are processed in the order in which they appear in the input stream.

If neither the SERIAL nor PARALLEL command has been issued, SERIAL is the default. The command syntax is:

➤ SERIAL ➤

PARALLEL command for DFSMSDss

The PARALLEL command initiates parallel task scheduling wherein multiple tasks are processed concurrently. When you use parallel processing, commands may not process in the order in which they appear in the input stream. The PARALLEL command is effective only when the required system resources (virtual storage, DASD, or tape volumes) are available. If there are resource conflicts (multiple tasks that use the same DASD or the same tape volume) or if there is not enough virtual storage that is available for all of the tasks to run concurrently, some tasks can be delayed until the resources are available (other tasks end).

DFSMSDss attempts to delay the initiation of additional tasks if there are insufficient available resources to initiate the tasks. DFSMSDss is not able to accurately assess which additional resources that a task might later require. If you attempt to run too many tasks in parallel for the available resources, it can result in unpredictable failures. In such cases, the user must establish a safe upper boundary on the number of tasks that can be supported as parallel tasks.

When you switch between SERIAL and PARALLEL modes, DFSMSDss waits for the completion of all previously scheduled tasks before switching. If DFSMSDss is in PARALLEL mode, an IF statement ensures that all prior commands are processed. The command syntax is:

➤ PARALLEL ➤

Controlling task processing

With the SET, IF-THEN-ELSE, and EOJ commands, you can direct DFSMSDss through a logical path in your command sequence, based on the condition (return) codes of previously completed operations. In addition, you can temporarily set patch bytes with the PATCH parameter of the SET command.

SET command—setting condition codes and patch bytes

As an alternative to setting flags in ADRPATCH (a module in the DFSMSDss load module ADRDSSU), you can customize certain DFSMSDss functions by temporarily setting patch bytes during DFSMSDss processing through a SET PATCH command.

The SET command also allows you to set the LASTCC and MAXCC variables to any value from 0 to 16, inclusive. By doing so, you can influence the logical path that DFSMSdss takes in the command sequence following the SET command.

You can set the following condition codes with the SET command. Use an IF-THEN-ELSE command sequence to test for these condition codes.

0

The function was processed as expected. Informational messages may have been issued.

4

A problem occurred, but processing continued. The result may not be exactly what you wanted, but no permanent harm was done. A warning message was issued.

8

A function did not process, began processing but ended prematurely, or the job ran without processing all requested functions. An error message is issued. If an abend occurs in any of the DFSMSdss subtasks, the return code is set to 8.

12

The job did not process. No functions were processed.

16

A function processed and left at least one volume or data set in an unusable condition. For example, a full-volume dump operation ended prematurely, leaving the output tape in an unusable condition.

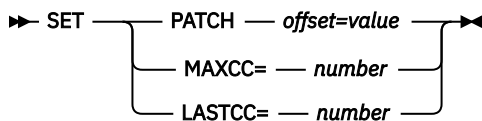
If you are running a batch job, the condition codes that are tested in the IF-THEN-ELSE command sequence, and that can be set by the SET command, cannot be passed from one job step to the next. However, the final maximum condition code is passed to the MVS system when DFSMSdss returns control to the system at the completion of step processing.

You can determine the condition code of the last-requested operation (LASTCC) and the maximum code of all completed operations (MAXCC) with an IF command. When an IF MAXCC or SET MAXCC command is encountered, DFSMSdss waits for all previously requested function commands to complete before the highest return code is determined. Also, when an IF LASTCC or SET LASTCC command is encountered, DFSMSdss waits for the last-requested function command to complete before the return code is determined. After the return code is determined and if the condition code is tested and satisfied, DFSMSdss ends the command or commands following the THEN keyword. If the tested condition is not satisfied, DFSMSdss bypasses the command or commands following the THEN keyword.

Note: Do not use SET LASTCC before the first function command.

SET command for DFSMSdss

The syntax of the SET command is:



The SET command allows you to customize certain DFSMSdss functions by temporarily setting patch bytes. You can also influence the logical path that DFSMSdss takes in the command sequence that follows the SET command. A SET command that occurs within an unprocessed THEN or ELSE clause is not processed.

PATCH

Specifies that DFSMSdss set the patch byte, at offset *offset*, to the value specified with *value*.

Your installation can limit the use of the SET PATCH command with the RACF FACILITY-class profile STGADMIN.ADR.PATCH. You need READ access authorization to that profile to use the SET PATCH command.

offset

Specifies, in hexadecimal, the value for the patch byte offset. The maximum offset that you can specify is X'0FFF', and the minimum offset that you can specify is X'08'.

value

Specifies, in hexadecimal, the value that DFSMSdss assigns to the patch byte at the specified offset. The value must be in the range X'00' to X'FF'.

MAXCC

Specifies that the MAXCC be set to a new condition-code value. Setting MAXCC does not affect the value of LASTCC.

LASTCC

Specifies that the LASTCC be set to a new condition-code value. If the value assigned to LASTCC is higher than the value of MAXCC, MAXCC is also set to the higher value.

number

Specifies the value assigned to MAXCC or LASTCC. The maximum value that can be assigned is 16; a higher value is reduced to 16.

Examples of the SET command

The examples that follow show the use of the SET command.

To set the patch byte at offset X'08' to X'FF', specify the following:

```
SET PATCH 8 = FF
```

To set the patch byte at offset X'44' to X'25', specify the following:

```
SET PATCH 44 = 25
```

To set the last condition code established to 12, specify the following:

```
SET LASTCC=12
```

To replace the highest condition code established in processing so far with 8, specify the following:

```
SET MAXCC=8
```

IF-THEN-ELSE command sequence for DFSMSdss—using condition codes

Condition codes are used to set up the IF-THEN-ELSE statements. LASTCC specifies a comparison to the last condition code and MAXCC specifies the maximum condition code for comparison.

IF-THEN-ELSE command sequence

The syntax of the IF-THEN-ELSE command sequence is:

THEN

Specifies that a single command or a group of commands (enclosed by DO and END) is to be processed if the tested condition is satisfied. THEN can be followed by another IF command.

ELSE

Specifies that a single command or a group of commands (enclosed by DO and END) is to be processed if the tested condition is not satisfied. ELSE can be followed by another IF command. The ELSE clause cannot be on the same line as the THEN clause nor on the same line as the continuation of the THEN clause.

DO

Specifies that the group of commands that follows is to be treated as a single unit, that is, to be processed as a result of a single IF command. The set of commands is ended by END. A command following a DO must begin on a new line.

END

Specifies the end of a set of commands initiated by the nearest unended DO. END must be on a line by itself.

Creating a null command

If THEN or ELSE is not followed by a continuation character or by a command on the same line, the result is a null command. A semicolon after the THEN or ELSE keyword also results in a null command. A null command specifies that no action is taken if the IF clause is satisfied (a null THEN command) or if the IF clause is not satisfied (a null ELSE command).

To specify a null THEN command, specify:

IF ... THEN ELSE ...	or	IF ... THEN; ELSE ...
-------------------------	----	--------------------------

To specify a null ELSE command, specify:

IF ... THEN ... ELSE	or	IF ... THEN ... ELSE;
-------------------------	----	--------------------------

Continuation rules for IF-THEN-ELSE command sequencing

The following continuation rules apply for the IF-THEN-ELSE command sequencing.

1. IF (condition) must be followed by a THEN on the same line or a continuation of that line.

Examples:

```
IF LASTCC = 0 THEN COPYDUMP ...
```

or

```
IF LASTCC = 0 -  
    THEN COPYDUMP ...
```

2. THEN must be followed by a command or DO on the same line or a continuation of that line.

Examples:

```
IF LASTCC = 0 -  
    THEN -  
        COPYDUMP ...
```

or

```
IF LASTCC = 0 -  
    THEN DO  
        COPYDUMP ...  
        PRINT ...  
    END
```

- ELSE must be the first word on a line and no continuation character should be used on the preceding line.

Example:

```
IF LASTCC = 0      -
    THEN
        COPYDUMP ...
    ELSE          -
        PRINT ...
```

Nesting IF commands

An IF command in a THEN or ELSE clause is a nested IF command. The maximum level of nesting is 10, starting with the first time you specify IF.

Within a nest of IF commands, the innermost ELSE clause is associated with the innermost THEN clause, the next innermost ELSE clause with the next innermost THEN clause, and so on. If there is an IF command that does not require an ELSE clause, use a null ELSE clause (see [“Creating a null command”](#) on page 512), unless the nesting structure does not require one. If a nesting structure does not require a null ELSE clause, the DFSMSDss job stream tells you.

Examples of controlling task processing for the IF-THEN-ELSE command

The following are examples of controlling task processing for the IF-THEN-ELSE command.

Example 1

Nested IF commands are used to determine whether a COPYDUMP, EOJ, or PRINT command is to be processed:

```
IF LASTCC > 4      -
    THEN IF MAXCC < 12 -
        THEN COPYDUMP ...
        ELSE EOJ
    ELSE IF LASTCC = 4 -
        THEN
        ELSE PRINT...
```

If the value of LASTCC is greater than 4, the value of MAXCC is to be tested. If the value of MAXCC is less than 12, the COPYDUMP command is processed. Otherwise, the EOJ command is processed. If LASTCC is 4, no action is taken. If LASTCC is less than 4, the PRINT command is processed.

Example 2

Nested IF commands are used to determine whether a COPYDUMP or a PRINT command is to be processed:

```
IF LASTCC > 4      -
    THEN IF MAXCC < 12 -
        THEN COPYDUMP ...
        ELSE
    ELSE IF LASTCC = 4 -
        THEN PRINT ...
```

If the first IF clause finds that LASTCC is greater than 4 and the second IF command finds that MAXCC is 12 or greater, no function command is processed. The null ELSE command specifies that the next ELSE corresponds to the first THEN.

Common continuation errors

The continuation rules, described in [Chapter 15, “Specifying DFSMSDss commands,”](#) on page 245, must be followed carefully when auxiliary commands, comments, or blank records appear in your input. You must also be careful not to inadvertently specify a null clause when continuing auxiliary commands.

The following examples show common continuation errors.

Error example 1

```
IF LASTCC = 0 -  
  THEN  
    PRINT ...
```

A continuation character (hyphen) is missing after *THEN*; consequently, a null THEN clause is assumed. The PRINT command is unconditionally processed.

Error example 2

```
IF LASTCC = 0 -  
  THEN -  
    COPYDUMP ...  
    /* ALTERNATE PATH */  
  ELSE -  
    PRINT ...
```

Because no continuation character (hyphen) follows the comment, a null ELSE clause is assumed. ELSE is not matched with THEN, and an error message is issued. The PRINT command is ignored. Notice the correct use of the continuation character on the other lines.

Error example 3

```
PRINT INDD( - /*COMMENT*/  
  DDN1)
```

The DDN1 on the second line is ignored and nothing is printed because characters other than blanks appear after the continuation character (hyphen).

EOJ command—ending your DFSMSdss step

With the end-of-job (EOJ) command, you can end your DFSMSdss step after the currently processing operation or scheduled tasks are completed. The command syntax is:

➤ EOI ➤

Chapter 20. DFSMSdss Utilities

DFSMSdss stand-alone services

This topic, which describes the Stand-Alone Services function, is intended for the storage administrator, the system programmer, or anyone who runs the Stand-Alone Services program.

You can use Stand-Alone Services to perform either a full-volume or a tracks restore from dump tapes produced by DFSMSdss. The Stand-Alone Services function offers the following benefits:

- Provides user-friendly commands to replace the previous control statements
- Supports all supported IBM Z[®] tape libraries and tape subsystems.
- Supports IPLing from a DASD volume, in addition to tape and card readers
- Allows you to predefine the operator console to be used during Stand-Alone Services processing

Preparing to run the stand-alone services program

The Stand-Alone restore function is a single-purpose program designed to allow the system programmer to restore vital system packs during disaster recovery without needing to rely on a z/OS environment. Stand-Alone Services runs independently from a system environment either as a "true" stand-alone system or under a VM system.

This topic helps you to prepare your environment before you IPL and run the Stand-Alone Services program. Information is provided about running Stand-Alone Services in different processor operating modes, running with a predefined console, setting up tape library and device options, and using command syntax and processing options.

The Stand-Alone Services program operates on an IBM Z[®] processor in z/Architecture mode. The Stand-Alone Services program can run in a z/OS LPAR or in a virtual machine under z/VM in an LPAR with a general purpose machine processor (CP).

The Stand-Alone Services program operates in the z/Architecture architectural mode and requires 2MB of real storage.

Running stand-alone services

The following conditions apply to Stand-Alone Services operations in XA or ESA mode:

- For DASD and tape devices:
 - Stand-Alone Services issues Assign and Unassign commands for tape devices, and Device Reserve and Device Release commands for DASD devices (if the command is supported by the device).
- VM Note:** When running Stand-Alone Services under VM, if the Stand-Alone Services program does not run to completion or is unable to free the device, an Assign or Reserve condition may be left outstanding.

 - The user must ensure that the devices to be used by Stand-Alone Services are not accessed by other systems while Stand-Alone Services IPL and operations are in progress.
- Potential interference from other devices can occur as follows:
 - When the Stand-Alone Services is IPLed and loaded with the operator console not predefined, a Wait PSW is loaded with the rightmost bytes containing X'FFFFFF'. The Stand-Alone Services program waits for the operator to identify the operator console. The first interrupt presented at this time is expected to be a console and is treated as such.
 - When the operator console is predefined and a problem is detected while attempting initial communication with the predefined console, Stand-Alone Services loads a Wait PSW with the

rightmost bytes containing X'DDDDDD', giving the operator an opportunity to identify a console (other than the predefined console) to be used as the operator console.

Other devices can generate interrupts that interfere with Stand-Alone Services operations. If this happens, determine which device is causing the interference. If the interrupting device is not on the same channel path ID (CHPID) as the devices you are using for Stand-Alone Services processing, configure that CHPID offline, re-IPL the Stand-Alone Services program, and configure the CHPID back online after processing is complete. If the interrupting device is on the same CHPID as the devices you are using for Stand-Alone Services processing, follow your installation's procedures to prevent the device from interrupting until after the operator console has been identified.

Note: The operator console or HMC must not have any peripherals attached to it as these may cause unnecessary interrupts.

Running stand-alone services with a predefined console

Use the OPERCNSL keyword of the BUILDSEA command to predefine the console when creating the Stand-Alone Services program. The OPERCNSL keyword allows you to specify the address of the device to be used as the operator console, or you can specify OPERCNSL(SERV) to use an IBM Z® System console.

The device must also be a valid device within the configuration that is running the Stand-Alone Services program. DFSMSdss performs limited validation of the OPERCNSL keyword during the BUILDSEA processing. This is because the Stand-Alone Services program may be run with a system configuration different from the one that is used to build the core image with the BUILDSEA command. (The core image is the executable module that is loaded into the processor's storage during IPL).

After it is IPLed, Stand-Alone Services attempts to use the predefined device as the operator console instead of waiting for the first interrupt to identify the operator console. If a problem with the predefined device is detected, the processor enters a wait-state with the rightmost bytes of the PSW containing "DDDDDD". This PSW indicates that Stand-Alone Services is unable to use the predefined device and is waiting for the operator to identify another console to be used as the operator console. If this happens, do the following:

1. Determine the cause of the problem with the predefined device and take steps to correct the problem. Some of the possible reasons for a problem being detected are listed below.
2. Generate an interrupt on a different console that you can use as the operator console.

The following are some of the reasons for a problem being detected with the predefined console:

- An error has occurred during the Stand-Alone Services program's initial communication with the predefined console.
- The console address may have been incorrectly specified when the Stand-Alone Services program IPL-able core image was built. For example, the address that was specified with the OPERCNSL keyword of the BUILDSEA command does not exist in the configuration in which the Stand-Alone Services program is being IPLed.
- The device at the address specified with the OPERCNSL keyword cannot be identified as a supported operator console.
- If the console was predefined to be the service console, the necessary features may not exist on the processor for Stand-Alone Services to communicate with the console.

Using a tape library

This topic explains how to use supported IBM tape libraries to IPL Stand-Alone Services and restore your dump data set tapes, and how to use the tape library menu options.

Note:

1. Tape drives to be used for Stand-Alone Services must remain offline to other systems.
2. The IPL tape must be mounted and ready prior to performing the IPL.

The Stand-Alone Services RESTORE and TAPECNTL commands are supported by devices within supported IBM tape libraries. Table 27 on page 517 shows the options available when you use a tape library to IPL the core image or restore from dump tapes. Stand-Alone Services can use the tape library in different ways depending on the features that exist on the tape library.

You cannot use the DFSMSdss Stand Alone Restore program with a tape encrypted through an encryption capable tape drive. If you attempt to do so, DFSMSdss issues message ADY3501I to indicate that the dump data set resides on an encrypted tape and thus, cannot be read with the Stand Alone Restore program. DFSMSdss also issues message ADY509D to prompt the operator to continue or end the function.

Similarly, you cannot use the DFSMSdss BUILDSEA command to build a stand-alone image on a hardware encrypted tape. If encryption is to be used in the encryption-capable tape drive, DFSMSdss fails your request with message ADR992E.

IPLing and restoring from a tape library

Use one of the options shown in Table 27 on page 517 to either IPL the Stand-Alone Services program or to restore data from dump data set tapes.

Table 27. Stand-Alone Services Options when Using an IBM Supported Tape Library. Procedures referred to in this table appear later in this section.			
Task ↓	Can You Perform the Task Using an IBM Tape Library with:		
	No "Setup Stand-Alone Device" Feature?	"Setup Stand-Alone Device" Feature Only?	Both "Setup Stand-Alone Device" and "Transient Mount" Features?
IPL the Core image	No.	Only for tapes inside the library. Use Procedure A.	For tapes inside the library, use Procedure A. For tapes outside the library, use Procedure B.
Restore from Dump Tapes	Only for tapes inside the library. Use the TAPEVOLSER keyword of the RESTORE command.	Only for tapes inside the library. Either use Procedure A or use the TAPEVOLSER keyword of the RESTORE command. Only one method can be used for a single invocation of the RESTORE command.	For tapes inside the library, either use Procedure A or use the TAPEVOLSER keyword of the RESTORE command. For tapes outside the library, use Procedure B. Only one method can be used for a single invocation of the RESTORE command.
Note: When the "Setup Stand-Alone Device" feature is used to mount tapes, Stand-Alone Services treats the tape drive as if the drive is not part of a tape library.			

Identifying procedures to mount and demount tapes using the IBM Tape Library Stand-Alone Device setup features

The following procedures to mount and demount the Stand-Alone Services IPL tape and dump data set tapes are referenced within Table 27 on page 517. Use Procedure A for tapes that reside inside the library. Use Procedure B for tapes that reside outside the library.

Note: When both the IPL tape and the dump data set tapes are mounted from the input station (Transient Mount), the same tape drive must be used for both the IPL tape and the dump data set tapes, rather than using different tape drives.

PROCEDURE A

Use this procedure for tapes residing **inside** the library.

Mounting —

Mount tapes using the Library Manager Console "Setup Stand-Alone Device" window.

1. Select the **Mount a single volume** option.
2. Type the device type.
3. Type the volume serial number.
4. Click **Enter**.
5. Repeat steps 1–3 for each tape that you want to mount.

Requirement: Mount the first dump data set tape prior to IPLing the Stand-Alone Services program when separate tape drives are used for the IPL tape and the dump data set tapes.

Demounting —

Demount and unload tapes using the Library Manager Console "Setup Stand-Alone Device" window or use the TAPECNTL command. (These instructions apply only to tapes that are not unloaded by Stand-Alone Services [for example, the IPL tape]).

1. Select the **Demount a single volume** option.
2. Type the device type.
3. Click **Enter**.

PROCEDURE B

Use this procedure for tapes residing **outside** the library.

Mounting —

Mount tapes using the Library Manager Console "Setup Stand-Alone Device" window.

1. Select the **Mount from Input Station** option.
2. Type the device type.
3. Type the volume serial number.
4. Click **Enter**.
5. Repeat steps 1–3 for each tape that you want to mount.

Requirement: Mount the first dump data set tape prior to IPLing the Stand-Alone Services program when separate tape drives are used for the IPL tape and the dump data set tapes.

Demounting —

When the IPL tape is the only tape that is mounted from the input station, unload and demount the tape after the Stand-Alone Services Restore operation has completed. Do this by canceling the mount from the input station using the Library Manager Console.

For additional information about tape library operations, refer to the appropriate IBM Tape Library operator's guide.

Using an automatic cartridge loader

A tape drive with an automatic cartridge loader can be set to manual mode or auto mode. When the loader is set to manual mode, you must manually remove the tape and mount subsequent tapes when Stand-Alone Services has unloaded the tape in the drive and is waiting for a subsequent tape to be mounted.

When the loader is set to auto mode, you can premount the tapes in the order that they will be needed. When Stand-Alone Services unloads the tape in the drive, the cartridges continue to feed and load without requiring intervention.

Controlling command sequence processing

You can control Stand-Alone Services command processing by using SET and IF-THEN-ELSE command sequences. For more information about these commands, see [“Controlling task processing”](#) on page 508.

IPLing and running the Stand-Alone Services Program

This topic contains an overview of the Stand-Alone Services process, the procedure to IPL Stand-Alone Services, and specific information about the RESTORE and TAPECNTL commands.

[Figure 24 on page 519](#) shows an overview of the Stand-Alone Services data restoration process.

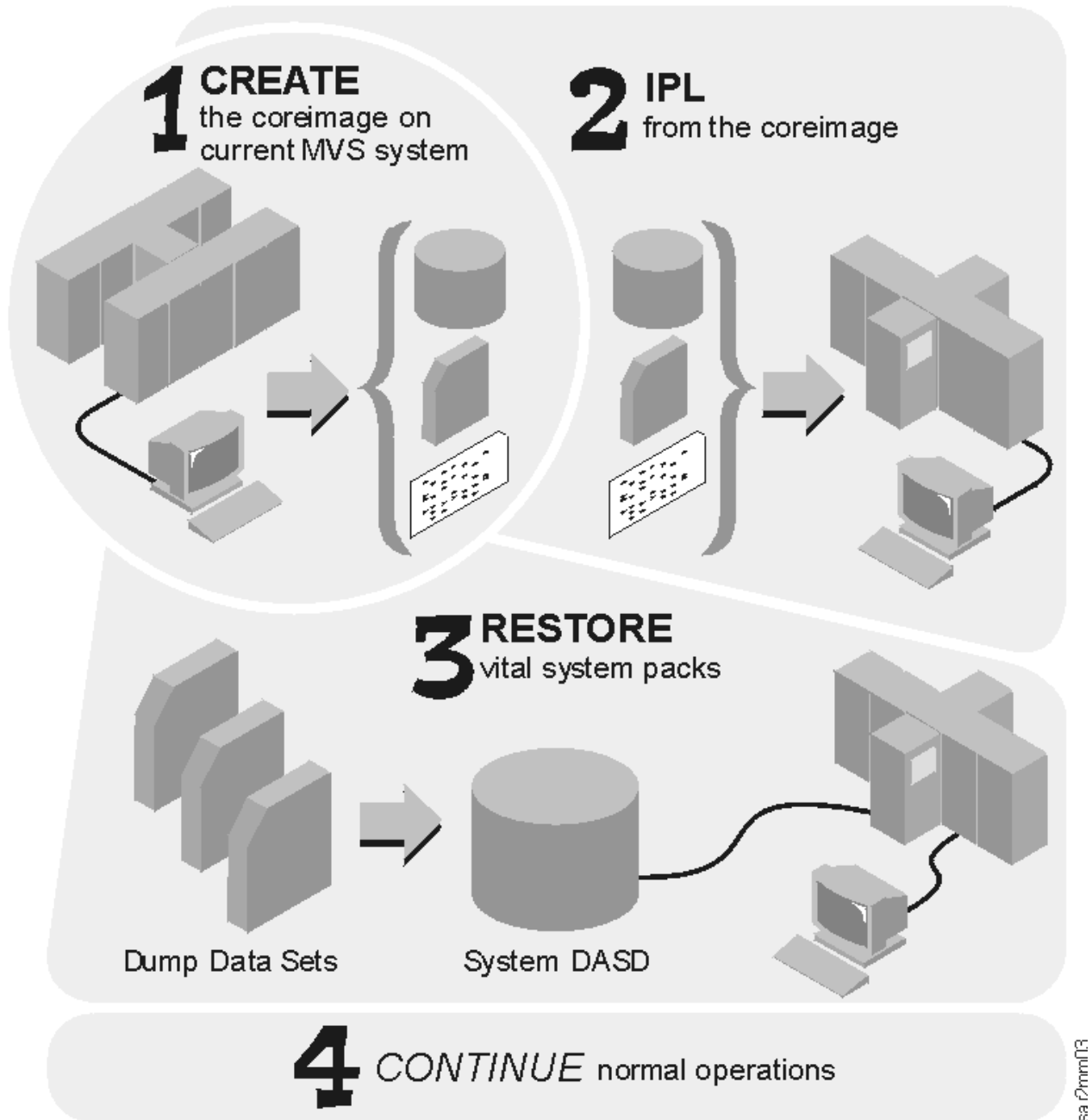


Figure 24. Stand-Alone Services Restore Process Overview

The following is an overview of the steps necessary to implement the Stand-Alone Services program:

1. **Prepare** for Stand-Alone Services by setting up the environment as described in [“Preparing to run the stand-alone services program”](#) on page 515, and creating a Stand-Alone Services IPL-able core image with the BUILDSEA command. The BUILDSEA function is not part of Stand-Alone Services, yet it is necessary before Stand-Alone Services can be IPLed in a stand-alone environment. The BUILDSEA command is described in [“BUILDSEA command for DFSMSdss”](#) on page 269.
2. **IPL** the Stand-Alone Services program from your specified tape, DASD, or card reader device.
3. **Restore** your dumped volumes with the RESTORE command. The RESTORE command performs a full-volume or tracks restore from a DFSMSdss-formatted dump tape.

Use the TAPECNTL command to rewind and unload a tape under Stand-Alone Services control rather than perform this function manually.

IPLing Stand-Alone Services

This topic lists the steps to IPL the Stand-Alone Services program. The programming status word (PSW) wait-state codes (encountered during Stand-Alone Services processing) are found in [“Interpreting wait-state codes”](#) on page 522. An example of the Stand-Alone Services system IPL is included in [“IPL example”](#) on page 521.

To IPL from the Stand-Alone Services core image (created with the BUILDSEA command), proceed as follows:

1. Load the Stand-Alone Services Program.

Load Stand-Alone Services from the processor’s IPL console by specifying the IPL address for the device (card, tape, or DASD) that contains the IPL-able core image and then, by performing a Load Clear operation (also referred to as an IPL Clear).

2. Select the Operator Console.

When the Stand-Alone Services program has finished loading, one of the following conditions exist:

- If the Stand-Alone Services core image was created *without* specifying the OPERCNLS keyword, then the processor enters a wait-state with the rightmost bytes of the PSW containing "FFFFFF". Press the Enter key on the operator console you are using. The first interrupt presented is expected to be a console and is treated as such.
- If the Stand-Alone Services core image was created *with* the OPERCNLS keyword specified, then Stand-Alone Services attempts to use the device address specified by OPERCNLS as the operator console, rather than waiting for the first interrupt. See [“Running stand-alone services with a predefined console”](#) on page 516 for more information.

3. Specify the Input Device.

After the operator console is identified, the following message is displayed:

```
ADRY005E  DEFINE INPUT DEVICE, REPLY 'dddd,ccuu' or 'CONSOLE'
```

To specify the operator console as the input device, enter CONSOLE or a null line. To specify a different device type, enter *dddd,ccuu*, where *dddd* is either the device type or card, and *ccuu* is the unit address. For example, to select a 3505 card reader at address 502, enter:

```
card,502
```

The input device can be one of the supported console devices or a card reader device.

Note: The operator console or HMC must not have any peripherals attached to it as these may cause unnecessary interrupts.

4. Specify the Message Output Device.

After the input device is identified, the following message is displayed:

```
ADRY006E  DEFINE OUTPUT DEVICE, REPLY 'dddd,ccuu' or 'CONSOLE'
```

To specify the operator console as the output device for operator communication, enter `CONSOLE` or a null line. To specify a different device type, enter `dddd,ccuu`, where `dddd` is either the device type or `print`, and `ccuu` is the unit address. For example, to select a 3800 print subsystem at address 510, enter:

```
print,510
```

The output device can be a supported console device or a supported printer device.

5. **Specify the correct date and time, if needed.**

Stand-Alone Services automatically picks up the time and date from the processor's time-of-day (TOD) clock, usually in the Greenwich Mean Time (GMT) format. When the TOD clock is incorrect or is not set, the following message is displayed:

```
ADRY015E  SUPPLY TODAY'S DATE, REPLY 'MM/DD/YY'
```

When you have entered the correct date in the format indicated, the following message is displayed:

```
ADRY016E  SUPPLY TIME OF DAY, REPLY 'HH:MM:SS'
```

Enter the correct time in the format indicated. If you press `Enter` without specifying a date or time, the value is set to zero.

At this point the IPL is complete. You can now enter Stand-Alone Services commands from the specified input device. Multiple commands can be entered without requiring a re-IPL of the Stand-Alone Services program.

Note: Operator messages such as `ADRY003D` are sent to the operator console, not the specified output device.

IPL example

In the following example, the operator console is defined as both the input device and output device. System messages are highlighted in **bold** followed by the user's response. The example shows how more than one command can be entered without requiring a re-IPL.

```

ADRY005E DEFINE INPUT DEVICE, REPLY 'DDDD,CCUU' OR 'CONSOLE'
ENTER INPUT/COMMAND:

{The Enter key is pressed}

ADRY006E DEFINE OUTPUT DEVICE, REPLY 'DDDD,CCUU' OR 'CONSOLE'
ENTER INPUT/COMMAND:

{The Enter key is pressed}

SA/XA/ESA 5650-Z0S DFSMSDSS STAND-ALONE V2.01.0
TIME: 16:36:23 08/19/13

ENTER INPUT/COMMAND:

restore frmdv(tape) frmadr(faf) toadr(f4a) vfy(tstb04)

RESTORE FRMDV(TAPE) FRMADR(FAF) TOADR(F4A) VFY(TSTB04)
ADRY0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
16:38:05 08/19/13
ENTER INPUT/COMMAND:

restore frmdv(tape) frmadr(faf) toadr(f4a) -

RESTORE FRMDV(TAPE) FRMADR(FAF) TOADR(F4A) -
ENTER INPUT/COMMAND:

vfy(tstb04)

VFY(TSTB04)
ADRY0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
16:39:48 08/19/13
ENTER INPUT/COMMAND:

```

Interpreting wait-state codes

DFSMSDss stores a code in the six right-most bytes of the programming status word (PSW). The following wait-state codes can occur during the Stand-Alone Services IPL:

Code

Explanation

000033

A program check occurred while Stand-Alone Services was being loaded. Contact your software service representative.

000044

The IPL device is not operational. Contact your hardware service representative. If the IPL device is a tape drive, try to IPL from another tape drive.

000055

An I/O error occurred on the IPL device or channel while Stand-Alone Services was being loaded. Contact your hardware service representative to correct the cause of the problem.

000066

The IPL loader is unable to determine if the entire Stand-Alone Services core image has been loaded. This could be due to a software or hardware error. Determine the cause of the problem and contact the appropriate service representative.

000077

The IPL loader is unable to locate the SYS1.ADR.SAIPLD.Vvolser data set on the IPL volume.

000088

The device type being used to IPL is not a supported DASD device for IPLing the Stand-Alone Services.

0000AA

An attempt was made to IPL a processor that could not switch to zArchitecture. zArchitecture is required on this release of Stand-Alone Services.

0000AE

An external interrupt occurred while Stand-Alone Services was being loaded. Contact your software service representative.

0000AF

A supervisor call instruction (SVC) interrupt occurred while Stand-Alone Services was being loaded. Contact your software service representative.

0000E2

A machine check has occurred. Contact your hardware service representative.

111111

Stand-Alone Services is waiting for an I/O interrupt. If the processor stops with this code loaded it may be necessary to re-IPL and rerun the command. Contact your hardware service representative if the problem persists.

888888

A temporary wait for a service-signal interrupt.

999999

A temporary wait for a service-signal interrupt.

BBBBBB

Stand-Alone Services is waiting for the operator to enter the input in response to a prompting message on the service console.

DDDDDD

Stand-Alone Services has detected an error related to the predefined console, and is waiting for the operator to identify another console to be used as the operator console. Possible reasons for the error with the predefined console and actions to take are listed in [“Running stand-alone services with a predefined console” on page 516](#).

EE4990

Stand-Alone Services cannot find a required module. Refer to message ADRY4990I for more information.

EEEEnn

The processor is in a wait-state. The error is indicated by *nn* as follows:

nn

Indicates:

13

An SVC interrupt has occurred. Run the SADMP* service aid to dump the contents of real storage to tape, and contact your software service representative.

14

A program interrupt has occurred. Run the SADMP* service aid to dump the contents of real storage to tape, and contact your software service representative.

15

There is insufficient main storage. Stand-Alone Services requires 2MB of storage.

16

An I/O error has occurred.

17

Stand-Alone Services is unable to open a data set or access a device, possibly because the device type is not supported.

18

Stand-Alone Services cannot send an operator message because the console is either not defined or is unavailable.

19

An end-of-data routine is missing. Run the SADMP* service aid to dump the contents of real storage to tape, and contact your software service representative.

1A

The predefined console is not attached or is not operational. Possible reasons for this error are listed in [“Running stand-alone services with a predefined console” on page 516](#).

1B

Stand-Alone Services is unable to communicate with the predefined service console. This may be due to an error, or because the necessary features do not exist on the processor to communicate with the predefined console.

Note: See *z/OS MVS Diagnosis: Tools and Service Aids* for information on creating a stand-alone dump with the AMDSADMP (SADMP) service aid.

EECC03

A condition code 3 (not operational) has been received when attempting to communicate with the service console. Contact your hardware service representative.

EECCCC

An error occurred while trying to communicate with the service console. This can result from a hardware or a software problem.

F1F1F1

Waiting for an I/O interrupt while Stand-Alone Services is being loaded. This may also indicate a hardware problem with the device you are IPL'ing from.

FFFFFF

Stand-Alone Services is waiting for the operator to identify the operator console. Generate an interrupt from the console that is to be used as the operator console.

RESTORE—restoring a formatted dump tape

Use the RESTORE command to perform either a full-volume or a tracks restore from dump tapes that are produced by DFSMSdss without the use of a system environment.

The RESTORE command can restore from tape volumes that are created by a full-volume or tracks dump, or it can restore a track or tracks from the first logical volume of a physical data set dump. It cannot restore from tapes that are created by a DFSMSdss logical dump, from a DFSMSdss tracks dump using the CPVOLUME keyword, or from dump tapes produced by other utilities. It cannot restore from dumps created with the ENCRYPT, HWCOMPRESS, or ZCOMPRESS keywords because the facilities required to process encrypted or compressed data do not exist in the stand-alone environment.

With the RESTORE command, you specify both the source tape volume (containing the dump data set) and the DASD target volume. Use either IBM standard label or nonlabeled tapes as the source tape volumes (dump tapes) for the Stand-Alone Services restore operation. If the volume serial number of the DASD target volume is different from the volume serial number of the original DASD source volume, the restore operation changes the DASD target volume serial number to that of the DASD source volume.

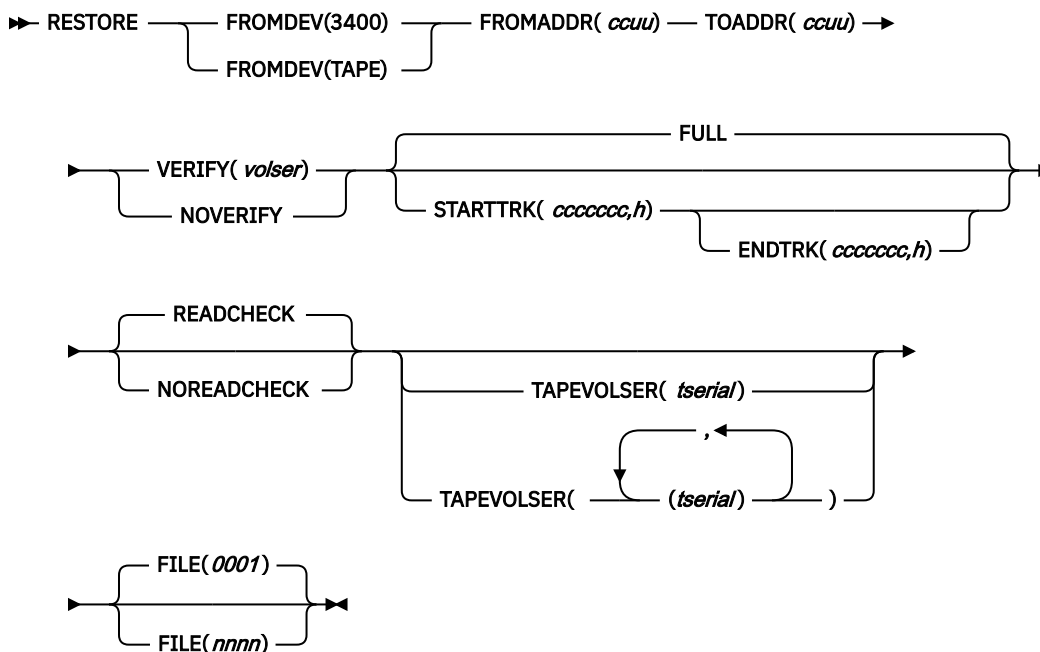
When IPLing from tape, the data to be restored can be mounted on a tape drive other than the IPL tape drive. Alternatively, a single tape drive can be used to mount the tape to be IPLed and the tape to be restored.

The device type of the DASD source volume used for the system dump must match the device type of the receiving volume used in a Stand-Alone Services restore operation. However, dump data from a smaller capacity (fewer cylinders) device can be restored to a larger capacity (more cylinders) device of the same device type.

Note: When data is restored from a smaller capacity device to a larger capacity device, the free space information becomes invalid. The free space information in the VTOC is rebuilt when the next data set is allocated on the volume.

RESTORE command syntax

The syntax of the Stand-Alone RESTORE command is:



See [“Command syntax” on page 245](#) for command and comment formatting specifics

Required parameters

FROMDEV

Specifies the device type that the dump data set resides on. Eligible device names are 3400 and TAPE. When either 3400 or TAPE is specified, Stand-Alone Services attempts to determine the device type from the self-description information. If the device self-description information indicates a supported device type, Stand-Alone Services uses the returned device type for processing. When a device does not support self-description, Stand-Alone Services processes the device differently depending on whether 3400 or TAPE is specified. Abbreviations: FRMDEV and FRMDV.

3400

specifies device types 3420, 3422, and 3430. If the device type cannot be determined from the self-description information, Stand-Alone Services processes the device as a 3400-type device.

TAPE

specifies all other supported tape devices. If the device type cannot be determined from the self-description information, Stand-Alone Services processing ends.

FROMADDR

Specifies the address of the device that the dump data set resides on. You can specify a 3-digit or 4-digit address. Abbreviations: FRMADDR and FRMADR.

TOADDR

Specifies the address of the DASD target device to be restored. The device type must be the same as the device type of the volume originally dumped. You can specify a 3-digit or 4-digit address. Abbreviation: TOADR.

VERIFY(volser)

Specifies that the volume serial number that is currently on the DASD target volume must be verified before restoring the data. Specify either VERIFY or NOVERIFY, not both.

Initialize DASD volumes with a readable volume label and VTOC before restoration. Abbreviation: VFY.

NOVERIFY

Specifies that no action be taken to either verify the volume serial number, or to verify that any volume serial number exists. When NOVERIFY is specified, a prompting message is issued for the user to reply with permission to continue. Specify either VERIFY or NOVERIFY, not both. Abbreviations: NOVIFY and NVFY.

Optional keywords**FULL**

Specifies that the full volume be restored. The dump data set must be a DFSMSdss full-volume physical dump. The default is FULL.

STARTTRK

Specifies that the designated range of tracks be restored. The dump data set can be a full-volume physical dump, a tracks dump, or a physical data set dump. Specify the STARTTRK keyword with a tracks dump or a physical data set dump. STARTTRK is not valid when FULL is specified.

Specify the track in the form *(ccccccc,h)*, where *ccccccc* is the cylinder number and *h* is the head number. You can specify the cylinder and head numbers in either decimal or hexadecimal (for example, X'AC', X'E' for hexadecimal or 172, 14 for decimal). Leading zeros are not required. The STARTTRK keyword is processed as follows:

- When the ENDTRK keyword is not specified, the ending track is set to the last track on the volume.
- When the starting track value is higher than the ending track value, an error message is issued and the restore operation ends.
- When the starting track value exceeds the volume limits, an error message is issued and the restore operation ends.

Abbreviations: STRTRK and STRK.

ENDTRK

Specifies the ending track to be restored when STARTTRK is specified. This keyword is not valid when FULL is specified.

Specify the track in the form *(ccccccc,h)*, where *ccccccc* is the cylinder number and *h* is the head number. You can specify the cylinder and head numbers in either decimal or hexadecimal (for example, X'AC', X'E' for hexadecimal or 172, 14 for decimal). Leading zeros are not required. The ENDTRK keyword is processed as follows:

- When the STARTTRK keyword is specified and the ENDTRK keyword is not specified, the ending track is set to the last track on the volume.
- When the ending cylinder value exceeds the volume limits, the ending cylinder value is set to the last cylinder on the volume, and a warning message is issued.
- When the ending head value exceeds the volume limits (exceeds the last head in a cylinder), the ending head value is set to the last head on the ending cylinder. In addition, a warning message is issued.
- When the starting track value is higher than the ending track value, an error message is issued and the restore operation is ended.

Abbreviation: ETRK.

READCHECK

Specifies that a read-back check of the restored data be performed. READCHECK is the default. Abbreviations: READCHK, RDCHECK, RDCHK, and READ.

NOREADCHECK

Specifies that a read-back check of the restored data *not* be performed. Abbreviations: NOREADCHK, NREADCHK, and NREAD.

TAPEVOLSER (*tserial*)

Specifies the tape volume serial numbers of the tapes to be mounted by Stand-Alone Services when the tapes are in an IBM tape library. Volumes are mounted by Stand-Alone Services in the order in which they are specified.

The maximum number of tape volume serial numbers that can be specified is 32. All of the specified tape volumes must be part of the same dump data set that is used for the restore.

The TAPEVOLSER keyword is ignored if the FROMDEV keyword does not specify a valid tape drive in a tape library.

Do not specify the TAPEVOLSER keyword when the tapes are mounted from the Library Manager Console with the Setup Stand-Alone Device Pop-up Window. See [“Using a tape library”](#) on page 516 for information about tape libraries. Abbreviations: TAPEVOL and TPVOL.

FILE (*nnnn*)

Specifies the relative position, from the beginning of the tape volume, where the dump data set begins. Allowable values are from 1 to 9999. When the FILE keyword is not specified, the default value is 1. Leading zeros are not required. If the specified file number does not exist on the tape, unpredictable results can occur. For example, on 3400 tape devices with tape reels, an incorrectly specified file number can cause the tape to run off the end of the reel.

Note: When restoring from a multi-volume /multi-file data set, the FILE number for data sets on other than the 1st volume being restored, is the physical file number on that volume.

Example: File sequence on volume 2 begins at sequence 60. If restoring file sequence 60, then FILE(1) will need to be coded since the catalog or TMS is not available in the Stand-alone environment.

RESTORE command examples

In the following example, device address 2FAF is a 3490 tape drive with a tape mounted that contains the dump data set to be restored. Device address 4791 is a 3380 DASD with volume serial number D3380K. Before writing on the volume, the RESTORE command verifies that the DASD at address 4791 has volume serial number D3380K. The full volume is restored.

```
RESTORE FRMDV(TAPE) FRMADR(2FAF) TOADR(4791) VFY(D3380K)
```

In the following example, device address F01 is a 3420 tape drive with a tape mounted that contains the dump data set to be restored. Device address 9B9 is a 9345 DASD with volume serial number TS9345. Before writing on the volume, the RESTORE command verifies that the DASD at address 9B9 has volume serial number TS9345. The range of tracks to be restored is cylinder 0, head 0 through the end of the volume.

```
RESTORE FRMDV(3400) FRMADR(F01) TOADR(9B9) VFY(TS9345) STRK(0,0)
```

In the following example, device address F77 is a 3480 tape drive with a tape mounted that contains the dump data set to be restored on file 3 of the tape. Device address F4A is a 3390 DASD. The NOVERIFY keyword prompts the operator for permission to write on the device at address F4A.

```
RESTORE FRMDV(TAPE) FRMADR(F77) TOADR(F4A) NVFY FILE(3)
```

In the following two examples, device address F77 is a 3480 tape drive with a tape mounted that contains the dump data set to be restored. Device address 9B9 is a 9345 DASD with volume serial number TS9345. Before writing on the volume, the RESTORE command verifies that the DASD at address 9B9 has volume serial number TS9345. The range of tracks to be restored is from cylinder 200, head 5 through cylinder 205, head 14. The first example specifies the tracks in decimal:

```
RESTORE FRMDV(TAPE) FRMADR(F77) TOADR(9B9) VFY(TS9345) -  
      STRK(200,5) ETRK(205,14)
```

The second example specifies the tracks in hexadecimal:

```
RESTORE FRMDV(TAPE) FRMADR(F77) TOADR(9B9) VFY(TS9345) -
STRK(X'C8',X'5') ETRK(X'CD',X'E')
```

In the following example, device address FDD is a tape drive in a 3495 Tape Library, and the tape volume with volume serial number BCD103 contains the dump data set to be restored. Device address F4A is a 3390 DASD. The NOVERIFY keyword of the RESTORE command prompts the operator for permission to write on the device at address F4A. Stand-Alone Services mounts the tape volume with volume serial number BCD103 on the tape drive with address FDD. The range of tracks to be restored is cylinder 0, head 0 through cylinder 5, head 5.

```
RESTORE FRMDV(TAPE) FRMADR(FDD) TOADR(F4A) NVFY -
TAPEVOL(BCD103) STRK(0,0) ETRK(5,5)
```

In the following example, device address FDD is a tape drive in a 3495 Tape Library, and the tape volumes with volume serial numbers BCD101 and BCD102 contain the dump data set to be restored. Volume BCD101 is the first volume in the sequence, and BCD102 is the second volume. Device address 791 is a 3380 DASD. The NOVERIFY keyword of the RESTORE command prompts the operator for permission to write on the device at address 791. Stand-Alone Services mounts the tape volumes on the tape drive with address FDD. Volume BCD101 is mounted first, and when the end of the tape is reached volume BCD102 is mounted.

```
RESTORE FRMDV(TAPE) FRMADR(FDD) TOADR(791) NVFY -
TAPEVOL((BCD101) (BCD102))
```

TAPECNTL—rewinding and unloading a tape

Use the TAPECNTL command to either rewind, or rewind and unload a tape.

Stand-Alone Services gives you the ability (with the REWIND and UNLOAD keywords) to rewind and unload tapes under Stand-Alone Services control rather than doing so manually.

The TAPECNTL command supports devices that are part of supported IBM Tape Libraries.

TAPECNTL command syntax

The syntax of the TAPECNTL command is:

```
➡ TAPECNTL — DEVTYPE(3400) — UNITADDR(ccuu) — REWIND —
              — DEVTYPE(TAPE) — UNLOAD — ➡
```

See [“Command syntax” on page 245](#) for command and comment formatting specifics.

Required keywords

DEVTYPE

Specifies the tape device type that the TAPECNTL operation is to be performed against. Eligible device names are 3400 and TAPE. When either 3400 or TAPE is specified, Stand-Alone Services attempts to determine the device type from the self-description information. If the device self-description information indicates a supported device type, Stand-Alone Services uses the returned device type for processing. When a device does not support self-description, Stand-Alone Services processes the device differently depending on whether 3400 or TAPE is specified. Abbreviation: DEV.

3400

specifies device types 3420, 3422, and 3430. If the device does not support self-description, Stand-Alone Services may not be able to determine if the device is a tape drive. Stand-Alone Services issues the rewind or unload instruction to the device anyway. When the device is not a tape drive, unpredictable results may occur.

TAPE

specifies all other supported tape devices. Stand-Alone Services does not issue rewind or unload instructions to the device when Stand-Alone Services cannot determine the device type from the self-description.

UNITADDR (ccuu)

Specifies the unit address of the device against which the TAPECNTL operation is performed. You can specify a 3-digit or 4-digit address. Abbreviations: UNIT and ADDR.

REWIND

Specifies a rewind operation. Specify either the REWIND keyword or the UNLOAD keyword, not both. Abbreviation: REW.

UNLOAD

Specifies that a rewind *and* unload operation be performed. If the tape drive is in an IBM tape library, the tape is demounted if necessary. Abbreviation: UNL.

TAPECNTL command examples

In the following example, the TAPECNTL command rewinds the tape mounted in the 3490 tape drive at address 2FAF:

```
TAPECNTL DEV(TAPE) UNIT(2FAF) REW
```

In the following example, the TAPECNTL command rewinds and unloads the tape mounted in the 3480 tape drive at address F77:

```
TAPECNTL DEV(TAPE) UNIT(F77) UNL
```

In the following example, the TAPECNTL command rewinds and unloads the tape mounted in the 3420 tape drive at address F01:

```
TAPECNTL DEV(3400) UNIT(F01) UNL
```

Building the IPL-able core image

This topic describes how to use the BUILDSEA command to build the Stand-Alone Services IPL-able core image. The core image is the executable module that is loaded into the processor's storage during the IPL and load process.

BUILDSEA function

The BUILDSEA function is not part of Stand-Alone Services, yet it is necessary before Stand-Alone Services can be IPLed in a stand-alone environment. The DFSMSdss BUILDSEA command and examples are presented in [“BUILDSEA command for DFSMSdss” on page 269](#).

Understanding BUILDSEA command authorization levels

Your ability to use the BUILDSEA command is determined by your level of access authorization to source (input) data sets and target (output) data sets or volumes used by the BUILDSEA operation. Storage administrators with any of the following access levels may use the BUILDSEA command:

- Without special authorization
- DASDVOL-access authority to a volume
- DFSMSdss authorization

Using BUILDSEA without special authorization

To use the BUILDSEA command, you must have the following data-set-level authorization to build the IPL-able core image for card or tape:

- READ access to the input data sets used by the SYS1.SADRYLIB target library
- UPDATE access to the output card or tape data sets

Without special authorization, you cannot create an IPL-able core image for DASD. Even though you may have UPDATE or ALTER access to the output SYS1.ADR.SAIPLD.Vvolser data set, you are not authorized to update cylinder 0 head 0 of the DASD volume.

Using BUILDSEA with DASDVOL-access authorization

To use the BUILDSEA command for IPL(DASD), you must have the following access:

- READ access to the input data set (SYS1.SADRYLIB)
- UPDATE access at the DASDVOL level for each DASD volume on which you create an IPL-able core image

Using BUILDSEA with the ADMINISTRATOR keyword

Instead of having DASDVOL update access to volumes for IPL(DASD), you can act as a DFSMSdss-authorized storage administrator for the BUILDSEA command by specifying the ADMINISTRATOR keyword.

The FACILITY-class profile STGADMIN . ADR . STGADMIN . BUILDSEA for the BUILDSEA command ADMINISTRATOR keyword lets you build the Stand-Alone Services IPL-able core image without having UPDATE access to the output data sets or UPDATE access to a DASD volume when you create a core image for IPLing from DASD. You must still have access at the data set level for the input data set.

See the ADMINISTRATOR keyword in [“Explanation of BUILDSEA command keywords” on page 270](#) for more information.

DFSMSdss Compression Tool

This topic, which describes the DFSMSdss compression tool, is intended for anyone who wishes to determine if their volume data would benefit from compression.

The compression tool is a utility that:

- Invokes DFSMSdss.
- Establishes zEnterprise Data Compression (zEDC) services capability*.
- Obtains information from the source volume.
- Compresses volume tracks using zEDC in 16K Block increments.
 - Keeps an average input track size
 - Keeps an average output track size (post compression)
- Issues volume free space statistics and an overall average compression savings.

Note:

1. This tool requires the use of zEDC compression services. For further information on requirements for using zEDC, see: [Requirements for zEnterprise Data Compression in z/OS MVS Programming: Callable Services for High-Level Languages](#)

Control

The following utility control statements are required:

Statement	Use
JOB	Initiates the job
EXEC	Specifies the program name (PGM=ADRVCMPT)

Statement	Use
SYSPRINT DD	Defines a sequential message data set. The data set can be written to a system output device, a tape volume, or a direct-access device.
SAMPLEDD DD	Defines the input (also called the source) volume.
<i>output</i> DD	Defines the output. The ddname, <i>output</i> , should be defined as a DUMMY DD.
SYSIN DD	Defines a command data set containing a DFSMSdss command. It usually resides in the input stream, however, it can be defined as a blocked or unblocked sequential data set or as a member of a partitioned data set. Records must be fixed format, LRECL=80 The SYSIN should describe a DFSMSdss DUMP FULL operation.
PARM statement (required)	The EXEC statement should contain a PARM field with the following: 'ACTION=16KZEDC' - Indicates that the tool should use zEnterprise Data Compression services in 16K block increments.
Parm statement (optional)	The EXEC statement may contain a PARM field with the following: 'OPTION=CSVOUT' - Indicates that the tool should additionally output a line capturing the statistics and compression results as CSV.

If the above control statements are not as describes - unexpected behavior may occur, if not otherwise captured and identified via an error message.

Example of invoking the compression tool

The following example shows how to invoke the compression tool on volume VOL001

```
//MYJOB    JOB      accounting information,REGION=nnM
//STEP1    EXEC     PGM=ADRVCMPT,PARM='ACTION=16KZEDC'
//SYSPRINT DD      SYSOUT=A
//SAMPLEDD DD      UNIT=3390,VOL=SER=VOL001,DISP=OLD
//OUTPUTDD DD      DUMMY
//SYSIN     DD      *
DUMP FULL INDD(SAMPLEDD) OUTDD(OUTPUTDD)
/*
```


Chapter 21. Data security and authorization checking

This section describes the data security protection and access authorization checks done by DFSMSdss. The DFSMSdss functions available to a user depend on the access authorizations as defined by:

User and group profiles

Define the authorized users of a RACF-protected system. The user or group identifier (ID) used for access-authority checking must be defined to RACF.

Data set and general resource profiles

Protect the resources in a RACF-protected system and identify the access levels that users have to those resources.

z/OS UNIX files

DFSMSdss does not invoke SAF/RACF during z/OS UNIX files access and authorization checking. Instead, DFSMSdss depends on z/OS UNIX System Services to perform file-access checking on our behalf. File access (including privileged user) checking is performed using the identity of the invoker as defined by RACF. Refer to the z/OS UNIX System Services User's Guide under the topic 'Handling security for your files' for more information on this topic.

DFSMSdss supports data-security protection and access-authorization checking through the system authorization facility (SAF), Resource Access Control Facility (RACF), and system services such as catalog management services and allocation. The phrase "RACF-protected" implies that SAF and RACF (or equivalent) are installed and active.

DFSMSdss uses the SAF interface and checks to ensure that RACF at the 1.8.1 level or later is installed and active. If an equivalent to RACF is used, either it must set the same level information that DFSMSdss checks, or your installation must use the DFSMSdss installation options exit to tell DFSMSdss that SAF with a RACF equivalent is installed. The proper level of RACF must be installed for the data-security features to work as described. The primary features and their levels of RACF are listed below:

Feature	Required Level of RACF
Generic profile handling:	RACF 1.5 or later
DASDVOL access authority:	RACF 1.6 or later
FACILITY class authority:	RACF 1.7 or later
Erase-on-scratch:	RACF 1.7 or later
Group data set creation:	RACF 1.8.1 or later
Storage class and management class:	RACF 1.8.1 or later

Related reading: For information about the installation options exit, see *z/OS DFSMS Installation Exits*. For information about data security and RACF, see *z/OS Security Server RACF Security Administrator's Guide*.

Effects of SPECIAL, OPERATIONS, and DASDVOL

The SPECIAL and OPERATIONS attributes and DASDVOL-access authority can affect the results when you use DFSMSdss to access and process data sets.

SPECIAL

When you use DFSMSdss, the SPECIAL attribute does not give you the authority to define or rename discrete profiles during a copy, move, or restore of user data sets that you do not own. This is because DFSMSdss uses the DEFINE function of SAF and RACF and the ALTER function of catalog management services to define and rename discrete profiles on your behalf. Instead of the system-SPECIAL attribute,

you need the system-OPERATIONS attribute to define or rename discrete profiles for user data sets that you do not own.

OPERATIONS

If you have the system-OPERATIONS attribute, you have authority to the protected resources in resource classes such as DATASET, DASDVOL, and TAPEVOL. You are limited to the access specified in the access list if either:

- Your current connect group (or any connect group when list-of-groups checking is active) is in the access list of a resource profile.
- Your user ID is in the access list.

As a user with the system-OPERATIONS attribute, you have full control over data sets, and you can do the following:

- Copy, reorganize, catalog, and scratch (delete) data sets
- Perform input and output operations on protected tape volumes
- Define profiles for group data sets to which you are not connected
- Create user data sets and data sets in groups to which you are not connected. However, if your ID is in the access list, you need at least UPDATE access.

You need the system-OPERATIONS attribute to define or rename discrete profiles when you use DFSMSdss, even if you are acting as a DFSMSdss-authorized storage administrator (see [“DFSMSdss storage administrator”](#) on page 542). DFSMSdss can define or rename discrete profiles during copy or restore operations. However, the OPERATIONS attribute does not necessarily let you perform all DFSMSdss functions:

- If you have group-OPERATIONS, your authority is restricted to the resources within the scope of the group.
- The data set to be processed may be a data set over which you have no authority. For example, your user ID only has READ access to a data set that you want to copy and delete.
- A copy of a group data set may be disallowed because you are connected to the group and thus not authorized to define a discrete profile for the data set.
- Access can be denied due to security-level, security-category, or security-label checking.

DASDVOL

DASDVOL is supported directly for volume-level operations, physical operations for both SMS-managed and non-SMS-managed data sets, and logical operations for non-SMS-managed data sets. With DASDVOL-access authority to a volume, you can perform DFSMSdss operations as described in [“Volume access and DASDVOL”](#) on page 544 and [“DASDVOL limitations”](#) on page 546. When you do so, DFSMSdss bypasses access checking to the data sets and catalogs on that volume.

You have DASDVOL-access authority to one or more DASD volumes if all of the following are true:

- DASDVOL class is active.
- Profiles for the DASD volumes are defined in the DASDVOL class.
- You have the level of access needed for the function you are trying to perform.

Instead of DASDVOL-access authority, your installation can authorize you to act as a storage administrator for one or more DFSMSdss operations. See [“DFSMSdss storage administrator”](#) on page 542 for details.

Note: The system programmer or storage administrator uses ICKDSF to format tracks, write a volume label, and create a VTOC. The system programmer or storage administrator needs to have RACF DASDVOL authority to do that. No one needs DASDVOL authority to allocate space on volumes. The system controls space on SMS volumes by other means such as ACS routines, storage group definitions, and Interactive Storage Management Facility (ISMF) commands.

General data security information

This section contains information on the following topics:

- [“Protecting resources and data sets” on page 535](#)
- [“Protecting the usage of DFSMSdss” on page 535](#)
- [“Password protection” on page 536](#)
- [“Protected user and group data sets” on page 537](#)
- [“Generic and discrete profile considerations” on page 537](#)
- [“Security-level, category, and label checking” on page 539](#)
- [“Protect-all and always-call” on page 539](#)
- [“Standard naming conventions” on page 539](#)
- [“DFSMSdss temporary data set names” on page 539](#)
- [“Discretely protected multivolume data set” on page 541](#)
- [“Erase-on-scratch” on page 541](#)
- [“SMS-managed data set protection” on page 541](#)
- [“Logging” on page 542](#)

Protecting resources and data sets

A security administrator or resource owner can control access to resources by creating a discrete profile, which protects one resource, or a generic profile, which protects one or more resources. Each profile has a defined, universal access level and an access list that permit user and group IDs specific levels of access authority. When profiles control resources, DFSMSdss usage can be restricted.

Protecting the usage of DFSMSdss

Your installation can put DFSMSdss in a protected library to restrict its use. Your installation can also limit the use of certain DFSMSdss commands, functions, and keywords by defining resource profiles in the FACILITY class and restricting access to those profiles. To use a protected command, function, or keyword, you need READ access authority to the applicable profile. The FACILITY class profiles do not allow users to process data sets they do not have authority to. See [“Protected user and group data sets” on page 537](#) for requirements for access to data sets and volumes.

Table 28 on page 535 lists those keywords and functions, and their associated RACF FACILITY class profiles. For information about the ADMINISTRATOR keyword, see [“FACILITY class profiles for the ADMINISTRATOR keyword” on page 543](#).

<i>Table 28. DFSMSdss FACILITY Class Profiles.</i> This table maps the DFSMSdss keywords and functions to their associated FACILITY class profiles.	
Keyword or Function	Profile Name
BYPASSACS with COPY	STGADMIN.ADR.COPY.BYPASSACS
BYPASSACS with RESTORE	STGADMIN.ADR.RESTORE.BYPASSACS
CGCREATED	STGADMIN.ADR.CGCREATE
CLOUD with DUMP	STGADMIN.ADR.DUMP.CLOUD
CLOUD with RESTORE	STGADMIN.ADR.RESTORE.CLOUD
CONCURRENT with COPY	STGADMIN.ADR.COPY.CNCURRNT
CONCURRENT with DUMP	STGADMIN.ADR.DUMP.CNCURRNT
CONSOLIDATE	STGADMIN.ADR.CONOLID

Table 28. DFSMSdss FACILITY Class Profiles. This table maps the DFSMSdss keywords and functions to their associated FACILITY class profiles. (continued)

Keyword or Function	Profile Name
CONVERTV	STGADMIN.ADR.CONVERTV
DEFRAG	STGADMIN.ADR.DEFRAG
DELETECATALOGENTRY with RESTORE	STGADMIN.ADR.RESTORE.DELCATE
FCCGFREEZE with COPY	STGADMIN.ADR.COPY.FCFREEZE
FCFASTREVERSERESTORE	STGADMIN.ADR.COPY.FCFR
FCSETGTOK with COPY	STGADMIN.ADR.COPY.FCSETGT
FCTOPPRCPPRIMARY with COPY	STGADMIN.ADR.COPY.FCTOPPRCP
FCTOPPRCPPRIMARY with DEFRAG	STGADMIN.ADR.DEFRAG.FCTOPPRCP
FlashCopy with COPY	STGADMIN.ADR.COPY.FLASHCPY
FlashCopy with CONSOLIDATE	STGADMIN.ADR.CONOLID.FLASHCPY
FlashCopy with DEFRAG	STGADMIN.ADR.DEFRAG.FLASHCPY
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
RESET with DUMP	STGADMIN.ADR.DUMP.RESET
RESET with RESTORE	STGADMIN.ADR.RESTORE.RESET.YES
SET PATCH	STGADMIN.ADR.PATCH
SPACEREL	STGADMIN.ADR.SPACEREL
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
ZCOMPRESS with DUMP	STGADMIN.ADR.DUMP.ZCOMPRESS

Related reading: For more information about FACILITY class profiles, see [z/OS Security Server RACF Security Administrator's Guide](#).

Password protection

DFSMSdss supports password checking at the data set level, but not the catalog level. If a data set is password-protected and RACF-protected, access to the data set is determined only through RACF-authorization checking (password checking is bypassed).

Data set passwords in the DFSMSdss input command do not appear in the SYSPRINT data set listing. However, if your job abnormally ends, those passwords may appear in any resultant dump. To prevent this, you can put the passwords in a data set that is referred to with a DD statement.

Protected user and group data sets

DFSMSdss does authorization checks to ensure that you have the authority to access data sets. Your ability to access and protect a data set is affected by whether the data set is a user data set or a group data set.

User data set

The high-level qualifier of the name of a user data set is a RACF-defined user ID. As a RACF-defined user, you can protect your own data sets. However, when you use DFSMSdss to discretely protect a data set for another user, you need the system-OPERATIONS attribute.

Usually, you can use DFSMSdss to create new user data sets if you own them or have ALTER access to them through a data set profile or an entry in the global access checking table. You can also create a new user data set in any of the following situations:

- You are acting as a DFSMSdss-authorized storage administrator through the ADMINISTRATOR keyword.
- You have DASDVOL-access authority to the non-SMS-managed volume that the data set is being created on.
- You have the system-OPERATIONS attribute, and you have not been explicitly denied access to the data set.
- The system has *always-call*, the data set name is protected by a generic profile, and you do not have Automatic Data Set Protection (ADSP).
- The data set is not protected by a generic profile, and you do not have ADSP.

Related reading: For more information about *always-call*, refer to [“Protect-all and always-call” on page 539](#). For more information about ADSP, refer to [“Automatic data set protection \(ADSP\) attribute” on page 538](#).

Group data set

The high-level qualifier of the name of a group data set is a RACF-defined group ID. As a RACF-defined user, you can RACF-protect a group data set under any of these conditions:

- You have JOIN, CONNECT or CREATE authority in the group.
- You have the group-SPECIAL attribute in the group that owns the user profile.
- You have the system-OPERATIONS attribute, and you are not connected to the group.

You can create new group data sets with the DFSMSdss COPY or RESTORE command in the following situations:

- You are acting as a DFSMSdss-authorized storage administrator through the ADMINISTRATOR keyword.
- You have DASDVOL-access authority to the non-SMS-managed volume that the data set is being created on.
- You have the OPERATIONS attribute, and you have not been explicitly denied access to the data set.
- The system has *always-call*, the data set name is protected by a generic profile, you have ALTER-access authority to the data set profile, and you do not have ADSP. Instead of ALTER access, you can have CREATE authority in the group and UPDATE access to the data set.
- The data set is not protected by a generic profile, and you do not have ADSP.

Generic and discrete profile considerations

A generic or a discrete data set profile can protect a data set. The type of profile affects DFSMSdss functions, especially copy, dump, and restore.

Generic profiles

Generic profiles can be used to protect existing data sets without turning on the RACF-indicator flag in the data set VTOC entry for a non-VSAM data set or in the catalog entry for a VSAM data set. Generic profiles can be used to permit access to data sets, deny access to data sets, and control who can create data sets. A small number of generic profiles can be used to protect many data sets.

Generic profile checking for the DATASET class must be activated by the RACF SETROPTS GENERIC(DATASET) command. An entry in the global access checking table can let a user access a data set that the generic data set profile does not. For more information, see [“Global access checking table” on page 538](#).

Discrete profiles

A discrete profile should be used to protect a data set only if the data set has a unique security requirement from any other resource. You can protect data sets with discrete profiles when you use the DFSMSdss COPY or RESTORE command if you have the authority to define a discrete profile to protect that data set.

When a data set is protected with a discrete profile, an indicator is set in the VTOC entry for a non-VSAM data set, in the catalog entry for a VSAM data set, or in the tape volume profile for the tape volume that contains a tape data set. This condition is called RACF-indicated.

The following are special considerations that apply to discrete profiles:

- If the data set is scratched, RACF deletes the discrete profile.
- If you rename a data set to your high-level qualifier, the data set and the discrete profile are renamed, and the owner information of the profile is changed to your user ID.
- If you rename a data set to a high-level qualifier other than your own, the owner of the high-level qualifier becomes the owner of the data set.

Discrete and generic profile checking

For RACF-indicated data sets, RACF searches first for a discrete profile, and then, if one is not found, for a generic profile. If neither is found, access is denied. If a data set is not RACF-indicated, the data set is RACF-protected only if there is a covering generic profile. The search is done in the following order:

1. Discrete profile if the data set is RACF-indicated.
2. Fully-qualified generic profile.
3. Other generic profiles from the most specific to the least specific profile name.

Global access checking table

Your installation can use entries in the global access checking table to permit, but not deny, access to data sets. Only data set profiles can be used to deny access to data sets. Each entry in the table should have a corresponding generic profile to ensure consistent processing results.

Automatic data set protection (ADSP) attribute

If you have the ADSP attribute, RACF automatically defines a discrete profile whenever you create a permanent DASD or tape data set that is not already protected. For a tape data set, TAPEDSN and TAPEVOL must be active. If you have the ADSP attribute, you can create and protect data sets:

- Whose names begin with your user ID.
- Whose high-level qualifier belongs to a RACF group in which you have CREATE or higher authority. Besides CREATE in the group, you also need UPDATE access to the data set when you use DFSMSdss to copy or restore a data set.

Security-level, category, and label checking

Data set and volume

DFSMSdss does not perform any explicit security-level, security-category, or security-label (SECLABEL) checking. For example, SECLABELs are not dumped or restored unless they are embedded within the data itself such as zFS data sets.

z/OS UNIX files

SECLABELs are processed during dump and restore if they are present in the file attributes.

Protect-all and always-call

If protect-all has been activated, you can access a data set only through a data set profile or through an entry in the global access checking table. When always-call is in effect, RACF is called whenever a data set is accessed or DASD space is allocated. When RACF is called because of always-call (and not because of RACF-indication), it only checks generic profiles and the global access checking table. If protect-all is not in effect and RACF cannot find an appropriate data set profile or entry in the global access checking table, RACF accepts the request by default. Be aware of the following conditions:

- Always-call is in effect if the data sets are cataloged.
- Data sets that are not RACF-indicated, but which are protected by generic profiles and always-call, are not protected if they are transferred to another system that does not have RACF, always-call, and appropriate generic profiles.
- VSAM data sets are protected only by the RACF profile for the cluster name. Profiles for the index- and data-component names are ignored, as are the profiles associated with any PATHs or with the cluster's alternate indexes (AIXs).

Standard naming conventions

By default, RACF expects a data set name to consist of at least two qualifiers, with the high-level qualifier either a RACF-defined user or group ID. Single-qualifier data set names, especially for DASD data sets, affect your ability to manage or protect your data sets. For example:

- Data set name filtering is less usable for selecting data sets to be processed.
- DFSMSdss definitions of discrete data set profiles fail due to lack of correct prefix information.
- Your installation has trouble protecting the temporary data set names that DFSMSdss uses.

DFSMSdss temporary data set names

DFSMSdss must allocate 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. DFSMSdss prints 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. ....volser.DUMMY
```

where 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 or CONSOLIDATE operation ends successfully.

If the operation is interrupted (for example, if DFSMSdss is canceled), the data set is left on the volume. To delete the data set, repeat the DEFRAG or CONSOLIDATE operation. Before doing so, observe the following considerations:

- You might need to convert an index VTOC (IXFORMAT) volume to non-indexed VTOC (OSFORMAT) before rerunning the DEFRAG or CONSOLIDATE operation. Otherwise, the volume free space values might be incorrect.
- 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 that is 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, 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 *hmmssstt* 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 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 with the following name format:

```
CATHLQ.EXPORT.Thmmsstt
```


- where,

CATHLQ

First three high-level qualifiers of the catalog being copied

hmmsstt

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 with the following name format:

```
CATHLQ.Thmmsstt
```

- where,

CATHLQ

First four high-level qualifiers of the catalog that is being copied

hmmsstt

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 to be rebuilt or the volume free-space values need to be recalculated. The data set name is in the following format:

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

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

Discretely protected multivolume data set

To create a discrete profile for a multivolume, non-VSAM, DASD data set, you must define each volume of the data set to RACF. When the data set is extended to another volume or deleted from a volume, that volume's serial number is automatically added to or deleted from the data set profile.

Erase-on-scratch

When the erase indicator is set in a DASD data set profile, the tracks of any scratched or released data set extents that are part of the protected DASD data set are erased. Erase-on-scratch is supported for the following DFSMSDss commands:

- DUMP with DELETE
- COPY with DELETE
- DEFRAg
- RELEASE

When DFSMSDss deletes a data set or moves data extents, the original tracks are erased if the erase indicator is set. This also applies during a copy or restore when preallocated data sets are deleted and reallocated.

SMS-managed data set protection

A data set profile may contain a DFP segment. The DFP segment contains a RESOWNER field, which may be used to specify the owner of an SMS-managed data set that is protected by the data set profile. If a new SMS-managed data set is allocated, the user or group ID in the RESOWNER field must have at least READ access to any MGMTCLAS and STORCLAS profile used in the allocation when:

- Your installation has activated the RACF general resource classes MGMTCLAS and STORCLAS.
- Profiles have been defined in those classes to protect against unauthorized usage of an SMS management class name or a storage class name.

If RESOWNER is not specified when the data set is allocated, the user or group ID that matches the high-level qualifier is used as the data set owner. Regardless of who owns an SMS-managed data set, you can select different default values for MGMTCLAS and STORCLAS by using those parameters with the COPY or RESTORE command. When you do, RACF checks to ensure that the data set owner, rather than you, is authorized to use the specified MGMTCLAS and STORCLAS. The user or group ID must not be revoked.

Storage class and Management class authorization checking can be bypassed for logical and physical data set restore and physical data set copy processing by using the ADRPATCH Serviceability Aid with the ADMINISTRATOR keyword.

Related reading: For more information about using the ADRPATCH Serviceability Aid, see [Chapter 14, “DFSMSdss patch area,”](#) on page 213.

Logging

DFSMSdss automatically supports RACF logging as follows:

- Logging of DASDVOL-access checks is allowed for successful authorizations only. Unsuccessful attempts are not logged because the user can still gain access at the data set catalog level.
- Logging of access checking for data sets occurs as specified in the applicable data set profile. However, because catalog management services are used, DFSMSdss does not fully control logging for VSAM data sets.
- Logging of access to the FACILITY class profiles for DFSMSdss is allowed as defined for those profiles.
- DFSMSdss does not control logging of RACF DEFINE requests. DEFINE requests are made to define discrete profiles and to determine if a user has CREATE authority in a group.
- Logging of access-level checks for a catalog occurs only once.
- Logging cannot be disabled by specifying RACFLOG=NO in the PARM statement of a DFSMSdss job. The only way to turn off logging is through the DFSMSdss installation options exit.

DFSMSdss storage administrator

Your installation can use DASDVOL access authority to designate users who can act as DFSMSdss storage administrators. DASDVOL access authority lets you perform any DFSMSdss function against the data sets on that volume. However, DASDVOL access is not supported for logical operations against SMS-managed data sets. Further, DASDVOL access authority must be granted in every volume for all the data sets for which you are the storage administrator.

DFSMSdss provides an alternative to DASDVOL access authority. Your installation can define special FACILITY class profiles to let you act as a DFSMSdss-authorized storage administrator.

ADMINISTRATOR keyword

To act as a DFSMSdss-authorized storage administrator, specify the ADMINISTRATOR keyword on the appropriate DFSMSdss command. DFSMSdss-initiated access checking to data sets, z/OS UNIX files, and catalogs is bypassed. If you are not authorized to use the ADMINISTRATOR keyword, the command ends with an error message.

Note that certain authorization checking to RACF for data sets is unavoidable, regardless of whether the ADMINISTRATOR keyword is specified. For example, checks that are necessary to determine whether a data set has a defined RACF profile, or authorization checks that are initiated by callable services or utilities that DFSMSdss invokes such as Catalog and IEBCOPY, cannot be bypassed.

To use the ADMINISTRATOR keyword, all of the following must be true:

- FACILITY class is active.
- Applicable FACILITY class profile is defined.
- You have READ access to that profile.

As a DFSMSdss-authorized storage administrator, your authority is for physical, logical, and path operations. For example, a DFSMSdss-authorized storage administrator for the COPY command can copy tracks, a full volume, or data set without having access to the individual data sets or their catalogs.

DFSMSdss tries to define a discrete profile when you copy a discretely-protected data set or when you use the MENTITY keyword. The authority to do so is not given to you by the DFSMSdss ADMINISTRATOR support. For example, to define or rename a discrete profile for a user data set that you do not own, you also need the system-OPERATIONS attribute. The ADMINISTRATOR support also does not give you authority to bypass checking for MGMTCLAS and STORCLAS authority.

Storage class and Management class authorization checking can be bypassed for logical and physical data set restore and physical data set copy processing by using the ADRPATCH Serviceability Aid in conjunction with the ADMINISTRATOR keyword.

Related reading: For more information about the ADRPATCH Serviceability Aid, see [Chapter 14](#), “DFSMSdss patch area,” on page 213.

FACILITY class profiles for the ADMINISTRATOR keyword

The following are the names and descriptions of the FACILITY class profiles for the ADMINISTRATOR keyword.

STGADMIN.ADR.STGADMIN.COMPRESS

Lets you compress data sets without having UPDATE access authority to those data sets.

STGADMIN.ADR.STGADMIN.CONOLID

Lets you perform a CONSOLIDATE operation without having READ access to the data sets that are moved. You can call RACF to determine if erase-on-scratch processing for a given data set must be done.

STGADMIN.ADR.STGADMIN.COPY

Lets you copy data sets without having access authority to the source (READ) or target (UPDATE or ALTER) data sets and their catalogs. This profile does not give you the authority to rename a data set.

STGADMIN.ADR.STGADMIN.COPY.DELETE

Lets you copy or copy and delete data sets without having access authority to the source (ALTER) or target (UPDATE or ALTER) data sets and their catalogs. This profile does not give you the authority to rename a data set.

STGADMIN.ADR.STGADMIN.COPY.RENAME

Lets you copy or copy and rename data sets, through the RENAMEUNCONDITIONAL keyword, without having access authority to the source (READ) or the target (ALTER) data sets and their catalogs. This profile does not give you the authority to delete a data set.

STGADMIN.ADR.STGADMIN.DEFRAG

Lets you perform a DEFRAG operation without having READ access to the data sets that are moved. RACF can still be called for DFSMSdss to determine if erase-on-scratch processing for a given data set must be done.

STGADMIN.ADR.STGADMIN.DUMP

Lets you dump data sets without having READ access to the data sets. This profile does not give you the authority to delete a data set.

STGADMIN.ADR.STGADMIN.DUMP.DELETE

Lets you dump or dump and delete data sets without having ALTER access to the data sets and their catalogs.

STGADMIN.ADR.STGADMIN.DUMP.NEWNAME

Lets you rename data sets during dump processing. This profile does not give you the authority to delete a data set.

STGADMIN.ADR.STGADMIN.DUMP.PATH

Lets you dump z/OS UNIX files without having the applicable access authority as imposed by z/OS UNIX System Services.

STGADMIN.ADR.STGADMIN.PRINT

Lets you print data without having READ access to the data sets containing the data to be printed.

STGADMIN.ADR.STGADMIN.RELEASE

Lets you release unused space without having UPDATE access to the data sets.

STGADMIN.ADR.STGADMIN.RESTORE

Lets you restore data without having READ, UPDATE, or ALTER access to source and target data sets and their catalogs. This profile does not give you the authority to rename a data set.

STGADMIN.ADR.STGADMIN.RESTORE.PATH

Lets you restore z/OS UNIX files without having the applicable access authority as imposed by z/OS UNIX System Services.

STGADMIN.ADR.STGADMIN.RESTORE.RENAME

Lets you restore or restore and rename data sets, through the RENAME or RENAMEUNCONDITIONAL keyword, without having READ, UPDATE, OR ALTER access to source and target data sets and their catalogs.

DFSMSdss access authority

DFSMSdss checks your access authority to the volume before it performs any data set access-authorization checking. You have the required volume-level authority whenever any of the following is true:

- Bypass authorization checking is requested by the DFSMSdss installation authorization exit. This exit cannot be used to bypass authorization checking initiated by utilities such as IEBCOPY that are invoked by DFSMSdss.
- You have the required DASDVOL access authority for the function you are trying to perform.
- You are acting as a DFSMSdss-authorized storage administrator.

This is one of the recommended ways to allow limited authorize volume level access. See the preceding information in [“ADMINISTRATOR keyword” on page 542](#) and [“FACILITY class profiles for the ADMINISTRATOR keyword” on page 543](#).

z/OS UNIX files

DFSMSdss does not directly invoke SAF/RACF during z/OS UNIX files access and authorization checking. Instead, DFSMSdss depends on z/OS UNIX System Services to perform file-access checking on our behalf. File access (including privileged user) checking is performed using the identity of the invoker as defined by RACF. Refer to the z/OS UNIX System Services User's Guide under the topic 'Handling security for your files' for more information on this topic.

For SMS-managed data sets, access authorization to create, update, or delete data sets also authorizes you to create, update, or delete entries in user catalogs. However, to add or delete any entry for an SMS-managed data set in a protected master catalog, you also need UPDATE access to the master catalog. To add or delete any non-SMS entry in any protected catalog, you need UPDATE access to the catalog.

Figure 25 on page 545 shows the major decisions that are made to determine if you are authorized to perform a function against a data resource. The figure does not represent the actual program flow in DFSMSdss.

Related reading: For more information about DFSMSdss use of utilities, see [“Moving data sets with utilities” on page 104](#).

Volume access and DASDVOL

You can use DFSMSdss to perform volume-level operations such as a full-volume dump. Instead of requiring you to have sufficient access authority to each data set on the volume, DFSMSdss supports DASDVOL access authority.

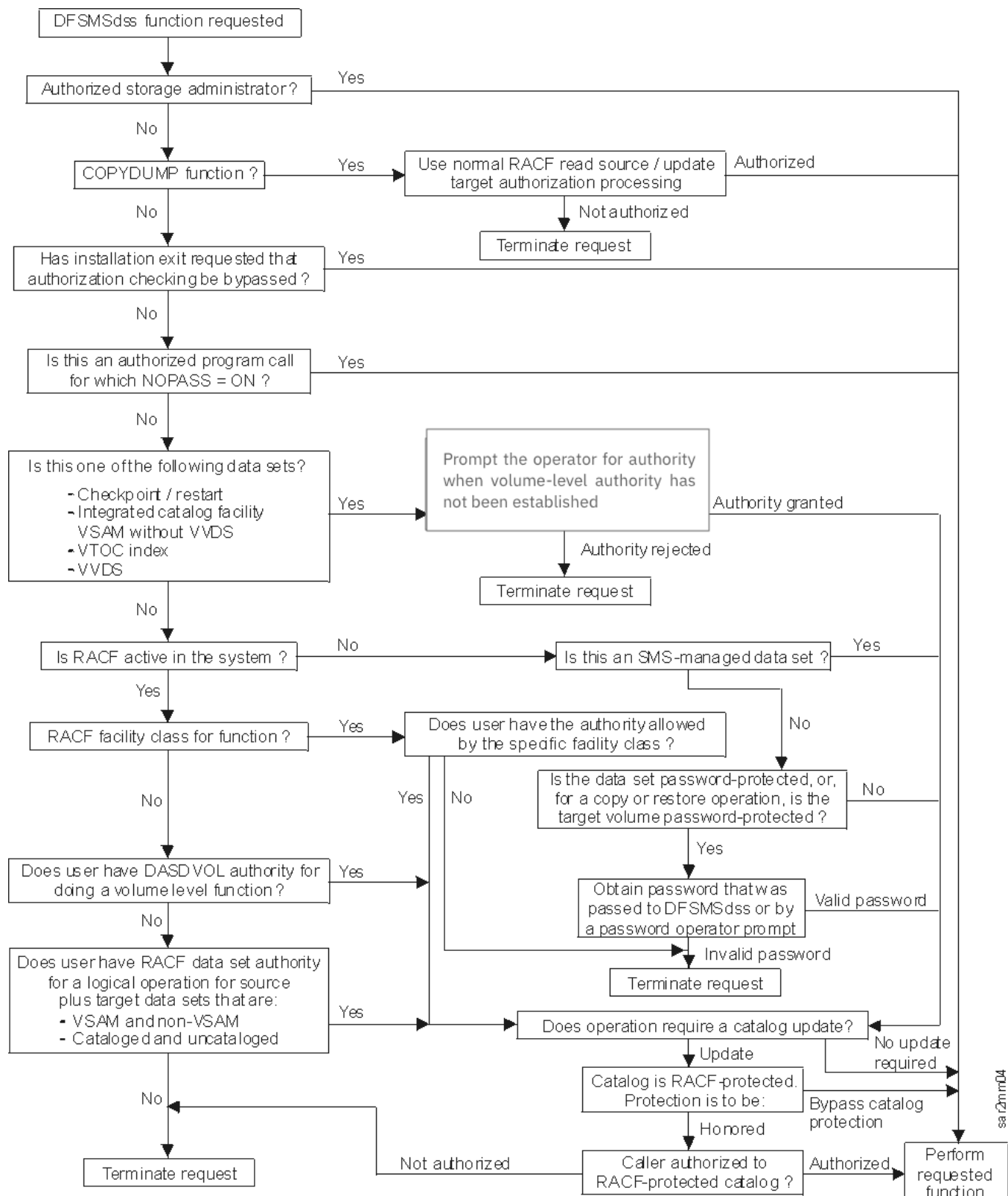


Figure 25. DFSMSdss Data Security Decisions

DASDVOL access authority

When you have the required level of DASDVOL authority to a volume, as shown in Table 29 on page 546, you can perform maintenance operations such as dump, restore, scratch and rename of the data sets on that volume regardless of your access authority to those data sets. DFSMSdss lets you catalog, re-catalog,

and uncatalog the data sets on the volumes to which you have DASDVOL authority, even if you do not have RACF authority to the specific catalog. You also have access to the catalogs on the volumes to which you have DASDVOL access authority, regardless of the particular RACF definition for that catalog.

When you restore a data set, DASDVOL access checking is determined as follows:

- The volume serial number of the source volume from which the data set was originally dumped, regardless of the current status of that data set or volume.
- The volume serial number of the volume on which the data set is being restored. This volume (or volumes) may not be the same as the source volume.

Table 29. DASDVOL Access Authority. This table shows the DASDVOL-access authorities needed to perform volume and data-set-level functions.

Function		DASDVOL Access Needed		
Command	Volume	FULL	TRACKS	Data Set
COMPRESS	Source	N/A	N/A	UPDATE
CONSOLIDATE	Source	N/A	N/A	UPDATE
CONVERT	N/A	N/A	N/A	N/A
COPY	Source	READ	READ	READ
	Target	ALTER	ALTER	UPDATE
COPY with DELETE	Source	N/A	N/A	ALTER
	Target	N/A	N/A	UPDATE
COPYDUMP	N/A	N/A	N/A	N/A
DEFRAG	Source	N/A	N/A	UPDATE
DUMP	Source	READ	READ	READ
DUMP with DELETE	Source	N/A	N/A	ALTER
PRINT	Source	N/A	READ	READ
RELEASE	Source	N/A	N/A	UPDATE
RESTORE	Source	N/A	N/A	READ
	Target	ALTER	ALTER	UPDATE
SPACEREL	Source	ALTER	N/A	N/A

DASDVOL limitations

DASDVOL access authority is not supported for logical operations on SMS-managed data sets, nor does it let you define discrete profiles. For DFSMSdss to define or rename discrete profiles, you need the authority to do so.

While DASDVOL access authority to a non-SMS-managed volume lets you move, copy, dump, or delete a catalog, you do not automatically have access to the data sets that are cataloged in it. Also, for certain operations (such as CONVERTV on an SMS-managed data set), an authorization check is performed to see if the data set owner is authorized to use the applicable MGMTCLAS and STORCLAS routines. This is true even if you have DASDVOL access authorization.

For more information about discrete profiles, see [“Copy and data set profile considerations” on page 551](#) and [“Restore and data set profile considerations” on page 557](#).

Data set access authorization levels

Access to a data set is controlled by a data set profile which permits or denies access. Access-authorization checking to data sets is performed if you do not have the required volume-level authority, the volume is unprotected, or DFSMSdss is unable to determine the protection status of the volume.

If the data set and its catalog are unprotected, no authorization is required to read, update, delete, rename, move, or scratch the data set. For protected data sets, the following access authorization may be required to perform DFSMSdss functions:

- Password specification is required. If the data set is password-protected and not RACF-protected, you must specify the appropriate password to read, update, delete, rename, move, or scratch the data set.
- The data set is RACF-protected. Your ability to perform operations is determined by your access-authorization level:

NONE

denies you access to the data set.

READ

lets you read the data set, and even make a copy of it, provided you have authority to create or replace the target data set.

UPDATE

lets you modify (update) an existing RACF-protected data set. UPDATE access does not let you delete, rename, move, or scratch a data set.

ALTER

lets you read, update, delete, rename, move, or scratch a RACF-protected data set.

ALTER access lets you create a user or group data set. CREATE authority in the group and UPDATE access also lets you create a new group data set.

The level of access authority you need to access data sets is also dependent on the following:

- Whether the data set is SMS-managed or not.
- Whether a catalog entry for the data set is added (cataloged) or deleted (uncataloged).
- Whether the catalog is an integrated catalog facility catalog.
- Whether the catalog is RACF-protected.

DFSMSdss supports VSAM data sets only if they are cataloged in an integrated catalog facility catalog. For VSAM data sets, access authorization is determined solely by your authority to access the base-cluster name: Data set profiles are ignored for any data component, index component, alternate index (processed as part of a sphere) or PATH.

Protected catalogs

DFSMSdss only supports RACF-protection of catalogs; catalog passwords are never checked.

Catalog access authority

DFSMSdss supports cataloged non-VSAM and VSAM data sets in an integrated catalog facility catalog.

For more information about the access authority you need, see [“Access authorization for DFSMSdss commands”](#) on page 548.

Non-SMS versus SMS authorization

Access authorization requirements differ for SMS-managed data sets and non-SMS-managed data sets.

SMS-managed data sets:

Access to SMS-managed data sets gives you access to the user catalog for the data sets. However, for a RACF-protected master catalog, you also need UPDATE-access authority to add or delete an SMS-managed data set entry. DASDVOL-access authority is not supported for logical operations.

Non-SMS-managed data sets:

Besides access authority for data sets, you need UPDATE access to RACF-protected user or master catalogs to add or delete an entry in the catalog.

System operator authorization, special data set types

Some system and VSAM data sets accept authorization from the system operator when RACF or password checking cannot be performed. Unless you are using DFSMSdss with special authorization, such as DASDVOL or the ADMINISTRATOR keyword, system-operator authorization is required in order for you to update the following:

- Volume table of contents (VTOC)
- VTOC index data set
- VSAM volume data set (VVDS)
- Checkpoint/restart data set.

The system operator is only prompted for the first data set encountered in one of the above classes for the DFSMSdss command that is being processed. The reply given is used for that command invocation for all other data sets of that type on the volume.

Access authorization for DFSMSdss commands

The tables in this section can help you determine if you have sufficient access authority to use the DFSMSdss commands. The access authority you need depends on what you are trying to do. For example:

- If the data set is SMS-managed and you have access to the data set, you have access to its user catalog.
- If you have ALTER access to the data set, you can copy and delete (or dump and delete) the data set.
- If the source data set is cataloged, you may need UPDATE or ALTER access to its catalog if the data set is going to be deleted, uncataloged, or cataloged.
- To replace an existing data set using the COPY or RESTORE command, you need UPDATE access to the data set.
- If you are doing a restore with rename, ALTER-access authority lets you create a new user or group data set. If the data set belongs to a group, CREATE authority in the group and UPDATE access to that group data set name also lets you create that group data set.
- To restore a data set, you need READ access to a data set with the same name as the one that was dumped. You also need either UPDATE or ALTER access to the target data set.
- If the target data set is going to be discretely protected, and you do not own the data set, you need additional authority to define a discrete profile. This is true even if you are acting as a DFSMSdss-authorized storage administrator for the COPY or RESTORE command.
- To copy or dump a master catalog, you need ALTER access if you are not acting as a DFSMSdss storage administrator. This level of access is needed because the master catalog may contain passwords.
- If you are acting as a DFSMSdss storage administrator, you have access to the applicable data sets and catalogs.

CGCREATED

Anyone can use the CGCREATED command. However, your installation can limit usage of the CGCREATED command by use of a RACF FACILITY class profile if:

- RACF FACILITY class is active.
- The FACILITY class profile STGADMIN.ADR.CGCREATE has been defined. Then need READ access authorization to that profile to use the CGCREATED command.

CLOUDUTILS

Anyone can use the CLOUDUTILS command. However, you can limit usage of the CLOUDUTILS command by using a RACF® FACILITY class profile. To use the profile, the following conditions must be met:

- The RACF FACILITY class is active.
- The FACILITY class profile STGADMIN.ADR.CLOUDUITLS has been defined, and the user has READ access authorization to that profile.

COMPRESS

To compress partitioned data sets on a specified volume, either you need DASDVOL-access authority to the volume or UPDATE access authority to the data sets, or you must be acting as a DFSMSdss-authorized storage administrator.

CONSOLIDATE

To use the CONSOLIDATE command to relocate data extents on a DASD volume, you need DASDVOL-access authority to that volume or READ-access to the data sets that are relocated. Otherwise, only unprotected data sets are relocated.

As part of the CONSOLIDATE operation, DFSMSdss creates a work data set (SYS1.DFDSS.DEFRAG.xxxxxxxx.volser.DUMMY), which is deleted after successful completion of the CONSOLIDATE function. If the CONSOLIDATE operation is prematurely ended, rerun the CONSOLIDATE function to clean up this data set.

CONVERTV

Anyone can use the CONVERTV command. However, your installation can limit usage of the CONVERTV command by use of a RACF FACILITY class profile if:

- RACF FACILITY class is active.
- The FACILITY class profile STGADMIN.ADR.CONVERTV has been defined. Then need READ access authorization to that profile to use the CONVERTV command.

To convert a volume to SMS-managed, the user and group IDs of the owners of all the data sets on that volume must be RACF-defined, and they must not be revoked.

COPY

Table 30 on page 550, which shows the access authority needed to perform the copy function, is based on the following assumptions:

- The data set is a user data set.
- Both the data set and the catalog are RACF-protected.
- If the data set is going to be cataloged in the master catalog, then you need UPDATE access to that catalog.

Table 30 on page 550 also applies to group data sets. To create a new group data set, you need one of the following:

- UPDATE access to the catalog when a non-SMS-managed group data set is unprotected, but the catalog is RACF-protected.
- CREATE in the group and UPDATE access to the data set.
- ALTER access to the data set and be a member of the group that owns the data set. To discretely protect a group data set, you need additional authority such as CREATE in the group.

Table 30. Access Authority for the DFSMSdss COPY Command. This table shows the minimum access levels required to perform a COPY command.

SOURCE DATA SET			TARGET DATA SET		
Non-SMS			Non-SMS		
With DELETE	Access Level		Preallocated Target	Access Level	
	Data Set	Catalog		Data Set	Catalog
NO	READ	Automatic	NO	ALTER	UPDATE
NO	READ	Automatic	YES	UPDATE	Automatic
YES	ALTER	Automatic	NO	ALTER	UPDATE
YES	ALTER	Automatic	YES	UPDATE	Automatic
Non-SMS			SMS		
With DELETE	Access Level		Preallocated Target	Access Level	
	Data Set	Catalog		Data Set	Catalog
NO	READ	Automatic	NO	ALTER	Automatic
NO	READ	Automatic	YES	UPDATE	Automatic
YES	ALTER	Automatic	NO	ALTER	Automatic
YES	ALTER	Automatic	YES	UPDATE	Automatic
SMS			Non-SMS		
With DELETE	Access Level		Preallocated Target	Access Level	
	Data Set	Catalog		Data Set	Catalog
NO	READ	Automatic	NO	ALTER	UPDATE
NO	READ	Automatic	YES	UPDATE	Automatic
YES	ALTER	Automatic	NO	ALTER	UPDATE
YES	ALTER	Automatic	YES	UPDATE	Automatic
SMS			SMS		
With DELETE	Access Level		Preallocated Target	Access Level	
	Data Set	Catalog		Data Set	Catalog
NO	READ	Automatic	NO	ALTER	Automatic
NO	READ	Automatic	YES	UPDATE	Automatic
YES	ALTER	Automatic	NO	ALTER	UPDATE
YES	ALTER	Automatic	YES	UPDATE	Automatic

Note:

1. "Automatic" means that you have automatic access to the user catalog if you can access the data sets.
2. In the course of copying data sets, DFSMSdss will rename the source data set using the conventions described under DFSMSdss Temporary Data Set Names. (See [“Temporary copied data sets”](#) on page 540 under Temporary Copied Data Sets.) Whenever DFSMSdss renames a RACF protected data set to a temporary name, a RACF profile must exist for the temporary data set name.

COPY and deleting data sets

For SMS-managed data sets, ALTER access lets you delete the data set. For non-SMS-managed data sets, either ALTER access to the data set or ALTER access to the catalog and READ access to the data set lets you copy and delete the data set. The ability to delete and uncatalog a data set with READ access to the data set and ALTER access to the catalog lets you move a data set without having ALTER access to the data set. You only need UPDATE access to the catalog to delete an unprotected, non-SMS-managed data set.

Copy and data set profile considerations

When you copy a RACF-indicated data set, DFSMSdss performs special target data set processing. A predefined, discrete profile is not used to check your access authority unless the data set is already RACF-indicated.

Copy and data set profiles

If a data set is protected by a generic or discrete profile when it is copied, DFSMSdss tries to ensure that the target data is also protected (see Table 31 on page 551). To do so, DFSMSdss uses the RACF DEFINE function or the catalog ALTER function. You need authority to define or rename discrete profiles, even if you are acting as a DFSMSdss-authorized storage administrator.

Table 31. DFSMSdss COPY Command and RACF Profiles. This table summarizes how DFSMSdss defines discrete profiles to protect the target set.

COPY with:		Source Data Set Protected?	RACF Profile:	
RENAME	DELETE		Source	Target
NO	NO	NO	None	As predefined
NO	NO	GENERIC	No change	As predefined
NO	NO	DISCRETE	No change	DEFINE
NO	YES	NO	None	As predefined
NO	YES	GENERIC	No change	As predefined
NO	YES	DISCRETE	DELETE	DEFINE
YES	NO	NO	None	As predefined
YES	NO	GENERIC	No change	MENTITY
YES	NO	DISCRETE	No change	DEFINE
YES	YES	NO	None	As predefined
YES	YES	GENERIC	No change	MENTITY
YES	YES	DISCRETE	DELETE	DEFINE

Table 31. DFSMSdss COPY Command and RACF Profiles. This table summarizes how DFSMSdss defines discrete profiles to protect the target set. (continued)

COPY with:		Source Data Set Protected?	RACF Profile:	
RENAME	DELETE		Source	Target
Terms defined:				
None The data set remains unprotected.				
As predefined DFSMSdss does not ensure that the target data set is RACF-protected. If a covering data set profile is not already predefined, the data set may be unprotected.				
No change The data set profile that is protecting the source data set remains unchanged.				
DEFINE A discrete profile is defined by DFSMSdss.				
DELETE The data set and the discrete profile are deleted.				
MENTITY If the target data set is not RACF-protected by a generic or discrete profile and MENTITY is specified, DFSMSdss defines a discrete profile. Otherwise, no profile is defined.				

COPY and the MENTITY keyword

By default, when DFSMSdss defines a discrete profile to protect a copied data set, it uses the discrete profile of the source data set as a model. The MENTITY keyword lets you specify a different data set profile as the model. Besides the model profile, you can also specify a volume serial (through MVOLSER). If you do so, specify the volume serial number of the volume containing the non-VSAM model entity or the volume containing the catalog in which the VSAM model entity is cataloged.

Use the MENTITY keyword of the COPY command when one of the following is true:

- The source data set is RACF-protected by a generic profile, but the target data set name is unprotected.
- The source data set is RACF-protected by a discrete profile, but you wish to use a different data set profile as the model for defining a new discrete profile.

COPY and define discrete profile summary

Table 32 on page 552 summarizes the circumstances during copy operation when a discrete profile is defined, when MENTITY is effective, when the target data set is RACF-indicated, and when related messages are issued. To define discrete profiles for data sets that you do not own, you need additional authorization such as the system-OPERATIONS attribute or CREATE in the group.

Table 32. DFSMSdss Copy and Define Discrete Profile Summary. This table summarizes what happens when DFSMSdss defines discrete profiles for a copy operation.

Protection		MENTITY Specified?	Define Profile?	RACF Indicated?	Warning Message?
Source	Target				
None	None	Yes	No	No	No
		No	No	No	No
None	Generic	Yes	No	No	No
		No	No	No	No

Table 32. DFSMSdss Copy and Define Discrete Profile Summary. This table summarizes what happens when DFSMSdss defines discrete profiles for a copy operation. (continued)

Protection		MENTITY Specified?	Define Profile?	RACF Indicated?	Warning Message?
Source	Target				
None	RACF Indicated	Yes	No	Yes	No
		No	No	Yes	No
Generic	None	Yes	Yes	Yes	No (W)
		No	No	No	Yes
Generic	Generic	Yes	No	No	No
		No	No	No	No
Generic	RACF Indicated	Yes	No	Yes	No
		No	No	Yes	No
RACF Indicated	None	Yes	Yes	Yes	No (W)
		No	Yes (C)	Yes	No (E)
RACF Indicated	Generic	Yes	Yes	Yes	No (W)
		No	Yes (C)	Yes	No (E)
RACF Indicated	RACF Indicated	Yes	No	Yes	No
		No	No	Yes	No

Terms defined:

Yes (C)

If there is a discrete profile for the source data set, a discrete profile is defined for the target data set.

No (W)

If the define of the discrete profile fails and the copy is successful, then the target data set is left as RACF-indicated and a warning message is issued. This is true even if DELETE is specified.

No (E)

If the define of the discrete profile fails and DELETE is specified, then the copy is unsuccessful and an error message is issued.

If the define of the discrete profile fails and DELETE is not specified, then the target data set is left as RACF-indicated and a warning message is issued.

Protection states after a copy or move

When a data set is copied or moved, the protection state of the target depends on the initial protection states of the source and the target data sets, whether MENTITY was specified, and whether discrete profiles are predefined. The following conditions may exist:

The source data set is unprotected.

The target data set may or may not be protected after the copy. A discrete profile is not defined by DFSMSdss, even if you specify MENTITY.

The source data set is protected but not RACF-indicated.

The target data set may or may not be protected after the copy:

- If the target data set is not already protected and MENTITY is not specified, the data set remains unprotected. If you do specify MENTITY, DFSMSdss defines a discrete profile:

- If the define is successful, the target data set is RACF-indicated. It is accessible according to the access list for the model profile that was used.
- If the define is unsuccessful, the target data set remains RACF-indicated. It may be inaccessible until a data set profile is successfully defined.
- If the target data set is already protected because it is RACF-indicated or RACF-protected, DFSMSdss does not define a discrete profile (even if you specify MENTITY). The target data set is accessible according to the access list of a predefined, protecting, data set profile.

The source data set is RACF-indicated.

The target data set is RACF-indicated after the copy:

- If the target data set is not RACF-indicated and if either there is a discrete profile for the source or MENTITY is specified, DFSMSdss defines a discrete profile:
 - If the define is successful, the data set is RACF-indicated. It is accessible according to the access list for the model profile that was used.
 - If the define is unsuccessful and MENTITY was specified or DELETE was not specified, the data set is RACF-indicated. It may be inaccessible until a data set profile is successfully defined.
 - If the define is unsuccessful and DELETE was specified (without MENTITY), the copy or move is unsuccessful. The target data set is deleted, and the source data set is kept.
- If the target data set is already RACF-indicated, a discrete profile is not defined by DFSMSdss (even if you specify MENTITY or DELETE). The target data set remains RACF-indicated.

Other COPY command considerations

To copy or move a data set that is discretely protected, you need authority to define discrete profiles because the move operation temporarily renames the data set and the discrete profile.

COPYDUMP

The access authority you need to the input and output dump data sets depends on many factors not under DFSMSdss control because access-authorization checking is performed by Open/Close/End-of-Volume (O/C/EOV) processing. The primary considerations are as follows:

- You need READ access if the input dump data set is protected.
- You need either UPDATE or ALTER access if the output dump data set is protected.
- You may need UPDATE access to the catalog if the output data set is going to be cataloged.

DEFRAG

To use the DEFRAG command to relocate data extents on a DASD volume, you need DASDVOL-access authority to that volume or READ-access to the data sets that are relocated. Otherwise, only unprotected data sets are relocated.

As part of the DEFRAG operation, DFSMSdss creates a work data set (SYS1.DFDSS.DEFRAG.xxxxxxxx.volser.DUMMY), which is deleted after successful completion of the DEFRAG function. If the DEFRAG operation is prematurely ended, rerun the DEFRAG function to clean up this data set.

DUMP

The access authority you need to the output dump data set depends on many factors not controlled by DFSMSdss because access-authorization checking is done by O/C/EOV. The primary considerations are as follows:

- You need access to a protected dump data set with:
 - ALTER access to create it.
 - UPDATE access if it is already allocated.

- CREATE authority in the group and UPDATE access if it is a group data set.
- You may need UPDATE access to a catalog if the dump data set is going to be cataloged.
- You may have sufficient authority to the dump data set if you have the system-OPERATIONS attribute.
- You also need access authority to the data sets that are going to be dumped as shown in [Table 33 on page 555](#).

[Table 33 on page 555](#) is for the following conditions:

- Both the data set and the catalog are RACF-protected.
- You have the indicated level of access authority to the data set.
- The data set is cataloged in a user catalog.

<i>Table 33. Access Authority for the DFSMSdss DUMP Command.</i> This table shows the minimum access levels required to perform a DUMP command.			
SMS-Managed Data Set?	With DELETE?	Access Level:	
		Source Data Set	Catalog
NO	NO	READ	Automatic
NO	YES	ALTER	UPDATE
YES	NO	READ	Automatic
YES	YES	ALTER	Automatic
Note: "Automatic" means that you have automatic access to the catalog if you can access the data set.			

Dumping and deleting data sets

For SMS-managed data sets, ALTER access to the data set lets you delete a data set. If the data set is cataloged in the master catalog, you also need UPDATE access to the catalog.

For non-SMS-managed data sets, you need one of the following to dump and delete a data set:

- ALTER access to a protected data set.
- READ access to a protected data set and ALTER access to a protected catalog.
- UPDATE access to a protected catalog if the data set is unprotected.
- DASDVOL-access authority to the volume containing the data set.
- Authority to specify DELETE as a DFSMSdss-authorized storage administrator.

PRINT

To use the PRINT command, you need either DASDVOL-access authority at the READ level or READ access to the data sets.

RELEASE

To use the RELEASE command, you need either DASDVOL-access authority at the UPDATE level or UPDATE access to the data sets.

RESTORE

The access authority you need to restore a data set from an input dump data set depends on many factors not controlled by DFSMSdss because access-authorization checking to the input dump data set is performed by O/C/EOV processing. The primary considerations are as follows:

- You need at least READ access to the dump data set if the dump data set is protected.
- You may need READ access to the catalog if the dump data set is cataloged.

- You may have sufficient authority to the dump data set if you have the system-OPERATIONS attribute.
- You also need access authority to the source and target data sets and their catalogs as shown in [Table 34 on page 556](#).

Restore and access authorization

When a data set is either restored or both restored and renamed, DFSMSdss tries to verify that you have READ access to a DASD data set with the same name as the data set that was dumped. This is done to ensure that an unauthorized person is not allowed to restore and rename one of your data sets. For example, the restore described below would be unsuccessful:

- Data set USER1 . PERSONAL . PAYHIST is RACF-protected so that USER2 cannot read it.
- Data set USER1 . PERSONAL . PAYHIST is dumped.
- USER2 tries to restore USER1 . PERSONAL . PAYHIST as USER2 . TEST . DATA.

When you restore a data set, your authority to read that data set is checked using the current data set profiles rather than the data set profiles that were defined when the data set was dumped:

- If a discrete profile is predefined for the source data set name, it is used regardless of the current state of the data set.
- Otherwise, if a generic profile is defined, it is used.
- If neither a discrete nor a generic profile is defined, you are given access. However, for VSAM data sets that are already cataloged and RACF-indicated, access is denied in this situation.
- For VSAM data sets, if the catalog no longer exists or the data set is no longer cataloged, only generic-profile checking is performed.

Table 34. Access Authority for the DFSMSdss RESTORE Command. This table shows the minimum access-levels required to perform a RESTORE command.

SOURCE DATA SET			TARGET DATA SET		
Non-SMS			Non-SMS		
RENAME or RENAMEU	Access Level		Preallocated Target	Access Level	
	Data Set	Catalog		Data Set	Catalog
NO	No check	No check	NO	ALTER	UPDATE
NO	No check	No check	YES	UPDATE	Automatic
YES	READ	Automatic	NO	ALTER	UPDATE
YES	READ	Automatic	YES	UPDATE	Automatic
Non-SMS			SMS		
RENAME or RENAMEU	Access Level		Preallocated Target	Access Level	
	Data Set	Catalog		Data Set	Catalog
NO	No check	No check	NO	ALTER	Automatic
NO	No check	No check	YES	UPDATE	Automatic
YES	READ	Automatic	NO	ALTER	Automatic
YES	READ	Automatic	YES	UPDATE	Automatic
SMS			Non-SMS		
RENAME or RENAMEU	Access Level		Preallocated Target	Access Level	
	Data Set	Catalog		Data Set	Catalog

Table 34. Access Authority for the DFSMSdss RESTORE Command. This table shows the minimum access-levels required to perform a RESTORE command. (continued)

SOURCE DATA SET			TARGET DATA SET		
NO	No check	No check	NO	ALTER	UPDATE
NO	No check	No check	YES	UPDATE	Automatic
YES	READ	Automatic	NO	ALTER	UPDATE
YES	READ	Automatic	YES	UPDATE	Automatic
SMS			SMS		
RENAME or RENAMEU	Access Level		Preallocated Target	Access Level	
	Data Set	Catalog		Data Set	Catalog
NO	No Check	No check	NO	ALTER	Automatic
NO	No check	No check	YES	UPDATE	Automatic
YES	READ	Automatic	NO	ALTER	Automatic
YES	READ	Automatic	YES	UPDATE	Automatic

Note:

1. "No check" means that no check is made.
2. "Automatic" means that you have automatic access to the catalog if you can access the data sets.

Table 34 on page 556 is based on the following assumptions:

- The data set is a user data set.
- Both the data set and the catalog are RACF-protected.
- You have the indicated access authority to the source and target data sets.
- If an SMS-managed data set is going to be cataloged in the master catalog, you have UPDATE access to that catalog.

Table 34 on page 556 also applies to group data sets. To create a group data set by restoring it, you need one of the following:

- UPDATE access to the catalog for unprotected, non-SMS, group data sets. For unprotected, SMS-managed data sets, you automatically have access to the user catalog.
- CREATE authority in the GROUP and UPDATE access to the data set.
- ALTER access to the data set and be a member of the group that owns the data set. To discretely protect a group data set, you need additional authority such as CREATE in the group.

Restore and data set profile considerations

If the data set being restored was RACF-indicated when it was dumped, DFSMSdss does special target data set processing.

Restore and the MENTITY keyword

You can restore a data set and use the MENTITY keyword to request that DFSMSdss is to define a data set profile when one of the following is true:

- The data set to be restored was RACF-protected by a generic profile when it was dumped, and the target data set name is unprotected.
- The data set to be restored was RACF-indicated when it was dumped, and the target data set name is either unprotected or it is only protected by a generic profile.

DFSMSDss uses the data set profile specified by the MENTITY keyword, and any volume serial specified with MVOLSER, as a model to define a discrete profile to protect the target data set. If MVOLSER is not specified, the volume serial is one of the following:

- The volume on which a non-VSAM data set resides.
- The volume that contains the catalog for a VSAM data set.

To define discrete profiles for data sets that you do not own, you need additional authorization such as the system-OPERATIONS attribute or CREATE in the group.

Restore and physical data sets

DFSMSDss provides limited support for defining discrete profiles during physical restore of a data set:

- For VSAM data sets, a discrete profile is never defined.
- For non-VSAM data sets, a discrete profile is only defined if MENTITY is specified and the data set was RACF-indicated when it was dumped.

Restore and define discrete profile summary

Table 35 on page 558 summarizes the circumstances during a restore when MENTITY is effective, when a discrete profile is defined, when the target data set is RACF-indicated, and when related messages are issued.

Protection		MENTITY Specified?	Define Profile?	RACF Indicated?	Warning Message?
Source	Target				
None	None	Yes	No	No	No
		No	No	No	No
None	Generic	Yes	No	No	No
		No	No	No	No
None	RACF Indicated	Yes	No	Yes	No
		No	No	Yes	No
Generic	None	Yes	Yes	Yes	No (W)
		No	No	No	Yes
Generic	Generic	Yes	No	No	No
		No	No	No	No
Generic	RACF Indicated	Yes	No	Yes	No
		No	No	Yes	No
RACF Indicated	None	Yes	Yes	Yes	No (W)
		No	No	Yes	Yes
RACF Indicated	Generic	Yes	Yes	Yes	No (W)
		No	No	Yes	Yes
RACF Indicated	RACF Indicated	Yes	No	Yes	No
		No	No	Yes	No

Table 35. DFSMSdss Copy and Define Profile Summary. This table summarizes when DFSMSdss defines discrete profiles for a copy operation. (continued)

Protection		MENTITY Specified?	Define Profile?	RACF Indicated?	Warning Message?
Source	Target				
Note: "No (W)" means that if the define of the discrete profile fails, the restore is successful, but the target data set remains RACF-indicated. A warning message is issued.					

Protection states after a restore

When a data set is restored, the protection state of the target depends on the initial protection states of the source and the target data sets, whether MENTITY is specified, and whether discrete profiles are predefined. The following conditions may exist:

The data set was unprotected when it was dumped.

The target data set may or may not be protected after the restore. Even if you specify MENTITY, a discrete profile is not defined.

The data set was generically protected when it was dumped.

The target data set name may or may not be protected after the restore:

- If the target data set is not already protected and MENTITY is not specified, the data set remains unprotected. If MENTITY is specified, a discrete profile is defined:
 - If the define is successful, the data set is RACF-indicated. It is accessible according to the access list for the model profile that was used.
 - If the define is unsuccessful, the data set remains RACF-indicated. It may be inaccessible until a data set profile is successfully defined.
- If the target data set is already protected, a discrete profile is not defined (even if you specify MENTITY). The target data set is accessible according to the access list of a predefined, protecting, data set profile.

The source data set was RACF-indicated when it was dumped.

The target data set is RACF-indicated after the data set is restored:

- If the target data set is not already RACF-indicated and if MENTITY is not specified, a discrete profile is not defined. If MENTITY is specified, DFSMSdss defines a discrete profile:
 - If the define is successful, the data set is accessible according to the access list for the model profile that was used.
 - If the define is unsuccessful, the data set may be inaccessible until a data set profile is successfully defined.
- If the target data set is already RACF-indicated, a discrete profile is not defined by DFSMSdss (even if you specify MENTITY). The target data set remains RACF-indicated.
- DFSMSdss gives you the ability to change its default operation. Through the ADRPATCH Serviceability Aid, DFSMSdss logical restore can be instructed not to turn on the RACF indicator for the target data set when the source data set was RACF-indicated at dump time. The MENTITY keyword is not specified and the target data set is protected by a generic profile.

Related reading: For more information about changing the DFSMSdss default operation, see [Chapter 14, “DFSMSdss patch area,”](#) on page 213.

RESTORE command and the IMPORT keyword

By default, anyone can use the IMPORT keyword of the RESTORE command to bypass access checking of source data sets. You still need access authorization to create or update the target data sets and catalogs. Your installation may limit use of the IMPORT keyword as previously discussed in [“Protecting the usage of DFSMSdss”](#) on page 535.

Chapter 22. Data integrity—serialization

Volume and data sets

DFSMSDss uses volume serialization and data set serialization functions to ensure that data sets are not modified during the processing of DFSMSDss commands. Volume serialization is accomplished by using the RESERVE macro. Data set serialization is accomplished by using the ENQ macro and the DFSMSDss DYNALLOC function. In the case of shared DASD, volume serialization ensures that data sets are not being added, deleted, renamed, or extended from either the same processor or other processors.

You can control volume serialization with the enqueue installation exit routine. This allows other programs to access volumes while DFSMSDss is running. Volume serialization, however, cannot be overridden.

DFSMSDss supports data sets that are always open, or open for long periods of time, with backup-while-open serialization.

DFSMSDss supports backup-while-open processing for DFSMStvs and for two types of database applications: Customer Information Control System (CICS) and information management system (IMS).

CICS support includes both RLS CICS and non-RLS CICS backup-while-open.

DFSMStvs supports backup-while-open with or without CICS, as [“Backup-while-open data sets \(CICS and DFSMStvs\)”](#) on page 570 describes.

DFSMSDss also supports backup-while-open serialization for IMS data sets for:

- Indexed VSAM data sets, such as KSDS
- Non-indexed VSAM data sets, such as ESDS
- Non-VSAM data sets, such as OSAM

z/OS UNIX files

DFSMSDss interfaces with z/OS UNIX System Services via Virtual File System (VFS) application programmer's interface as a VFS server. This interface allows DFSMSDss to specify a particular type of access (reading, writing, or both) that is desired when a regular file is opened for access. The interface also allows DFSMSDss to specify shared reservations to prohibit concurrent access to the file by other applications or users. This interface allows such serialization for regular files only, user data integrity is assured by use of this mechanism.

The type of access DFSMSDss obtains depends on the function being performed. The following table describes the level of access:

<i>Table 36. Level of access obtained by DFSMSDss</i>				
FUNCTION	Request READ intent	Request WRITE intent	Deny shared Readers	Deny shared Writers
DUMP	YES	NO	NO	YES*
RESTORE	NO	YES	YES	YES

* - DFSMSDss wants to deny shared writers while it has the regular file open during dump operations so that the integrity of the file is preserved while backing up the user data. Denying shared writers during dump processing can be bypassed by specifying the TOLERATE(WRITERS) keyword. When specifying TOLERATE(WRITERS), you are required to guarantee consistency by ensuring the file is not updated during the dump.

Note:

1. If the desired serialization cannot be obtained, the file is not processed.

2. Other DFSMSdss supported file types are susceptible to change while processing is taking place. DFSMSdss only processes file attributes for these files (including ACLs).

Related reading: For more information about

- Enqueue installation exit routine, see [z/OS DFSMS Installation Exits](#).
- RLS CICS and non-RLS CICS backup-while-open support, see [“Backup-while-open data sets \(CICS and DFSMSStvs\)”](#) on page 570.
- IMS backup-while-open support, see [“Backup-while-open data sets \(IMS\)”](#) on page 574.
- DFSMSStvs, see [z/OS DFSMSStvs Planning and Operating Guide](#) and [z/OS DFSMSStvs Administration Guide](#).

Volume serialization

For volume serialization, DFSMSdss issues the RESERVE macro against the volume to prevent direct access device storage management (DADSM) functions (such as addition, deletion, extension, or renaming of data sets) from changing the VTOC entries during DFSMSdss processing. The RESERVE is issued before processing is begun on a particular volume, and released when processing is completed, during the following operations:

- DUMP (except logical)
- CONSOLIDATE
- COPY (except logical)
- RESTORE (except logical and physical data set)
- PRINT (except data set)
- DEFRAg
- SPACEREL

Note: Be aware that volume serialization (RESERVE performed against the volume) does not ensure integrity at the data set level. For example, in the case of a physical full volume dump, data sets are not serialized and the data sets could change while the DUMP is being taken.

You can use the DFSMSdss enqueue installation exit to request that DFSMSdss only reserve the VTOC for the duration of VTOC access during the following operations:

- Full, tracks, and physical data set DUMP
- Full and tracks COPY
- Tracks PRINT

In addition, DFSMSdss issues the RESERVE macro against a volume while updating information in the VTOC, such as the RACF-defined flag and the data-set-changed flag. DFSMSdss also issues the RESERVE macro against a volume while accessing the information in the VTOC to perform data set selection during the following operations:

- Logical DUMP with input volumes specified
- Logical COPY with input volumes specified
- CONVERTV
- COMPRESS
- RELEASE

For more information about the enqueue installation exit routine, see [z/OS DFSMS Installation Exits](#).

Avoiding lockout

The VTOC is locked to prevent activity on the VTOC at the volume level during the copy, defrag, dump, print (tracks) or restore (tracks and full) functions. These functions may require access to the catalogs for the data sets on the volume. A lockout can result between DFSMSdss and another job on the same system that has already locked the catalog but needs the VTOC for a DADSM function. To avoid this lockout,

you should not run any jobs (for example, Access Method Services jobs) that require control of both the catalog and the VTOC while DFSMSdss is executing.

The remainder of this section provides intended programming interfaces.

When an application program attaches a DFSMSdss dump or restore task and a task that invokes dynamic allocation to allocate a data set (for example, logical restore operation) or invokes DADSM to scratch a data set, the two tasks could lock each other out. To avoid this lockout, DFSMSdss uses the following ENQ scheme:

- Before DFSMSdss invokes DADSM SCRATCH, an exclusive ENQ is requested for the resource with a major name of SYSZADRV and a minor name of the volume serial number (padded with blanks to 8 bytes) with a scope of SYSTEMS.
- Before DFSMSdss invokes dynamic allocation to allocate a new data set on a specific volume, a shared ENQ is requested for the resource with a major name of SYSZADRV and a minor name of the volume serial number (padded with blanks 8 bytes) with a scope of SYSTEMS. If this is a nonspecific dynamic allocation request, an exclusive ENQ with the resource with a major name of ADRLOCK and a minor name of NONSPEC with a scope of STEP is requested.
- If an application program invokes a DFSMSdss FULL dump operation and plans on attaching another DFSMSdss logical restore task, the application should request the first resource in exclusive mode (SYSZADRV, volser,SYSTEMS). The application should request the second resource in shared mode (ADRLOCK, NONSPEC) before attaching the FULL DUMP command task. Multiple dumps can be active in the same address space, and a logical restore request will not lock out that task.
 - If an application program issues a DADSM request, the application must enqueue on SYSZADRV/volser/SYSTEMS in a shared ENQ or ADRLOCK/NONSPEC/STEP in an exclusive ENQ.

Note:

1. When DFSMSdss is invoked with JCL, lockout does not occur because DFSMSdss allocates a volume before invoking DADSM. If a FULL DUMP command has the volume enqueued and a restore task has been applied against the same volume, the restore task is held until the volume is available.
2. Only non-SMS, non-VSAM DADSM requests require serialization.

The WAIT option

DFSMSdss lets you change the WAIT/RETRY values for system resources.

Related reading: For more information about WAIT/RETRY, see [“Controlling the wait/retry time for serialization of system resources \(PN11523\)”](#) on page 215.

Data set serialization

This section describes the ENQUEUE and SERIALIZATION options for data set serialization, the WAIT option, and provides an example of RESERVE-ENQUEUE processing.

Enqueuing—ENQ

The following material describes enqueue options for data set serialization. Enqueues are done on the data set name for the data set copy, data set dump, data set restore, defragment, print, compress, and release operations to prevent multiple, simultaneous updates to the same data set.

The SHARE option has unique properties when applied to the following commands:

- For the RESTORE command, SHARE applies to all non-VSAM data sets except for PDSEs that are targets of a RESTORE operation. The SHARE keyword does not apply to VSAM data sets.
- For the DUMP and COPY commands, SHARE applies to all non-VSAM data sets, excluding PDSEs that are targets of a COPY operation. SHARE also applies to VSAM data sets that are defined with share options other than (1,3) and (1,4).

If you do not specify the SHARE option, DFSMSDss tries to provide the highest level of data integrity by defaulting to exclusive enqueueing. The command is in a wait state for *X* seconds if the enqueue fails. WAIT specifies *X* (numsecs, numretries), and retries the enqueue. If the wait-enqueue sequence fails after *Y* retries (where WAIT specifies *Y* (numsecs, numretries)), data set processing ends.

In the case of a multistep job where the initiator holds a shared enqueue on the data set, DFSMSDss upgrades the enqueue to exclusive unless SHARE is specified. The initiator holds the exclusive enqueue until the last step in the job that references the data set has completed.

If you specify the SHARE keyword, DFSMSDss tries an enqueue for share. If it fails, it goes through the same logic as if SHARE had not been specified. If the retries all fail, processing ends for the data set.

You can specify TOLERATE(ENQFAILURE) in addition to the default, ENQ, or the SHARE option. If TOLERATE(ENQFAILURE) is specified, DFSMSDss attempts to get the specified level of enqueue, exclusive or share. If the enqueue fails after the specified or default number of retries, DFSMSDss processes the data set without an enqueue even when specifying RENAME (or RENAMEU) or REPLACE (or REPLACEU). Specify TOLERATE(ENQFAILURE) if you are willing to tolerate the exposure of not having data integrity in order to force the successful completion of that particular data set operation. This is particularly useful when an installation has duplicate data sets (different data sets with the same name but on different volumes) and you want to run DFSMSDss on a data set on one volume while the data set with the same name is being used by the system or another job on a different volume.

Note: Because of ENQ contention, SYSPRINT data sets should not be allocated on volumes being processed. TOLERATE(ENQFAILURE) cannot apply to data movements that involve the use of utilities.

Table 37 on page 564 shows the data set enqueue options.

TOLERATE(ENQFAILURE) can be used on VSAM and non-VSAM data sets.

Table 37. Data Set Enqueue Options for Non-VSAM Data Sets, except for Poses that are targets of a COPY or RESTORE operation, Specified on DFSMSDss Commands

Options	None	SHARE	TOLERATE(ENQFAILURE)	SHARE and TOLERATE(ENQFAILURE)
Type of enqueue attempted	Exclusive	Share	Exclusive	Share
If enqueue is successful	Process	Process	Process	Process
If enqueue is not successful	Do not process	Do not process	Process without enqueue	Process without enqueue

Dumping HFS data sets

The serialization mechanism operates differently depending on whether you are performing a logical dump or a physical dump. Whether or not you mount the data set before you dump it affects serialization also.

Logical dump

The serialization mechanism used during a logical dump of a mounted HFS data sets consists primarily of the SYSZDSN enqueue macro and the BPX1QSE quiesce macro. A shared SYSZDSN enqueue provides serialization if a data set that you want to dump is not mounted for update. To mount an HFS data set for update, z/OS UNIX System Services (z/OS UNIX) must obtain an exclusive enqueue on the SYSZDSN. The shared enqueue on the SYSZDSN prevents z/OS UNIX from mounting the data set for update during the dump.

If you mount an HFS data set for update before you dump it, DFSMSDss issues the BPX1QSE macro to prevent updates to the data set for the duration of the dump. For non-shared file systems, you must run

the DFSMSdss dump job on the same system where the data set is mounted for update. For shared file systems, you can run the DFSMSdss dump job from any system in the sysplex.

DFSMSdss can honor the DELETE keyword for an HFS data set during logical dump only if the data set is unmounted. For delete processing, the data set must be enqueued exclusively.

Physical dump

DFSMSdss relies on a SYSDSN enqueue for physical dump operations. By default, DFSMSdss attempts an exclusive SYSDSN enqueue during physical dump. As long as the data set is not mounted at dump time, the exclusive enqueue on the SYSDSN provides sufficient serialization. If the data set is mounted before the physical dump job begins, z/OS UNIX will have a shared enqueue on the SYSDSN. DFSMSdss will therefore not be able to obtain an exclusive SYSDSN enqueue.

Avoid performing a physical dump of mounted HFS data sets because quiesce is not available during a physical dump. If you specify either the SHARE or TOL(ENQF) keyword during a physical dump, the internal control information and data inside the HFS can change during the dump. This can result in a dumped data set that contains an HFS data set that might not be usable after you have restored it. If you must physically dump a data set that is in use, use TOL(ENQF) instead of SHARE. With TOL(ENQF), you receive a return code of four, along with a warning message, if serialization was not adequate during the dump.

Guideline: Exercise caution when 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 can likely result in an unusable data set.

zFS data sets

The serialization mechanism operates differently depending on whether you are performing a physical operation or a logical operation. Serialization is also affected if the data set is mounted.

Logical copy or dump

The serialization mechanism used during a logical copy or dump of zFS data sets consists primarily of the SYSDSN enqueue macro, SYSVSAM enqueue macros and the zFS quiesce macro. A shared SYSDSN enqueue and SYSVSAM enqueues provide serialization if a data set that you want to copy or dump is not mounted.

If a zFS data set is mounted before you copy or dump it, the data set can still be quiesced. The quiesce action prevents updates to the data set for the duration of the copy or dump.

For non-shared file systems, you must run the DFSMSdss copy or dump job on the same system where the data set is mounted for update. For shared file systems, you can run the DFSMSdss job from any system in the sysplex.

For more information, including restrictions that apply to enqueues and dequeues, refer to [“Serialization” on page 151](#).

Physical dump

DFSMSdss relies on a SYSDSN and SYSVSAM enqueues for physical dump operations. By default, DFSMSdss attempts exclusive enqueues during physical dump. As long as the data set is not mounted at dump time, the exclusive enqueues provide sufficient serialization. If the data set is mounted before the physical dump job begins, z/OS UNIX will have the zFS data set allocated and opened. Allocation and open processing will have obtained SYSDSN and SYSVSAM enqueues. DFSMSdss will therefore not be able to obtain the exclusive enqueues.

Physical dump of mounted zFS data sets is not recommended because quiesce is not available during a physical dump. If you specify the TOL(ENQF) keyword during a physical dump, the internal control information and data inside the zFS can change during the dump. This can result in a dumped data set that contains a zFS data set that might not be usable after you have restored it.

Physical data set copy

Physical data set copy operations require DFSMSdss to obtain the SYSDSN and VSAM enqueues for the source data set. In order to allow this type of serialization the zFS will need to be unmounted from all systems.

Physical data set copy of mounted zFS data sets is not recommended because quiesce is not available during a physical copy. If you specify the TOL(ENQF) keyword during a physical data set copy, the internal control information and data inside the zFS can change during the copy. This can result in a data set copy that is unusable.

Dynamic allocation (DYNALLOC)

DYNALLOC is an option for the data set DUMP, data set COPY, data set RESTORE, data set PRINT, DEFRAG, COMPRESS, and RELEASE commands. It allows serialization of data sets across processors with shared DASD, with JES3, and with the interface between dynamic allocation and JES3 enabled. Dynamic allocation is used by DYNALLOC to serialize data sets. Processing time increases because overhead is involved in dynamic allocation and serialization across multiple processors.

If you use DYNALLOC to serialize data sets (as opposed to ENQ) the job run time increases. If you use INDYNAM instead of DD statements to allocate DASD volumes you do not appreciably increase run time and coding of JCL, and command input is easier.

DFSMSdss does *not* honor DYNALLOC for HFS source data sets for either logical data set copy or logical data set dump.

Enqueuing versus dynamic allocation of data sets

For data set operations, the default method of serializing usage of data sets is to enqueue on the data set name. If the DYNALLOC keyword is coded in the control cards, the time taken to serialize is much greater than that taken by using the ENQ macro. So use DYNALLOC keyword judiciously. Use it only on a JES3 system if the interface between the allocation function and JES3 is not disabled. If the interface is disabled, then the end result of serialization using ENQ is the same as using dynamic allocation; ENQ takes much less time to serialize data sets.

When you use DYNALLOC keyword, the selected data sets are allocated to DFSMSdss. If some data sets are allocated to the system, they can only be processed by DFSMSdss with DYNALLOC if SHARE is also specified. [Table 39 on page 568](#) provides more information on data set serialization.

Read/Write serialization scheme

[Table 38 on page 566](#) shows the serialization scheme used by all operations, except COPYDUMP, for read/write access.

Table 38. Read/Write Access Serialization Scheme				
Item	ENQ Major Name	ENQ Minor Name	ENQ Control, Exclusive (E) or Share (S)	ENQ Scope
For Source HFS Data Sets:				
followed by:	ADRDSN	Data set name	E (see Note 1)	STEP
	SYSZDSN	Data set name	S (see Note 1) E (see Note 2) S (see Note 2)	SYSTEMS
For zFS Data Sets:				
On the component during logical data set DUMP or COPY operations:	SYSVSAM	See Note 3	E (see Note 5) S (see Note 5)	SYSTEMS

Table 38. Read/Write Access Serialization Scheme (continued)				
Item	ENQ Major Name	ENQ Minor Name	ENQ Control, Exclusive (E) or Share (S)	ENQ Scope
On the component during physical data set DUMP operations:	SYSVSAM	See Note 3	E (see Note 1) S (see Note 1)	SYSTEMS
On the component during other operations:	SYSVSAM	See Note 3	E	SYSTEMS
On the cluster during logical DUMP or COPY operations:	ADRDSN	Cluster name	E (see Note 1) S (see Note 1) E (see Note 5) S (see Note 5)	STEP
followed by:	SYSDSN	Cluster name		SYSTEM
On the cluster during other operations:	ADRDSN	Cluster name	E (see Note 1) S (see Note 1) E (see Note 1) S (see Note 1)	STEP
followed by:	SYSDSN	Cluster name		SYSTEM
For Non-VSAM Data Sets:				
followed by:	ADRDSN	Data set name	E (see Note 1) S (see Note 1)	STEP
	SYSDSN	Data set name	E (see Note 1) S (see Note 1)	SYSTEM
For GDG Data Sets:				
On the GDG base:	ADRDSN	GDG base name	E (see Note 1) S (see Note 1)	STEP
followed by:	SYSDSN	GDG base name	E (see Note 1) S (see Note 1)	SYSTEM
On each GDG data set:	ADRDSN	Data set name	E (see Note 1) S (see Note 1)	STEP
followed by:	SYSDSN	Data set name	E (see Note 1) S (see Note 1)	SYSTEM
For VSAM Data Sets:				
On each component during data set DUMP or COPY operations:	SYSVSAM	See Note 3	E (see Note 1) S (see Note 1)	SYSTEMS
On each component during other operations:	SYSVSAM	See Note 3	E	SYSTEMS
On the cluster:	ADRDSN	Cluster name	E (see Note 1) S (see Note 1)	STEP
followed by:	SYSDSN	Cluster name	E (see Note 1) S (see Note 1)	SYSTEM
For Integrated Catalog Facility Catalogs:				
In addition to the above VSAM data set information:	SYSIGGV2	Catalog name	E	SYSTEMS
On the volume:	SYSVTOC	Volume serial number	E	SYSTEMS
For VVDS Data Sets:				

Table 38. Read/Write Access Serialization Scheme (continued)

Item	ENQ Major Name	ENQ Minor Name	ENQ Control, Exclusive (E) or Share (S)	ENQ Scope
On each data set: Input access: Output access:	SYSZVVDS	SYS1.VVDS.Vvolser	S E	SYSTEMS

Note:

1. If the selected control for the ENQs is EXCLUSIVE or SHARE, the SHARE keyword determines the type of control. If SHARE is *not* specified, exclusive control of the data set is obtained. Whereas, if SHARE is specified, shared control of the data set is obtained.

If shared control is obtained for a VSAM data set because the SHARE keyword was specified, other programs are able to obtain read access, but not write access, to the data set, while it is being processed. The SHARE keyword is honored for VSAM data sets only during DUMP and COPY processing, and only for VSAM data sets that are defined with SHARE options other than (1,3) or (1,4).
2. For a source HFS data set, DFSMSdss ignores DYNALLOC. If you specify DELETE, then DFSMSdss attempts to get an exclusive SYSZDSN enqueue. If you do not specify DELETE, then DFSMSdss attempts to get a shared SYSZDSN enqueue.
3. If DYNALLOC is used:

For a source HFS data set, DFSMSdss ignores DYNALLOC.

For non-VSAM data sets and GDG bases, instead of ENQ, the data set is dynamically allocated automatically, with a disposition of OLD if SHARE is not used; otherwise, SHARE is the disposition.

For VSAM data sets, in addition to the ENQ on the components, the cluster is allocated dynamically just as for non-VSAM data sets.
4. The minor name used for enqueueing VSAM components consists of the following: component name, catalog name, L1, L2, L3, A (where L1 is the total length of the minor name, L2 is the component name length, and L3 is the catalog name length).

On a data set DUMP, data set RESTORE, data set COPY, and DEFRAG operation, an enqueue is performed once with the character A=I and iteratively with A=O.
5. If you specify DELETE, then DFSMSdss attempts to get exclusive control of the zFS. If you do not specify DELETE, then DFSMSdss attempts to get a shared control of the zFS.

Programming Interface information: Although the ENQ on the major name ADRDSN is mostly intended to coordinate access to data sets between multiple DFSMSdss commands, it might be necessary for application programs that invoke DFSMSdss to make use of the ENQ; data sets that are serialized by the application program using dynamic allocation or an ENQ on the major name SYSDSN could be processed by DFSMSdss unless the application uses the ENQ on ADRDSN.

Table 39. Resource Serialization

Function	Data Set Type	Volume Level Serialization VTOC VVDS	Data Set Level Serialization (ENQ or DYNALLOC)
COMPRESS	Non-VSAM	Yes N/A	DSName
CONSOLIDATE	Non-VSAM VSAM	Yes N/A Yes Yes	DSName Component and Cluster Names
CONVERTV	Non-VSAM	Yes N/A	DSName

Table 39. Resource Serialization (continued)				
Function	Data Set Type	Volume Level Serialization VTOC VVDS		Data Set Level Serialization (ENQ or DYNALLOC)
Data Set COPY	Non-VSAM	Yes	N/A	DSName
	VSAM	Yes	Yes	Component and Cluster Names
DEFRAG	Non-VSAM	Yes	N/A	DSName
	VSAM	Yes	Yes	Component and Cluster Names
Full DUMP	Non-VSAM	Yes	N/A	N/A
	VSAM	Yes	Yes	N/A
Data Set DUMP	Non-VSAM	Yes	N/A	DSName
	VSAM	Yes	Yes	Component and Cluster Names
Data Set PRINT	Non-VSAM	Yes	N/A	DSName
	VSAM	Yes	Yes	Component Name
Tracks PRINT	N/A	Yes	N/A	N/A
RELEASE	Non-VSAM	Yes	N/A	DSName
	VSAM	Yes	Yes	Component and Cluster Names
Full RESTORE	Non-VSAM	Yes	N/A	N/A
	VSAM	Yes	Yes	N/A
Data Set RESTORE	Non-VSAM	N/A	N/A	DSName
	VSAM	N/A	Yes	Component and Cluster Names

Notes for [Table 39 on page 568](#):

- Refer to [“Backup-while-open data sets \(CICS and DFSMSdss\)” on page 570](#) for more information on resource serialization.
- VSAM data sets must be cataloged in an integrated catalog facility catalog.
- N/A means not applicable.

WAIT option

This option allows you to specify how long, in seconds, DFSMSdss is to wait for a resource and the number of times DFSMSdss is to retry, should an ENQ or RESERVE fail. The default is WAIT(2,2).

This means DFSMSdss is to retry twice at 2-second intervals. The WAIT keyword does not apply to system resources such as the VTOC and the VVDS.

For a data set copy or logical data set dump operation, the WAIT option has a different meaning for serializing data sets when multiple data sets are to be processed and WAIT(0,0) is not specified. Multiple passes are made through the list of data sets that are selected. On each pass, those data sets that can be serialized without waiting for the resource and that were not processed before are processed. At the end of a pass, if none of the data sets could be processed without waiting for a resource, then, in the next pass, at the first occurrence of a data set that was not processed a WAIT is issued.

That data set and the remainder of the list is processed if possible. The above procedure is repeated until all data sets are processed or the WAIT limits are reached.

For example, if WAIT(3,10) is specified and 5 data sets are left to be processed, up to 10 passes are made. On each pass, the first unprocessed data set is waited upon for 3 seconds. Thus, only a 30-second maximum is ever waited, not 150 (5 times 3 times 10).

For system resources, such as the VTOC and the VVDS data set, the default is WAIT(3,30). This means DFSMSdss is to retry 30 times at 3 second intervals, resulting in a total wait time of 90 seconds.

An example of RESERVE-ENQUEUE processing

If you were to enter the following:

```
DUMP INDD(IN1) OUTDD(OUT1) -  
DATASET(INCLUDE(MY.DUMP.DATASET1,MY.DUMP.DATASET2)) -  
WAIT(5,2) -  
SHARE
```

The following would result:

1. DFSMSdss issues a RESERVE command on SYSVTOC (this is the default)
 - If the reserve operation fails, DFSMSdss retries thirty times at 3-second intervals, assuming that the installation has not changed the default for system resources.
 - If the RESERVE command is successful, DFSMSdss continues.

If either of the data set was a VSAM or SMS-managed data set, the VVDS would also be serialized by DFSMSdss. If VVDS serialization fails, none of the data sets are dumped.
2. DFSMSdss issues a shared enqueue on data set name. If the shared enqueue fails, DFSMSdss retries two times at 5-second intervals.
 - If the shared enqueue is successful:
 - DFSMSdss adds the data set name to the list to be dumped.
 - DFSMSdss loops back to the beginning of this step until an enqueue is tried for all data sets.
 - If the shared enqueue fails, DFSMSdss:
 - Issues a message indicating that the data set failed serialization, and does not process the data set.
 - Loops back to the beginning of this step until an enqueue is tried for all data sets.
3. After an enqueue is tried for all specified data sets and at least one was successful, DFSMSdss:
 - Dumps the VVDS (if an integrated catalog facility data set was selected) and the VTOC and dequeues the VTOC.
 - Dequeues VVDS if it was enqueued.
 - Dumps each data set that was successfully enqueued.
 - Dequeues the enqueued data sets.
 - Issues message indicating which data sets were successfully processed.

Note: If you specify TOLERATE(ENQFAILURE) option with the DUMP command and the installation options exit routine does not override it, the VTOC is not dequeued until all the data tracks for all the data sets are dumped.

Backup-while-open data sets (CICS and DFSMStvs)

DFSMSdss supports backup-while-open serialization, which can perform backup of data sets that are open for update for long periods of time. It can also perform a logical data set dump of these data sets even if another application has them serialized. Backup-while-open is a better method than using SHARE or TOLERATE(ENQFAILURE) for dumping Customer Information Control System (CICS) VSAM

file-control data sets that are in-use and open for update. When you dump data sets that are designated by CICS as eligible for backup-while-open processing, data integrity is maintained through serialization interactions between CICS (data base control program), CICSVR (forward-recovery program), VSAM record management, DFSMSdftp, and DFSMSdss. When you dump data sets that are designated by DFSMStvs as eligible for backup-while-open processing, data integrity is maintained through serialization interactions between DFSMStvs, VSAM record management, DFSMSdftp, and DFSMSdss.

Although the BWO(TYPECICS) parameter applies to both CICS and DFSMStvs, DFSMStvs enables you to back up a data sets while they are open whether or not you are running CICS.

Figure 26 on page 571 shows the backup-while-open serialization for dumping CICS data sets that are open for update. Backup-while-open processing also ensures that any update activity that may invalidate the dump is detected. Simultaneous recovery or deletion of the data set while it is being dumped is also prevented.

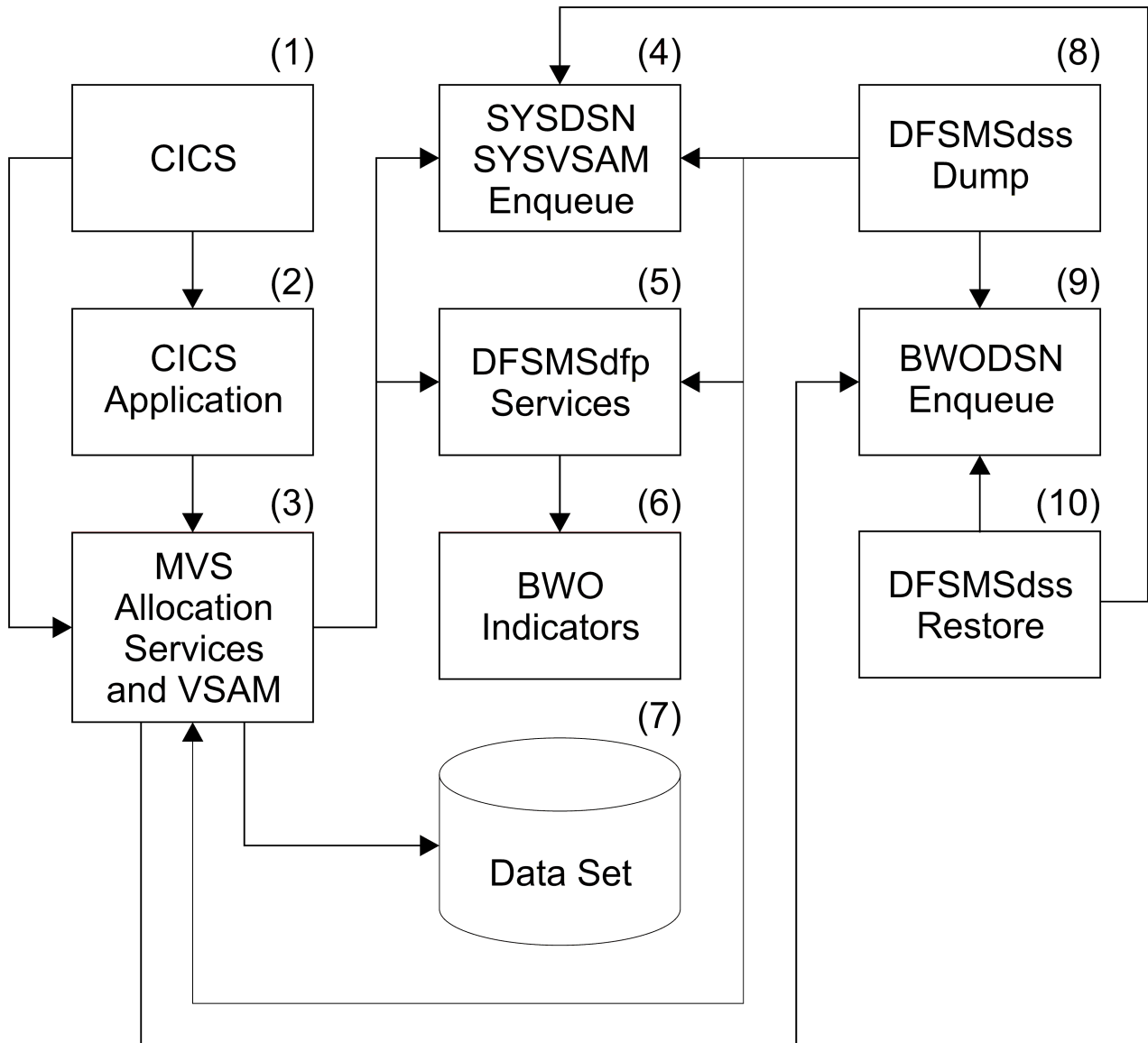


Figure 26. Block Diagram for Backup-While-Open Serialization

In Figure 26 on page 571, a VSAM file-control data set (7) is allocated for CICS (1) through MVS allocation services (3) using JCL or dynamic allocation methods. This results in serialization through an enqueue on the name of the data set and the resource name SYSDSN (4). When the VSAM data set is opened (3), another level of serialization occurs through an enqueue on the names of the components of the VSAM

data set and the resource name SYSVSAM (4). For eligible data sets, CICS uses DFSMSdfp (5) to set a status in the backup-while-open indicators (6) in the catalog entry for the data set.

For a dump operation, DFSMSdss (8) attempts to acquire the SYSDSN, SYSVSAM, and backup-while-open (9) enqueues for the data set. When the enqueue on the cluster name of the data set and resource name of BWODSN is acquired, but not the enqueues for both SYSDSN and SYSVSAM, DFSMSdss uses DFSMSdfp to get the backup-while-open indicators, and starts to dump the open data set if it is backup-while-open eligible.

The backup-while-open enqueue is used to prevent more than one DFSMSdss operation, such as a simultaneous dump and restore (10), and to prevent the data set from being deleted while it is being dumped by DFSMSdss.

While the data set is being dumped, a data base application program may update the data set in a manner that invalidates the data set. For example, a control-interval or control-area split may occur. When this happens, VSAM record management uses DFSMSdfp to change the backup-while-open status. When the backup of the open data set is completed, DFSMSdss obtains the current backup-while-open indicators and invalidates the dump of the data set if the indicators are different from when the dump was started. When concurrent copy is used, updates made while the data set is being dumped do not cause the dump to be invalidated.

Backup-while-open status definition

DFSMSdss interprets backup-while-open status numbers as follows:

Status

Meaning to DFSMSdss

000

Normal serialization and processing techniques are used to dump the data set.

001

CICSVR forward-recovers the data set.

010

A control-interval split, control-area split, or extend of the data set was either interrupted before the dump started or is currently in process.

011

A control-interval split, control-area split, or an extend of the data set completed successfully. CICS or DFSMSStvs closed the data set and there is no mismatch between the base cluster and any alternate index.

100

The data set may be dumped while it is open for update.

101

The data set was restored and is down-level; CICSVR or DFSMSStvs must process it before a database application uses it.

110

The data set was updated and a split or extend is completed, which invalidates any dump that was in progress. When concurrent copy is used, the chances of an in-progress-dump being invalidated by a split or extend is significantly reduced.

111

The data set is in an indeterminate state.

Backup-while-open processing

DFSMSdss actions resulting from dumping a data set are based on the following conditions:

- Success or failure to acquire an exclusive enqueue for the resource names of SYSDSN, SYSVSAM, and BWODSN.
- Backup-while-open status *before* the dump.

- Backup-while-open status indicators after the dump.

When DFSMSdss acquires all of the exclusive enqueues: The data set is not open for update, even though it has a non-zero backup-while-open status. The particular processing action is a direct result of the initial backup-while-open status as follows:

- Status is 000; the data set is dumped. When it is restored, the backup-while-open status is restored as 000.
- Status is 001. The data set is dumped and the backup-while-open status is preserved and restored when the data set is subsequently restored. This data set is in an indeterminate state because of an incomplete forward recovery by CICSVR, which cannot forward-recover the restored data set.

DFSMSdss lets you dump and restore this data set for nonstandard recovery purposes. After restoring the data set and correcting all errors in the data set, reset the BWO status to 000 by using CICS-provided methods. The preferred action is for you to restore an earlier dump of the data set and use CICSVR to do forward-recovery processing. This maintains the integrity of the data.

- Status is 010. DFSMSdss did not dump the data set. Take corrective action before using the data set by performing the following actions:
 - Determine if alternate indexes (AIX) are present for the sphere. If they are, do either of the following tasks:
 - Rebuild the AIXs from the base cluster, and reset the BWO status to 000 by using the CICS' methods.
 - Restore an earlier dump of the sphere, and use CICSVR to do forward-recovery processing.
 - If there are no AIXs, the data set is usable, as it is. Reset the BWO status to 000 by using the CICS methods.
- Status is 101. The data set is dumped and the backup-while-open status is preserved and restored when the data set is subsequently restored. This lets you process the data set with CICSVR before a data base application uses it.
- Status is 011, 100, 110, 111. The backup-while-open status is altered to 000 before the data set is dumped. When it is restored, the backup-while-open status is restored as 000.

When DFSMSdss acquires an exclusive enqueue on BWODSN (but not SYSDSN and SYSVSAM): Another program is using the data set or the data set is open. The particular processing action is a direct result of the initial backup-while-open status as follows:

- Status is 000. The data set is not eligible for backup-while-open. The data set is not dumped unless SHARE or TOLERATE(ENQFAILURE) is specified and the necessary conditions for those keywords are met.
- Status is 001, 010, 011, 101, or 111; the data set is not dumped.
- Status is 110; the backup-while-open status is altered to 100 and the data set is dumped.
- Status is 100. The data set is dumped, even though it is already in use (including being open for update by another program). When the data set is restored, the backup-while-open status is set to 101. This ensures that CICSVR or DFSMSdss can process the data set, prior to its use by a data base application. The dump is invalidated if the backup-while-open indicators change while the data set is being dumped. The chances of this invalidation occurring reduce significantly when you use concurrent copy.

Backup-while-open and concurrent copy

Concurrent copy improves backup-while-open processing by significantly reducing the chances of the invalidation of a backup-while-open dump because of updates to the data set. To use concurrent copy, specify the CONCURRENT keyword when you dump backup-while-open data sets. The following is a comparison of the various kinds of dumps you can ask for:

- **Normal dump.** Use of the data set must be quiesced. DFSMSdss obtains serialization, dumps the data set, and then releases the serialization. The data set cannot be used for the entire time.

- **Concurrent copy dump.** Use of the data set must be quiesced. DFSMSDss obtains serialization, performs concurrent copy initialization, releases serialization, and then dumps the data set. DFSMSDss completes concurrent copy initialization within a very short time (compared to the actual time to dump the data set). DFSMSDss can use the data set as soon as the concurrent copy initialization is complete.
- **Backup-while-open dump.** Use of the data set does not need to be quiesced. DFSMSDss dumps the data set without obtaining serialization. The data set can remain in use for the entire time, but update activity can invalidate the dump at any time during the dump.
- **Backup-while-open dump using concurrent copy.** Use of the data set does not need to be quiesced. DFSMSDss does not obtain serialization. DFSMSDss performs the concurrent copy initialization. DFSMSDss completes concurrent copy initialization within a very short time (compared to the actual time to dump the data set). The data set can remain in use for the entire time. Like the Backup-while-open dump, update activity during the dump can invalidate the dump. However, only update activity that occurs *before* DFSMSDss performs the concurrent copy initialization can invalidate the dump. The chances of the update activity invalidating the dump are significantly reduced because the concurrent copy initialization completes very quickly.

TOLERATE (ENQFAILURE) and SHARE considerations

Using TOLERATE(ENQFAILURE) modifies the processing and serialization for backup-while-open data sets. The data sets are dumped when the backup-while-open status is 100, even though none of the enqueues are successfully acquired.

To maintain data integrity and data security, do not specify either SHARE or TOLERATE(ENQFAILURE) when you dump backup-while-open data sets.

An exclusive enqueue on BWODSN resource name for the data set is required for DFSMSDss to alter the backup-while-open status. DFSMSDss only attempts an exclusive enqueue on the BWODSN resource name. Unless the backup-while-open status is already 100, the dump fails even though you specified SHARE or TOLERATE(ENQFAILURE).

CICS recovery data

CICS maintains recovery data in the form of a date and time-stamp in the catalog entry for backup-while-open data sets and RLS data sets. This information is not used or processed by DFSMSDss, but it is dumped and restored to preserve that information for CICSVR. The recovery data is also printed in selected messages to assist you in your recovery efforts.

DFSMSDss maintains recovery data in forward recovery log records. DFSMSDss does not use or process this recovery data. The data is dumped and restored to preserve the information for CICSVR or possibly for another forward recovery utility. The recovery data is also printed in selected messages to assist you in your recovery efforts.

Backup-while-open data sets (IMS)

DFSMSDss supports backup-while-open processing of IMS data sets by providing for fast replication copies and concurrent copy dumps. IMS requests backup through the application programming interface of DFSMSDss, described in [Chapter 23, “Application programming interface,” on page 577](#).

Backup-while-open serialization is applicable to HISAM, SHISAM, and index (primary and secondary) databases.

Backup as an open data set for IMS is triggered through a UIM request, rather than through a backup-while-open status or BWO(TYPEIMS) definition as CICS does.

Specifics of IMS backup-while-open support are outlined in [Table 40 on page 575](#).

Note: During a logical data set dump, you must specify VALIDATE except for encrypted VSAM data sets, (directly or by default) to ensure that an IMS backup-while-open data set that is dumped while updates are being made can be successfully restored. VALIDATE allows DFSMSDss to validate and correct the data

set during the dump process, or to end the dump (with an ADR943E message) if the data set is in an unrecoverable state.

DFSMSDss issues an ADR943E message during logical data set dump if the NOVALIDATE parameter is used with an indexed VSAM data set defined as BWO(TYPEIMS).

Note: DFSMSDss does not support backup-while-open (IMS) for backups directed to an object storage cloud.

Table 40. DFSMSDss Support for IMS Backup-While-Open Data Sets

Topic:	For IMS data sets:
Can non-VSAM data sets be copied or dumped as open data sets?	Yes
Can non-indexed VSAM data sets be copied or dumped as open data sets?	Yes
Must indexed VSAM data sets be specifically designated as being BWO-eligible to be copied or dumped as open data sets?	No
How are indexed VSAM data sets designated as being eligible for update activity detection through a BWO counter?	By defining the data set as BWO(TYPEIMS)
How is backup as an open data set triggered?	Through a UIM request
How is bypassing of enqueues on SYSDSN and SYSVSAM managed?	Automatically



Attention: The user must be aware that COPY with DELETE, DUMP with DELETE, and RESTORE with REPLACE can cause irreparable damage to an IMS data set for the following reasons:

- The enqueue serialization obtained by COPY and DUMP are insufficient to ensure that the data set is not also being used by an IMS application in an environment where GRS (or equivalent) is not being used.
- COPY and DUMP do not provide any special handling for any data set defined as BWO(TYPEIMS).
- RESTORE does not provide any protection against reallocating or overwriting any IMS data set for which RESTORE is able to obtain an enqueue serialization on SYSDSN and SYSVSAM.

Container serialization

DFSMSDss uses an enqueue for cloud processing that serializes access to containers in an object storage cloud. The enqueue prevents multiple DFSMSDss modifiers from accessing the same container in a manner that would render competing requesters inoperable for their requested operations.

Cloudutils Delete of a container

DFSMSDss will request an exclusive enqueue using major name SYSZADRC and minor name of *cloud_name/container* with a scope of SYSTEMS.

Dump, Restore, and other Cloudutils operations

DFSMSDss will request a shared enqueue using major name SYSZADRC and minor name of *cloud_name/container* with a scope of SYSTEMS.

Object serialization

DFSMSDss uses an enqueue for cloud processing that serializes access to objects in an object storage cloud. The enqueue prevents multiple DFSMSDss writers from accessing the same object. It also prevents concurrent DFSMSDss readers and writers. The enqueue is obtained such that multiple DFSMSDss readers will be allowed. DFSMSDss will request an exclusive enqueue using major name SYSZADRO and minor name of *cloud_name/container/objectprefix* with a scope of SYSTEMS. The object prefix is specified under the new OBJECTPREFIX keyword.

Chapter 23. Application programming interface

This topic provides information that you may need when you use the user interaction modules. General-use programming interface and associated guidance information is contained in this topic. Programming interface information is also included and is explicitly marked.

Calling block structure

The parameter structure outlined below is shown in block form in [Figure 27 on page 579](#).

ADDRSSU

The name of the DFSMSdss module main entry point.

OPTPTR

The pointer to the option list and similar to OPTIONADDR described in [z/OS DFSMSdss Utilities](#). This parameter lets you specify processing options and must be specified even if the list is a null list. If you do not want to specify any parameters to DFSMSdss, this pointer must point to a halfword of binary zeros.

You can invoke DFSMSdss using the standard application interface. This API allows you to specify in the options list those values that you can specify in the EXEC PARM field when you invoke DFSMSdss using JCL. You can also invoke DFSMSdss using the cross memory application interface. The values that you can specify in the options list include those values that can be specified in the EXEC PARM field when invoking DFSMSdss with JCL. Additionally, you can specify values that are specific to the cross memory application interface. For a list of the values that can be specified in the EXEC PARM field when invoking DFSMSdss with JCL, see [“How to control DFSMSdss through PARM information in the EXEC statement” on page 249](#). For a list of the values that are specific to the cross memory application interface, see [“Using the cross memory application interface to control DFSMSdss” on page 582](#).

The first field, called the *option-list length field*, is a halfword field specifying the length of the list (excluding the field itself) as a binary value. If you do not want to specify any options, set the option-list length field to binary zeros. In your JCL for the calling program, if you code parameters (PARM=value) on the EXEC statement, DFSMSdss does not recognize them unless OPTPTR points to them.

The options must comply with the parameter syntax of the DFSMSdss EXEC parameter values. If you do not want to specify subsequent parameters, you can omit them from the list.

DDPTR

The pointer to the DDNAME list and similar to DDNAMEADDR described in [z/OS DFSMSdss Utilities](#). The DDNAME list provides a way to specify alternate names for the SYSIN and SYSPRINT data sets. The DDNAME list is a variable length field made up of a halfword field followed by unseparated 8-character, left-justified (right padded with blanks) fields. Each 8-character field is reserved for a specific DDNAME. DFSMSdss uses only two of these fields: SYSIN and SYSPRINT. The other fields are provided as a standard implementation consistent with existing system utility invocation procedures, must be filled with binary zeros, and are ignored by DFSMSdss.

The first field, called the *DDNAME-list length field*, is a halfword field specifying the length of the list (excluding the field itself) as a binary value for the number 48.

If you do not want to specify an alternate DDNAME for the SYSIN or SYSPRINT data sets, set the DDNAME-list length field to the correct length and set all of the 8-character fields to binary zeros.

If you want to specify an alternate DDNAME for the SYSIN data set, specify this parameter and enter the alternate DDNAME in the fifth 8-character field. Otherwise, enter all binary zeros in the field.

If you want to specify an alternate DDNAME for the SYSPRINT data set, specify this parameter and enter the alternate DDNAME in the sixth 8-character field. Otherwise, enter all binary zeros in the field.

If you do not want to specify subsequent parameters, you can omit them from the list.

PAGEPTR

The pointer to the page-number list and similar to HDINGADDR described in *z/OS DFSMSdss Utilities*. This list provides a way to specify the starting page number for system output on the SYSPRINT data set. The page-number list is a fixed-length 6-byte field made up of a halfword field followed by a 4-byte EBCDIC page count, which specifies the starting page number for DFSMSdss to use for the SYSPRINT data set. If a value is specified both here and in the OPTPTR list, this value is used, and the OPTPTR value is ignored.

The first field, called the *page-number-list length field*, is a halfword field specifying the length of the list (excluding the field itself) as a binary value. The length is usually 4.

If you do not want to specify a starting page number, set the page-number-list length field to binary zeros. If you want to specify a starting page number, you must specify this parameter and a 4-character page value. If the page number is specified (page-number-list length field is not binary zeros), DFSMSdss resets this field to the current page number upon completion of the present invocation. If the page number is not specified, this field is not changed by DFSMSdss.

If you do not want to specify subsequent parameters, you can omit them from the list.

UIMPTR

The pointer to the user interaction module (UIM) list. There is no comparable parameter described in *z/OS DFSMSdss Utilities*. This parameter provides a way to specify the name or address of a vector-implemented exit module that is to interact with DFSMSdss for the various I/O and data set operations. The UIM list is of variable length and consists of a halfword field followed by a 4-byte address or 8-character left-justified (padded on the right with blanks) string field that specifies the address of the UIM entry point or the load module name (located through the normal LINKLIB structure) of the UIM. If you do not want to specify a UIM, do not specify this parameter.

The first field, called the *UIM-list length field*, is a halfword field specifying the length of the list (excluding the field itself) as a binary value. If you do not want to specify an option, set the option-list length field to binary zeros. If the address of the UIM is being passed, it specifies the length as 4, or as 8 if the name of the UIM is being passed.

If you want to specify a UIM, you must specify the name or address of the UIM in the field following the UIM-list length field. See [“System programming information” on page 584](#) for more details on the use of this module.

UAPTR

The pointer to the user-area list. There is no comparable parameter in *z/OS DFSMSdss Utilities*. This parameter provides a way to specify the address of an area to be passed to the UIM at each DFSMSdss exit point. The user-area list is of fixed length, consisting of a halfword field called the *user-area-list length field*, and a single 4-byte address that locates the user area starting address.

If you use the user area, the length must be set to 4 and the address of the user area must be specified in a second field.

ASIDPTR

The pointer to the address-space-identifier list. There is no comparable parameter described in *z/OS DFSMSdss Utilities*. This optional parameter, which is applicable only when you use ADRXMAIA instead of ADRDSSU, lets you specify an identifier for the address space that ADRXMAIA uses for the DFSMSdss program. The address-space-identifier list is of fixed length, consisting of a halfword field and a single 8-byte character field. The default value, if not set by the application program, is "DFSMSDSS." This value is set to "DSSBATCH" when JCL is used to invoke ADRXMAIA, but can be specified through the `ASPACE=name` PARM field.

ADRXMAIA automatically creates the address space as needed. The identified address space can also be created by the operator START command by specifying an appropriately named PROCLIB member name.

PARAM

On the ATTACH and LINK macros, the keyword with which you specify the names of the pointers that are passed to DFSMSdss

VL

Indicates that the list is variable length. Both the ATTACH and LINK macros require specifying VL=1.

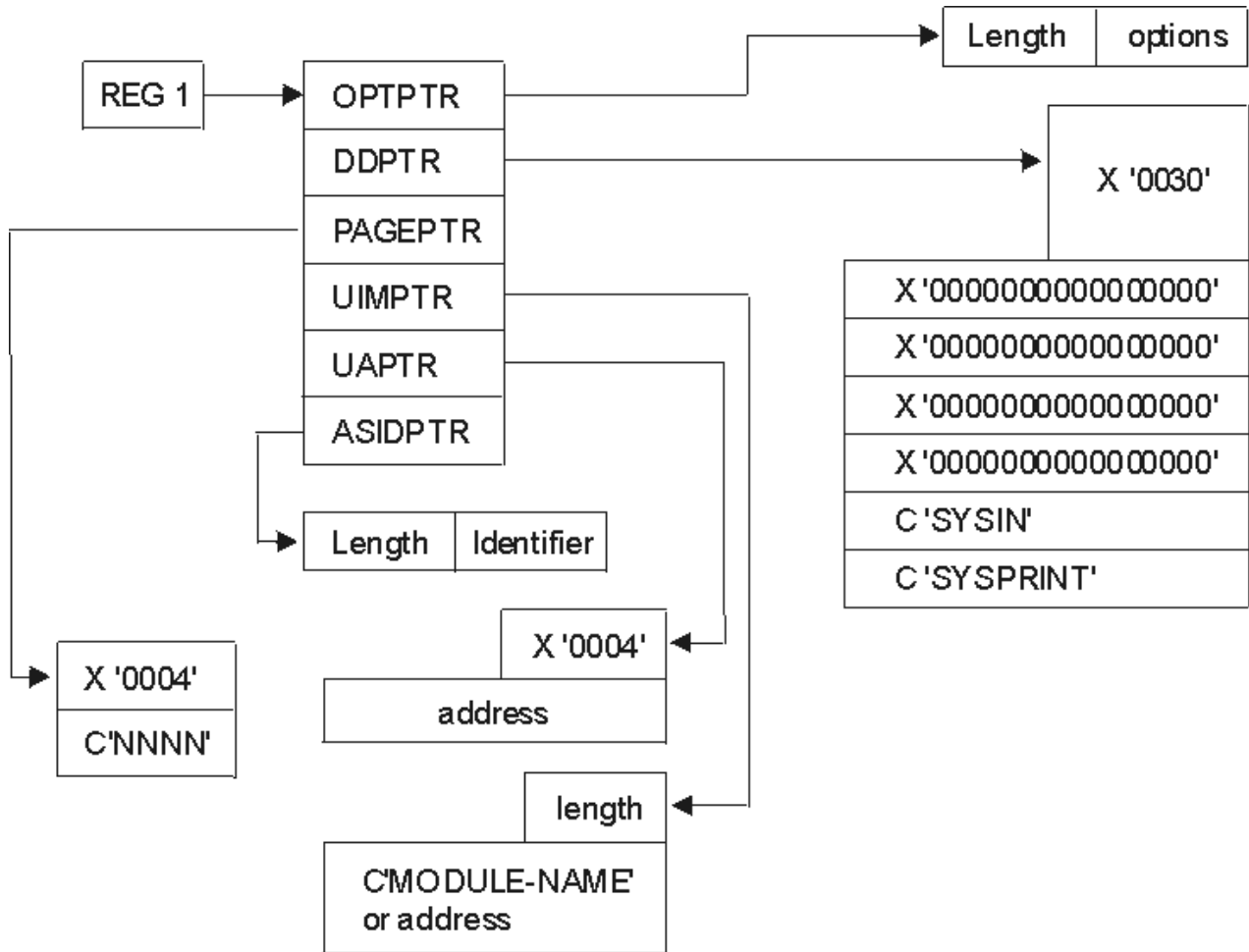


Figure 27. DFSMSdss Application Interface Structure

User interactions

For user interactions to take place, the application must invoke DFSMSdss and must provide a pointer to a user interaction module (UIM) list. DFSMSdss can be invoked by any of the following system macros:

```
ATTACH EP=ADDRSSU,PARAM=(OPTPTR,DDPTR,PAGEPTR,UIMPTR,UAPTR),VL=1
LINK EP=ADDRSSU,PARAM=(OPTPTR,DDPTR,PAGEPTR,UIMPTR,UAPTR),VL=1
CALL (15),(OPTPTR,DDPTR,PAGEPTR,UIMPTR,UAPTR),VL
```

Optionally, to use the DFSMSdss Cross Memory Application Interface, use one of the following system macros:

```
ATTACH EP=ADRXMAIA,PARAM=(OPTPTR,DDPTR,PAGEPTR,UIMPTR,UAPTR),VL=1
LINK EP=ADRXMAIA,PARAM=(OPTPTR,DDPTR,PAGEPTR,UIMPTR,UAPTR),VL=1
CALL (15),(OPTPTR,DDPTR,PAGEPTR,UIMPTR,UAPTR),VL
```

Optionally, to use the DFSMSdss Cross Memory Application Interface and specify an 8-character Address Space name, use one of the following system macros:

```
ATTACH EP=ADRXMAIA,PARAM=(OPTPTR,DDPTR,PAGEPTR,UIMPTR,UAPTR,ASNPTR),VL=1
LINK EP=ADRXMAIA,PARAM=(OPTPTR,DDPTR,PAGEPTR,UIMPTR,UAPTR,ASNPTR),VL=1
CALL (15),(OPTPTR,DDPTR,PAGEPTR,UIMPTR,UAPTR,ASNPTR),VL
```

As shown in [Figure 27 on page 579](#), a pointer to the UIM exit is passed in the parameter list (UIMPTR). User interactions involve:

- Record management
- Controlling processing of data sets
- Gathering statistics on DFSMSdss operations

When a UIM exit routine is specified, DFSMSdss processes normally, then at each point in the process (DFSMSdss exit points), the UIM exit routine is called conditionally to allow some types of user operations.

If a DFSMSdss subtask abnormally ends for any reason, it cannot make any further calls to the UIM, including the *function ending* call.

Note:

1. DFSMSdss runs as an authorized problem program.
2. Any program invoking DFSMSdss must also be authorized and in a nonsupervisor state.
3. ADRXMAIA is the name of the DFSMSdss Cross-Memory Application Interface module main entry point. Use ADRXMAIA instead of ADRDSSU when DFSMSdss functions are to be executed in a specific address space.
4. ADRXMAIA runs as an authorized program and supports both supervisor-state callers and problem-state callers.

Service considerations

After applying maintenance to any of the DFSMSdss load modules, you cannot update the version of DFSMSdss that is resident in your address space with LLA REFRESH. Instead, you must restart your application, or cause the ADRDSSU or ADRXMAIA load modules to be reloaded. Additionally, if your application uses DFSMSdss Cross Memory services, you must cause the DFSMSdss Cross Memory server address space to shut down in order for the updated ADRDSSU and other load modules to be reloaded.

The server address space will not shut down as long as the original address space that invoked the DFSMSdss Cross Memory services has not been stopped. Once the original address space has been stopped, you should shut down the DFSMSdss Cross Memory server. The server may not shut down if the SRVTIME parameter with a value of 00:00 was specified to the DFSMSdss Cross Memory client.

The name of the Cross Memory server can be specified with the ASPACE parameter on DFSMSdss Cross Memory services invocations. You should inform users of your application what name was specified. To identify DFSMSdss Cross Memory server address spaces, you can use the SDSF DA (Display Active) panel. Look for address spaces that are executing the ADRXMAIB load module. For information on using SDSF, see SDSF's online help (press PF1).

Related reading: For additional information about the various UIM exits, see [Chapter 24, “Examples of the application program with the user interaction module \(UIM\),” on page 627](#).

Cross-memory Application Interface overview

The DFSMSdss Cross Memory Application Interface is based on a client/server model. IBM products that make use of the DFSMSdss Cross Memory Application Interface include IMS Image Copy (IC2) and DFSMSShsm. DFSMSdss also provides a JCL interface for this support.

DFSMSdss Cross Memory support is invoked using the LINK, CALL, or ATTACH macros. The support might also be invoked with JCL or through system invocations. To use the JCL interface, modify the JCL that is normally used for DFSMSdss batch mode processing to execute program ADRXMAIA instead of program ADRDSSU, for example:

```
//S1 EXEC PGM=ADRXMAIA,PARM='TYPRUN=NORUN'  
           in place of  
//S1 EXEC PGM=ADRDSSU,PARM='TYPRUN=NORUN'
```


The DFSMSdss Cross Memory Application Interface support provides the capability for client applications to connect to and interact with a separate DFSMSdss server address space using a user-provided Interaction Module (UIM). DFSMSdss processing that is performed on behalf of a connected client application is controlled by one or more jobstep tasks that are executing in the server address space. The DFSMSdss server address space might be created by issuing a START command or might result from the ASCRE macro that is issued by the Cross Memory Application Interface support that is invoked by the client application.

After a DFSMSdss server address space begins execution, module ADRXMAIB is given control by the system, and directs server processing. The server processing that is performed on behalf of a client is referred to as a work thread, and it executes under a unique jobstep task. The jobstep task invokes the ADRDSSU program for each work thread request that the client application makes. A server can concurrently process multiple client work threads for multiple client-connected address spaces. Each work thread is a separate instance of an ADRDSSU jobstep task. Multiple server address spaces might exist and be concurrently processing on behalf of multiple client address spaces.

A client application invokes module ADRXMAIA to utilize the DFSMSdss Cross Memory Application Interface support. When a client application invokes ADRXMAIA, DFSMSdss attempts a connection to the client-identified server address space. If the client-identified DFSMSdss server address space does not exist, the Cross Memory Application Interface invokes the ASCRE macro to create the server. If the client does not provide a server address space identifier, "DFSMSDSS" is used as the default server address space identifier. The JCL interface uses "DSSBATCH" as the default identifier unless the ASPACE parameter supplies it.

You can also create a DFSMSdss server address space using a START command that specifies an appropriately named member in SYS1.PROCLIB. The procedure must invoke module ADRXMAIB. The proclib member name should match the server identifier that is used by client applications or match batch jobs that use the JCL interface that is connecting to the server that is created.

To start the DFSMSdss server address space, follow these steps:

1. Create the following started task procedure and add it to SYS1.PROCLIB(DFSMSDSS):

```
//*****
//*   THIS PROCEDURE WILL CREATE AN APPROPRIATE DFSMSDSS CROSS      *
//*   MEMORY SERVER TO BE USED WITH APPLICATIONS THAT INVOKE CROSS  *
//*   MEMORY REQUESTING THE DEFAULT DFSMSDSS SERVER NAME.          *
//*                                                                    *
//*   TO USE, ENTER THE FOLLOWING AT A CONSOLE:                      *
//* START DFSMSDSS,PROG=ADRXMAIB                                     *
//*                                                                    *
//*   WHEN THE DFSMSDSS CROSS MEMORY SERVER IS NO LONGER REQUIRED    *
//*   ISSUE THE FOLLOWING MODIFY COMMAND:                             *
//* F DFSMSDSS,STOP                                                *
//*                                                                    *
//*****
//DFSMSDSS PROC  PROG=IEFBR14
//IEFPROC  EXEC  PGM=&PROG,REGION=0M,TIME=1440,DYNAMNBR=1635
```

2. Start the DFSMSdss server address space by using either of the following methods:

- Issue the following command:

```
S DFSMSDSS,PROG=ADRXMAIB
```

- Add the procedure as a started task in your IPL procedure.

3. When the DFSMSdss server address space is started, run your DFSMSdss batch job with PGM=ADRXMAIA,PARM='ASPACE=DFSMSDSS'

4. When your jobs end, and if you want to stop the server address space, issue the following command:

```
F DFSMSDSS,STOP
```

A DFSMSdss server address space that was created using the ASCRE macro goes into termination mode if there is no more work to do after all connected client applications terminate. There is a delay period

prior to a server termination. Any connection request during this delay period causes the server to revert to processing mode to process the work threads that are associated with the connecting clients.

DFSMSDss server address spaces remain active as long as a client remains active unless the operator MODIFY command informs the server that it should stop processing after all current work threads are completed, for example F DFSMSDSS.DSSBATCH,STOP or F DSSBATCH,STOP.

A DFSMSDss server that is started with a START command does not enter termination mode when all work is completed. You must use the MODIFY command to stop it.

Using the cross memory application interface to control DFSMSDss

You can control DFSMSDss through PARM Information in the EXEC statement by using the Cross Memory Application Interface. The EXEC statement for DFSMSDss, when invoking DFSMSDss with the Cross Memory Application Interface, can contain PARM information that is used by the client and server, as well as the ADRDSSU program itself. The same values that can be specified in the EXEC PARM field in the JCL for ADRDSSU can also be specified in the EXEC PARM field for ADRXMAIA.

SRVRTIME=({minutes}:{seconds})

The SRVRTIME parameter specifies the amount of time the server address space should wait to shut down after the last piece of work has finished. If the Cross Memory Application Interface is invoked again, specifying this particular server before the specified time has passed, this server will take the piece of work. When the time expires and no more pieces of work have been submitted, the server will shut down, and a subsequent invocation of the Cross Memory Application Interface specifying this server name will cause a new server to be created. The first invocation of the Cross Memory Application Interface for a particular server determines the length of time while that particular server is running. Subsequent invocations to the same server while it is running will not change the time even if the SRVRTIME parm is specified on those later invocations.

minutes

Specifies the maximum number of minutes the server will wait after the last piece of work is finished before shutting down. The minutes must be a number from 0 through 357912 (248.55 days).

seconds

Specifies the maximum number of seconds the server will wait after the last piece of work is finished before shutting down. The seconds must be a number from 0 through 59.

The following examples demonstrate the affect on the system when you designate a SRVRTIME value:

- SRVRTIME=(1:30) - The server will wait for 1 minute 30 seconds before shutting down.
- SRVRTIME=(24:00) - The server will wait for 24 minutes before shutting down.
- SRVRTIME=(0:25) or SRVRTIME=(:25) - The server will wait for 25 seconds before shutting down.
- SRVRTIME=(0:00) - The server will shut down immediately after the last piece of work is finished.

If the SRVRTIME parameter is not specified, the time before the server shuts down is determined as follows:

- If the Cross Memory Application Interface is invoked from JCL, but the ASPACE parameter is not specified, the server will shut down after 4 minutes.
- If the Cross Memory application Interface is invoked with JCL and the ASPACE parameter is specified, the following will be true:
 - If ASPACE=DSSBATCH is specified, the server will shut down after 4 minutes.
 - If ASPACE=DFSMSDSS is specified, the server will shut down after 8 minutes.
 - If any other name is specified, the server will wait 1 minute.
- If the Cross Memory Application Interface is invoked using the LINK, CALL, or ATTACH macros, but an ASPACE name wasn't provided in the ASIDPTR field, the server will wait 8 minutes.
- If the Cross Memory Application Interface is invoked using the LINK, CALL, or ATTACH macros and an ASPACE name was provided in the ASIDPTR field, the server will wait 1 minute.

ASPACE=id

Where *id* is determined by the installation to identify which server to use for processing DFSMSdss SYSIN command streams. ASPACE=AFFINITY is a special use. It causes ADRDSSU to be used within the client address space instead of running in a separate address space.

ASPACE is only available when using JCL to invoke DFSMSdss with the cross memory application interface.

The following example demonstrates how you might designate the use of the ASPACE parameter:

```
//S1 EXEC PGM=ADRXMAIA,PARM='ASPACE=BACKUP'
```

SNAPX=*[nn](nn,nn[,nn])

The SNAPX parameter can be used to debug your user interaction module. The SNAPX parameter can be specified on JCL and system invocations of the Cross Memory Application Interface (API). Specifying the SNAPX parameter requests that ADRXMAIA write the contents of the EIDB and EIRECPTR to SYSPRINT whenever the specified exit is called. When an application writes its own messages to SYSPRINT, ADRXMAIA calls exit 2 of the user interaction module for that application.

When specifying the SNAPX parameter, you can specify:

- One exit to snap. For example, you can specify SNAPX=12 to see the contents of exit 12.
- More than one exit to snap. For example, you can specify SNAPX=(1,6,2) to see the contents of exits 1,2, and 6.
- All command processing exits. For example, you can specify SNAPX=*.

The following example demonstrates how you might designate the use of the SNAPX parameter:

```
//S1 EXEC PGM=ADRXMAIA,PARM='SNAPX=(21,22,23)'
```

Cross-memory application return codes

The cross-memory application interface uses the user interaction module (UIM) exit points to communicate with the application and exit option 14 returns the DFSMSdss return code. Those return codes are documented in *z/OS MVS System Messages, Vol 1 (ABA-AOM)*. However, when the cross memory client cannot communicate with the server, or is unable to schedule a task, a set of unique codes is returned in register 15.

Note: When an ABEND occurs, the cross-memory application interface may return an ABEND code in register 15 instead of a return code.

The following table lists the return codes and the action that needs to be taken.

Table 41. Cross-memory application return codes

Hexadecimal return code	Meaning	Action
800	Unable to acquire storage for task.	Provide adequate storage by increasing either the REGION size, the SIZE parameter, or both. See “Storage requirements” on page 19 for more information about storage estimates.
802	Unable to load application UIM.	Provide the name of a valid load module and make sure it is able to be located in the LNKLIST concatenation.
804	Unable to acquire storage for task.	Provide adequate storage by increasing either the REGION size, the SIZE parameter, or both. See “Storage requirements” on page 19 for more information about storage estimates.

Table 41. Cross-memory application return codes (continued)

Hexadecimal return code	Meaning	Action
806	Unable to start the server address space.	Contact IBM support.
990	The server is busy.	Contact IBM support.
997	The server is not responding. See the operator log for a corresponding ADR111I message.	Contact IBM support.

System programming information

Some bit definitions in the installation options control block (ADRUFO) permit DFSMSdss to communicate with the installation options exit routine. When the installation options exit does not want a function to be scheduled, it returns a code of 8. For the UIM to do this requires a bit definition of UFSTOP to be set in the ADRUFO. If a SYSIN or SYSPRINT data set is not to be allocated or all SYSIN/SYSPRINT is to be handled in storage, UFSYSIN and UFSYSPR specify this to DFSMSdss.

If the application allows DFSMSdss to handle SYSPRINT by not setting UFSYSPR, then the application cannot write to SYSPRINT directly. The application can only insert SYSPRINT records at UIM exit points 2 or 10. If the application chooses to handle SYSPRINT by setting UFSYSPR, then the application is responsible for printing DFSMSdss messages from UIM exit points 2 or 10.

Three additional bits determine allowances within the UIM interaction: UFAIINV, UFUIMAL and UFUIMCH. Two additional bits indicate the existence or absence of an input restore/copydump data set or an output dump/copydump data set: UFNOIN and UFNOOUT respectively. UFNOIN and UFNOOUT are not supported for backups to an object storage cloud, or restores from it. Refer to *z/OS DFSMS Installation Exits* for more information about the installation options exit routine.

When DFSMSdss has been invoked by using the Application Interface and the UIM has been specified and is to be called, the following information is passed to the UIM on every exit call:

- Register 1, which points to the interface parameter list pointer.
- The interface parameter list pointer, which points to the DFSMSdss exit identification block. See “ADREID0 data area” on page 609 for a detailed description of this block. This list consists of:
 - A halfword field specifying the length of the remainder of the list.
 - The remainder of the list that is mapped by the macro ADREID0. See “Exit identification block” on page 585 for the information contained in this block.

Upon return from the UIM, the user return code field is examined to determine the disposition of the current I/O record or data set (within the limits allowed to the exit).

Application interface blocks

The parameter structure described in [Figure 27 on page 579](#) can be viewed in block form as shown in [Figure 28 on page 584](#).

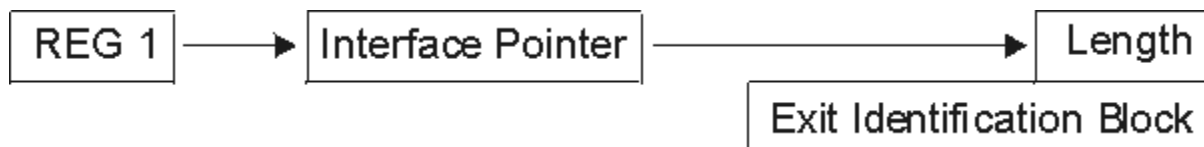


Figure 28. DFSMSdss Exit Interface Structure

Exit identification block

The exit identification block is passed to the user interaction module every time DFSMSdss gives control to it. Each field is described below, but see [“ADREID0 data area” on page 609](#) for the formal declarations.

Control Block Eye-Catcher

A 4-character string field that is supplied by DFSMSdss. It contains the character string *EIDB* and can aid in locating the control block when you are viewing a storage dump during DEBUG processing.

TASK-ID

A fullword binary field that is supplied by DFSMSdss. The number contained in this field is assigned by DFSMSdss to each function command statement submitted in SYSIN, whether obtained from the data set or from the UIM. This number is binary zero when DFSMSdss is calling the UIM for a function that is not related to a user command statement. Each command is numbered in sequence. When a task is scheduled to process this command, all messages associated with this task and all calls to the user interaction module for this task are accompanied by this unique number. In this way, the UIM can identify which task is being processed and what function is associated with that task.

User Exit Allowance

A fullword binary field supplied by DFSMSdss. The 32 bits defined in this field can be used as flags to determine what actions the UIM can perform with respect to the record presented. The following actions are conditionally allowed:

- View and conditionally override the installation options.
- Insert data prior to current record.
- Replace the current data record with an exit-record supplied.
- Delete the current data record.
- Modify the current data record.
- Disconnect the exit from further interaction.
- Recognize when a disallowed option has been attempted.
- End processing of the data set.
- End processing of the task.

DFSMSdss Processing Option

A halfword binary field that is supplied by DFSMSdss. The number contained in this field can be used by the user interaction module (UIM) to vector branch to the appropriate processing routine for the record or data set being presented to the exit.

At appropriate locations in the DFSMSdss processing modules, the user interaction module is considered for receiving control. These locations are referred to as DFSMSdss exit points.

User Return Code

A halfword binary field that is supplied by the UIM. This return code identifies the action expected by the exit on the record or data set being presented to the exit. DFSMSdss examines this field when control returns from the UIM. Not all the following return codes are allowed at any given exit call. If a disallowed code is returned from the UIM, DFSMSdss passes the EIDB back to the UIM with the EIXERR flag set. This allows the UIM one chance to correct the option. If a disallowed code is returned again, it is ignored by DFSMSdss and the record is processed as though the exit had returned the code zero (0). If a valid code is returned, DFSMSdss processing is the same as if the valid code had been passed back on the initial UIM invocation. This sequence is followed for any subsequent incorrect return codes. The return codes that are allowed at any given exit call are specified in the EIXALLOW field.

- The meaning of each return code for those exits presenting records to the UIM (Eioptions 01 to 26) is:

0

The record is processed as would normally have happened if there had not been a UIM. The original record is not changed in any way by the exit.

4

The record was replaced by the exit. The new record must be placed in the area pointed to by the original record pointer field and its length stored in the original record length field.

8

The record is to be inserted. The address of the new record must be stored in the original record pointer field or the new record must be stored in the area pointed to by the original record pointer field and its length must be stored in the original record length field. When this exit is next called, the original record is presented again.

12

The record is to be deleted. The record presented to the UIM is ignored in the processing, thus deleting it.

16

The record was modified by the exit. This return code is the only one allowing the original record to be altered and then to be processed by DFSMSdss. Any changes made to the record must be logically correct because DFSMSdss cannot assure the validity of these changes.

Note:

1. You cannot change the record length when you modify the record (as contrasted with reason code 4).
2. If the record being processed is the installation options record (ADRUF0) and any values have been changed, return code 16 must be returned or the changes are ignored.

20

The record is to be processed as though a return code zero (0) had been given, but this particular DFSMSdss exit point is no longer called from the current functional task, although it might be called from others. You must be cautious in using this code because multiple record types use the same DFSMSdss exit point. It would be better for the UIM to inspect each record type and only give a return code zero (0) when a record is not of interest to the exit.

24

The record is to be processed but at all future visits of DFSMSdss at this exit point, only user statistical records are to be presented to the UIM.

28

The notification-of-response return code. The WTOR has been handled by the UIM and the UIM has supplied the proper response from the WTOR in the area pointed to by the original record pointer. The interface allows you to handle all WTOR processing in the UIM exit routine and to supply the required response to DFSMSdss in lieu of DFSMSdss's issuing the WTOR itself. Note that the UIM must set the original record length to the proper value.

32

DFSMSdss ends the current functional task and issues message ADR356E.

- The meaning of each return code for those exits controlling data set processing (Eioptions 21, 22, 23, and 26) is:

0

The data set is processed as would normally have happened if there had not been a UIM. Data set processing is not altered in any way.

16

This return code is valid from exit 22 only. It indicates to DFSMSdss to examine the bypass processing flags and to modify processing of the data set according to which flags are on. These bypass flags are only examined by DFSMSdss if a return code of 16 is returned. See [“Bypass verification exit \(Eioption 22\)”](#) on page 596 for more details about these flags.

20

The record is to be processed as though a return code zero (0) had been given, but this particular DFSMSdss exit point is no longer called from the current functional task, although it might be called from others. You must be cautious in using this code because multiple record

types use the same DFSMSdss exit point. It would be better for the UIM to inspect each record type and only give a return code zero (0) when a record is not of interest to the exit.

32

DFSMSdss ends the current functional task and issues message ADR356E.

36

DFSMSdss ends processing of the data set named in the parameter passed to the exit. Processing continues with the next data set (if any). If Exit 23 sets this return code after the data set has already been processed, then DFSMSdss does not undo the processing, but deletes the data set from the successfully-processed message list (if applicable) and includes it in the unsuccessfully-processed message list even though the data set may have been successfully processed up to that point

Record Area Length

A fullword binary field that is supplied by DFSMSdss. The number contained in this field represents the total length of the area in which the original record is stored. The number can be used by the UIM to verify that replacement and inserted records fit in the provided buffer area pointed to by the original record pointer field.

Original Record Length

A fullword binary field that is supplied by DFSMSdss but can be changed by the UIM. The number contained in this field represents the total length of the record pointed to by the original record pointer. The length includes just the length of the record pointed to by the original record pointer field and does not include the length of this field itself. The value of this field is zero when the UIM is called to supply in-storage SYSIN data.

Original Record Pointer

A fullword address field that is supplied by DFSMSdss but can be changed by the UIM. This field contains the address of the record being passed to the UIM on this call. This address normally is not changed by the exit unless an insertion is being used. Refer to the proper return code descriptions for the effect on this field. The value of this field is zero when the UIM is called to supply in-storage SYSIN data. The location of the original record, above or below the 16-megabyte (MB) virtual storage line, is controlled by using the installation options exit. If this record is to be above 16MB, the UIM has to run in 31-bit addressing mode in order to address the record.

User Area Pointer

A fullword address field that is supplied by the application program and is maintained by DFSMSdss. This field contains the address of the user data/work area that was supplied by the application program as a communications area for UIM internal process controls. DFSMSdss saves this pointer and supplies it in the EIUSEPTR field of the exit identification block on each call to the UIM. If any UIM exit changes the user area pointer, DFSMSdss presents the updated pointer on subsequent calls to the UIM.

Each DFSMSdss functional task keeps its own copy of the user area pointer. If the pointer is changed by the UIM for one task, it is not changed for any other task. At the beginning of each task, the user area pointer is the one passed to DFSMSdss by the application program.

DDNAME/VOLID Pointer

A fullword address field supplied by DFSMSdss. It contains the address of an area containing the DDNAME of the output dump data set, left-justified in an 8-byte area, followed by a 6-byte area containing the volume serial number of the volume containing the dump data set, also left-justified. This pointer is valid only for the full-volume dump exit, EIOP06.

Application programming interface restrictions

If you write a program that invokes ADRDSSU and needs to use some features of the z/OS Unix System Services, your program makes a call to z/OS UNIX through API. To connect to the kernel for z/OS UNIX, your program has to make an address space known to it. This process is called dubbing. Once dubbed, the address space is considered a thread in the z/OS Unix System Services space, and has the same security authorization as your program has. Therefore, your program and the ADRDSSU program are dubbed with your security authorization.

To avoid authorization problems, perform undub before DFSMSdss invocation. If you cannot undub before DFSMSdss invocation, undub after the ADRDSSU program has finished executing.

Related reading: For more information about using z/OS UNIX System Services, see:

- [z/OS UNIX System Services Programming: Assembler Callable Services Reference](#).
- [z/OS UNIX System Services Planning](#).

Cross-memory application interface restrictions

Use SDUMP and MSGCNT in place of ABEND and AMSCNT when the Cross-Memory Application Interface is used. Although ABEND and AMSCNT are supported by DFSMSdss, no dump is printed because DD allocations (SYSABEND, SYSUDUMP) in the application address space are unavailable to the address space producing the abend.

Serialization may operate differently, as application address space DD allocations are unavailable to the address space executing the DFSMSdss functions. Evaluate your serialization requirements when considering the use of the Cross-Memory Application Interface.

When multiple ADRDSSU tasks are executing in a DFSMSdss server address space, and identical DDNAMEs are passed in the SYSIN stream, allocation errors (such as MSGIKJ56246I – "file in use") can result. To avoid this possibility, use 8-character DDNAMEs or leave enough room after the DDNAME to allow ddname replacement. The Cross Memory Application Interface replaces common DDNAMEs passed in the SYSIN stream with unique 8-character, system-generated DDNAMEs whenever possible.

If you use the DFSMSdss Cross Memory Application Interface for logical COPY or logical DUMP and RESTORE of HFS and zFS type data sets, you must define a DFSMSdss user ID for the DFSMSdss server address spaces to be able to access the HFS and zFS data sets. The DFSMSdss user ID must be set up for the z/OS UNIX System Services (z/OS UNIX) access as follows:

- The default group for the DFSMSdss user ID must have a z/OS UNIX segment defined and a group ID associated with it.
- The home directory should be the root file system.
- The DFSMSdss user ID must be defined as a superuser (UID 0). Use the following RACF commands to make this assignment:

```
ADDGROUP OMVSGRP OMVS(GID(1))  
ADDUSER DFSMSDSS DFLTGRP(OMVSGRP) NOPASSWORD OMVS(UID(0) HOME('/'))
```

You might want to set up a RACF (or equivalent) permission for the processing performed by the DFSMSdss cross-memory server address space. If you are using the ASpace parameter, or ASIDPTR feature to request a specific address space name for the DFSMSdss Cross Memory server, enter an RDEF command to add that address space name to the Started Class table of your security product and associate that started task name (address space name) with a user id that has sufficient authority to perform the DFSMSdss functions.

The following example shows how to define started tasks that begin with ARC to RACF:

```
RDEF STARTED ARC*.** UACC(NONE) OWNER(ADMIN) AUDIT(ALL(READ)) -  
STDATA(USER(DFSMSDSS) GROUP(STC) TRACE(YES))
```

where DFSMSDSS is a user id defined to RACF that has sufficient authority to perform the DFSMSdss functions and STC is a group that the user id is connected to. The above command sets a rule to match any new started tasks such as ARC1DUMP.ARC1DUMP with the DFSMSDSS user id. The wildcard usage allows other started tasks such as ARC1MIGR.ARC1MIGR to be associated with the DFSMSDSS id as well.

Specifying multiple commands (DUMP, RESTORE, or COPYDUMP) that process DFSMSdss dump data sets on the same tape volume is not supported when invoking DFSMSdss using the Cross-Memory Application Interface. This restriction applies whether or not the PARALLEL command is specified. To process multiple DFSMSdss dump data sets on the same tape volume using the Cross-Memory Application Interface, use multiple invocations of DFSMSdss with one command per invocation and ensure that the operations are executed serially.

The Cross-Memory Application Interface does not support overriding host-based encryption in favor of encryption provided by encrypting tape devices.

The DFSMSdss server's address space dispatching priority must be the same or higher than the priority of all clients' address spaces. The server's address space must be able to dispatch during the clients' termination process; otherwise, ABENDs may occur.

Related reading: For additional information about z/OS UNIX, see *z/OS UNIX System Services Planning*.

User interaction module exit option descriptions

This section contains intended programming interface information.

Following are the descriptions for all exit points available with DFSMSdss.

Function startup (Eioption 00)

This exit point is called during DFSMSdss initialization and again during function initialization (such as a dump or restore operation). The EIRECPtr points to the EIREC00 data area. The EIREC00 data area contains a field, EI00SBPL, which the application may use to return a subpool number that is to be shared between all DFSMSdss tasks. EI00SBPL is only valid when EITSKID is zero.

The application can set EI00NOLK to request that DFSMSdss bypass ADRLOCK ENQUEUE or DEQUEUE processing during full-volume and tracks RESTORE operations. The application must be able to resolve any deadlock conditions that result from the request to bypass ADRLOCK ENQUEUE or DEQUEUE processing.

The application can also set EI00SENQ to request that VTOC serialization be held only for the duration of VTOC access. This flag is only valid for full volume dump operations and when EITSKID is nonzero. This flag provides a function similar to that provided by the Enqueue Installation Exit Routine (see *z/OS DFSMS Installation Exits*). The VTOC serialization is released after VTOC access is complete if either the EI00SENQ flag is set or the Enqueue Installation Exit Routine requests it.

The application can also set EI00NENQ to request that DFSMSdss not reserve on VTOC of the source volume. This flag is only valid for physical COPY and DUMP operations during function startup (for example, when EITSKID is nonzero).

The application can also set EI00BSEC flag to request that RACF verification and all other data set or z/OS UNIX file security checking is bypassed. This flag is only valid for full/tracks COPY, full/tracks DUMP, and full/tracks RESTORE operations during function startup (that is, when EITSKID is nonzero).

DFSMSdss provides the source VOLSER to the UIM. This field is only valid for full/tracks COPY, full/tracks DUMP, and full/tracks RESTORE during function startup (that is, when EITSKID is nonzero). For RESTORE, this field contains the volume serial number of the input volume where the dump data set resides.

The application can also set:

- EI00CIRBA_DA, when processing a VSAM or non-VSAM data set
- EI00CIRBA_IX, if the VSAM data set is indexed during physical data set copy so that DFSMSdss can read the VVR RBA in the VVDS. This is allowed only if a single data set that is fully qualified is specified in the include criteria, and EIMVOLRECOV is enabled.

The valid return codes for function startup are:

0

Continue normal processing

16

Record modified

20

Inhibit all UIM calls—Not valid when EITSKID is zero and either UFSYSIN or UFSYSPR was set in the installation options exit, or EITSKID is nonzero and either UFNOIN or UFNOOUT was set in the installation options exit.

32

End processing. This return code is only valid for a full dump and a full restore operation.

Reading SYSIN record (Eioption 01)

This exit point is called after DFSMSdss reads a SYSIN record. You can replace, insert, delete, or modify a SYSIN record at this exit point. The EIRECPTR points to the SYSIN record. The valid return codes are:

0

Continue normal processing

4

Record replaced

8

Insert record

12

Delete record

16

Record modified

20

Disconnect exit

Printing SYSPRINT record (Eioption 02)

This exit point is called when DFSMSdss is ready to print a SYSPRINT record. You can replace, insert, delete, or modify a SYSPRINT record at this exit point. The EIRECPTR points to the SYSPRINT record. Task-related SYSPRINT records are presented in task order unless the UIM requested that there be no SYSPRINT (see [“System programming information” on page 584](#)).

The valid return codes for printing SYSPRINT records are:

0

Continue normal processing

4

Record replaced

8

Insert record

12

Delete record

16

Record modified

20

Disconnect exit

Note:

1. You can only modify or delete a page header record during the print operation.
2. EIXNTERR is set ON when the following conditions exist:
 - You have specified the TOL(IOERR) keyword.
 - DFSMSdss issues e-type messages with the exception of messages ADR324E, ADR347E, and ADR348E.
 - DFSMSdss issues t-type messages; the TOL(IOERR) keyword does not have to be specified.
3. EIXNTERR is set OFF when the following conditions exist:
 - An I-type or W-type message is issued by DFSMSdss.

- DFSMSDss issues the ADR324E, ADR347E, and ADR348E (all or some combination) messages, and the TOL(IOERR) keyword is specified.
4. EIXNTERR is unchanged when DFSMSDss issues a nonprefixed message.

Reading physical tape record (Eioption 03)

This exit point is called when DFSMSDss has read a record from a data set that has been dumped (on tape or DASD) by using the DUMP command. The EIRECPTR points to the tape record. The valid return codes are:

- 0** Continue normal processing
- 8** Insert record. This return code is only valid when UFNOIN is set in the installation options exit
- 20** Disconnect exit. This return code is only valid when UFNOIN is not set in the user installation options exit
- 32** End function

Reading logical tape record (Eioption 04)

This exit point is called when DFSMSDss has read a record from a data set that was created with a DUMP command and dumped on tape or DASD. The EIRECPTR points to the tape record. The valid return codes are:

- 0** Continue normal processing
- 8** Insert record
- 20** Disconnect exit
- 24** Select user statistics records
- 32** End function.

Note: If code 24 is returned, DFSMSDss only calls this exit point when DFSMSDss processes user statistics records.

Writing logical tape record (Eioption 05)

This exit point is called when DFSMSDss writes a logical record to a dump data set, on tape, or on DASD, while performing a dump operation. If you insert a record at this exit point, DFSMSDss marks it as a statistics record, not a data record. The EIRECPTR points to the tape record. The valid return codes are:

- 0** Continue normal processing
- 8** Insert record
- 20** Disconnect exit
- 32** End function.

Writing physical tape record (Eioption 06)

This exit point is called when DFSMSdss writes a physical record to a dump data set, on tape, or on DASD. The EIRECPTR points to the tape record and the EIDDID points to EIDDINFO. The valid return codes are:

- 0**
Continue normal processing
- 12**
Delete record. This return code is only valid when UFNOOUT is set in the installation options exit
- 20**
Disconnect exit. This return code is only valid when UFNOOUT is not set in the user installation options exit
- 32**
End function.

Reading disk track (Eioption 07)

This exit point is called when DFSMSdss has read a track from DASD. The EIRECPTR points to the track buffer. The first 64 bytes (X'40') of the track buffer are IBM internal information that is followed by the ADRTAPB area, the DTTTRK area, and the track image. The valid return codes are:

- 0**
Continue normal processing
- 20**
Disconnect exit
- 32**
End function. This return code is only valid for a full dump

Related reading: For additional information about ADRTAPB and DTTTRK, see [Chapter 13, “Format of the DFSMSdss dump data set,” on page 179.](#)

Writing disk track (Eioption 08)

This exit point is called when DFSMSdss is ready to write a track to DASD. The EIRECPTR points to the track buffer. The valid return codes are:

- 0**
Continue normal processing
- 20**
Disconnect exit
- 32**
End function. This return code is only valid for a full restore operation

Reading utility SYSPRINT (Eioption 09)

This exit point is called when DFSMSdss is reading output from an attached utility. You can replace, insert, delete, or modify a utility SYSPRINT record at this exit point. The EIRECPTR points to the Utility SYSPRINT record. The valid return codes are:

- 0**
Continue normal processing
- 4**
Record replaced
- 8**
Insert record
- 12**
Delete record

- 16**
Record modified
- 20**
Disconnect exit

Writing SYSPRINT record (Eioption 10)

This exit point is called when DFSMSdss is ready to write a SYSPRINT record. You can replace, insert, delete, or modify the SYSPRINT record at this exit point. The EIRECPTR points to the SYSPRINT record. Task-related SYSPRINT records are not presented in task order. The valid return codes are:

- 0**
Continue normal processing
- 4**
Record replaced
- 8**
Insert record
- 12**
Delete record
- 16**
Record modified
- 20**
Disconnect exit

Writing WTO message (Eioption 11)

This exit point is called when DFSMSdss is ready to write a WTO message. You can replace, insert, delete, or modify the WTO message at this exit point. The EIRECPTR points to the WTO message. The valid return codes are:

- 0**
Continue normal processing
- 4**
Record replaced
- 8**
Insert record
- 12**
Delete record
- 16**
Record modified
- 20**
Disconnect exit

Writing WTOR message (Eioption 12)

This exit point is called when DFSMSdss is ready to write a WTOR message (that is, ADR369D, ADR345D, and ADR371D). You can insert or modify the WTOR message at this exit point. You can also return the response to DFSMSdss with return code 28. The EIRECPTR points to the WTOR message. The valid return codes are:

- 0**
Continue normal processing
- 8**
Insert record

- 16**
Record modified
- 20**
Disconnect exit
- 28**
WTOR response

Presenting ADRUFO record (Eioption 13)

This exit point is called when DFSMSdss is ready to set up the installation options specified in the ADRUFO control block. You can modify the control block to override any options that have been specified thus far. You must use return code 16 for DFSMSdss to recognize the new options. The EIRECPTR points to ADRUFO. The valid return codes are:

- 0**
Continue normal processing
- 16**
Record modified
- 32**
End function. This return code is only valid for a full dump and a full restore operation.

Function ending (Eioption 14)

This exit point is called when DFSMSdss is ready to end the task. The EIRECPTR points to the last message with highest nonzero return code. The record contains the DFSMSdss return code and the DFSMSdss message (message number and message type). If any data was compressed, the record also contains a compression savings in terms of a percentage between 0 and 99. The valid return code is:

- 0**
Continue normal processing

Presenting WTOR response (Eioption 15)

This exit point is called when the operator has responded to the WTOR (either a DFSMSdss WTOR or a user inserted WTOR, see exit point 12). You can only examine the response at this exit point. If you would like to change the response, you must use exit point 12, change the response, and use a return code of 28 to DFSMSdss. The EIRECPTR points to the response. The valid return code is:

- 0**
Continue normal processing

OPEN/EOV tape volume security and verification exit (Eioption 16)

This exit point is called when DFSMSdss is ready to open a tape. You can bypass password and expiration date checking for tape volumes or reject the volume and request a scratch tape with this exit by placing a return code in the first word of the record pointed to by EIRECPTR. DFSMSdss passes this return code to OPEN/EOV. The EIRECPTR points to the parameter list described by the IECOEVSSE macro.

The valid return codes are:

- 0**
Continue normal processing
- 16**
Record modified
- 20**
Disconnect exit
- 32**
End function. This return code is only valid for a full dump and a full restore operation

Related reading: For additional information about OPEN/EOV, see [z/OS DFSMS Using Data Sets](#).

OPEN/EOV nonspecific tape volume mount (Eioption 17)

This exit point is called when a nonspecific tape is passed to DFSMSdss. The EIRECPTR points to the DCB exit parameter list. You can specify a specific volume serial with this exit by placing the volume serial in the first 6 bytes of the record pointed to by EIRECPTR. DFSMSdss passes this volume serial and return a code of 4 to OPEN/EOV informing it to use the specified volume serial. The EIRECPTR points to the parameter list described by the IEEOENTE macro.

Note: DEFER must be specified in the JCL for this exit to be called.

The valid return codes are:

- 0** Continue normal processing
- 16** Record modified
- 20** Disconnect exit
- 32** End function. This return code is only valid for a full dump and a full restore operation

Insert logical VSAM record during restore (Eioption 18)

This exit point is called during a logical restore operation of a VSAM KSDS using record-level I/O. You can replace or modify a record at this exit point. The EIRECPTR points to the logical record that DFSMSdss is preparing to write to the data set. The valid return codes are:

- 0** Continue normal processing
- 4** Record replaced
- 16** Record modified
- 20** Disconnect exit

Related reading: For additional information about Eioption 18, see [z/OS DFSMS Using Data Sets](#).

Output tape I/O error (Eioption 19)

This exit point is called during a dump when a tape has a permanent I/O error. The EIRECPTR points to EIDINFO. The valid return codes are:

- 0** Continue normal processing
- 20** Disconnect exit
- 32** End function

Volume notification (Eioption 20)

During physical data set restore operations, this exit is called to provide information about the data set being allocated. This information includes:

- Cluster/data set name (for RENAME and RENAMEUNCONDITIONAL keywords, the new data set name is presented)

- An indication of whether the data set is VSAM or non-VSAM
- Number of data and index components (VSAM only)
- For each component or data set:
 - The number of volumes it resides on
 - The volume serial number of each volume
 - An RBA token for each volume.

EIRECPTTR points to EIREC20. EI20DA@ and EI20IX@ point to EI20DSI.

Note: SMS-managed multivolume non-VSAM data sets have an RBA token only for the first volume. Non-SMS-managed data sets never have an RBA token.

The valid return codes are:

0

Continue normal processing

20

Disconnect exit

32

End function. This return code is only valid for full or physical data set dump and full or physical data set restore operations

Data set verification (Eioption 21)

This exit lets the UIM end data set copy, physical data set copy, dump, and restore processing for individual data sets. This exit is given control through the UIM at the start of data set copy, dump, and restore processing for each data set. DFSMSDss provides the UIM with the data set name through the EIREC21 structure within the exit identification block, ADREID0. EIREC21 is shown in [“ADREID0 data area” on page 609](#). The name provided is the original data set name prior to rename processing, if any, for copy and restore functions. The UIM returns to DFSMSDss with a return code in the exit identification block. The valid return codes are:

0

Continue normal processing

20

Disconnect exit

32

End function

36

End data set

DFSMSDss continues processing with the next data set, if any. If the user specifies the SPHERE keyword and processing for a base cluster is ended through this exit, processing for all AIXs related to the base cluster is also ended. If the user specifies the SPHERE keyword and the UIM attempts to end processing for an AIX that is related to a base cluster that was selected for processing, DFSMSDss invokes the UIM again. If the UIM attempts to end processing for the AIX again, DFSMSDss issues warning message ADR770W and ignores the request. If the user does not specify the SPHERE keyword, AIXs can be ended even if they are related to base clusters selected for processing. Ending a base cluster without the SPHERE keyword does not cause any related AIXs to be ended.

Note: See the "Note for Eioptions 21, 22, and 23" at the end of the description for eioption 23.

Bypass verification exit (Eioption 22)

This exit lets a user force DFSMSDss bypass serialization and security verification during data set copy, physical data set copy, dump, and restore processing for individual data sets. It also lets a user turn on a tolerate-migrated-volser indicator that forces DFSMSDss to restore a data set with a migrated volser. This exit is given control through the UIM at the start of data set copy, dump, and restore processing for each

data set. DFSMSDss provides the UIM with the data set name through the EIREC22 structure within the exit identification block, ADREID0. EIREC22 is shown in “ADREID0 data area” on page 609. The data set name provided is the original name prior to rename processing, if any, for COPY or RESTORE. The UIM returns to DFSMSDss with a return code in the exit identification block. The valid return codes are:

0

Continue normal processing

16

Bypass one or more of the following:

- If the Reset indicator (EI22RSET) is set on by the UIM for a VSAM data set during a logical data set dump, DFSMSDss resets the data-set-changed flag in the VTOC if the data set is successfully serialized and processed. DFSMSDss will not reset the data-set-changed flag for data sets dumped with RLS access and BWO data sets, even if EI22RSET is set on by the UIM.
- If the DB2 Source Bypass indicator (EI22DB2) is set on by the UIM during a logical data set copy operation, and RENAMEU is specified for a DB2 VSAM linear data set, then DFSMSDss does not verify that the old component duplicates the DB2 naming convention when it derives the new component name. Alternatively, if the EI22DB2 bit is set on, DFSMSDss generates a DB2 component name for the target component—provided the new cluster name (specified in the RENAMEU parameter) matches the DB2 naming convention.
- If the IMS indicator (EI22IMS) is set on, the caller is IMS.
- If the Shared SYSDSN ENQ indicator (EI22SSYS) is set on by the UIM for a VSAM data set during logical data set dump, DFSMSDss attempts to obtain a shared SYSDSN enqueue. If EI22SSYS is not set on, then the type of SYSDSN enqueue (shared or exclusive) that DFSMSDss attempts to obtain depends upon whether or not the SHARE keyword was specified.
- If the Mark-As-Recovery-Required indicator (EI22RRB) is set on by the UIM during logical restore, DFSMSDss marks the target data set as recovery required, provided that the target data set is SMS-managed.
- If the Log Information Passed indicator (EI22LINP) is set on by the UIM during a logical restore, then DFSMSDss uses the log parameter (EI22LPRM), the log stream ID (EI22LSID) and the LOGREPLICATE indicator (EI22LREP) as the log information for the target data set, provided that the target data set is SMS-managed. If EI22LINP is not set, or the target data set is not SMS-managed, then the contents of EI22LPRM, EI22LSID and EI22LREP are ignored.
- If the BWO_ALLOWED Passed indicator (EI22BWOP) is set on by the UIM during a logical restore, the DFSMSDss uses the BWO_ALLOWED field (EI22BWOA) for the target data set, provided the target data set is SMS-managed. If EI22BWOP is not set or the target data set is not SMS-managed, then the contents of EI22BWOA are ignored.
- Serialization is bypassed if the UIM exit turns on one of the following Bypass Serialization indicators:
 - EI22BSER - serialization is bypassed for the source data set for copy and dump processing, and for the target data set for copy and restore processing.
 - EI22_BYPASS_SOURCE_SER - serialization is bypassed for the source data set during a logical data set copy operation. Target data set serialization is never bypassed when this indicator is set ON. The Bypass Serialization indicator, EI22BSER, takes precedence over the Bypass Source Serialization indicator. Therefore, EI22_BYPASS_SOURCE_SER is ignored if EI22BSER is set ON.

DFSMSDss assumes that all necessary serialization has been performed by the invoker of DFSMSDss, but does not ensure that this is true. DFSMSDss performs normal serialization if the UIM exit does not turn on the bypass serialization indicator.

DFSMSDss does not delete or uncatalog data sets that were bypassed for serialization even if the DELETE or UNCATALOG keywords are specified. If a preallocated data set is not large enough during data set copy or restore operations and the bypass serialization indicator (EI22BSER) is on, DFSMSDss does not scratch and reallocate that target data set and the restore operation fails.

- RACF verification and all other data set security checking is bypassed if the UIM exit turns on the Bypass RACF indicator, EI22BSEC. If EI22BSEC is set on, DFSMSDss checks to ensure that the application program is authorized to bypass RACF and security processing. The application

program is authorized to bypass RACF and security processing if NOPASS was specified on the PPT statement of the SCHEDxx parmlib member. If the EI22BSEC indicator is on and NOPASS was specified, RACF verification and all other data set security processing including password checking is bypassed. DFSMSDss does not do any RACF authorization checks. DFSMSDss does not create any RACF profiles for the copied or restored target data set. DFSMSDss assumes that all necessary RACF authorization and security checking has already been performed by the invoker of DFSMSDss. For a copy or restore operation, RACF profiles are not created for the target data set.

If the user turns on the bypass RACF indicator and PASS was specified on the PPT statement of the SCHEDxx parmlib member, DFSMSDss sets the error flag, EIXERR, on in the exit identification block and invokes the UIM again. If the user sets the bypass RACF indicator again, DFSMSDss issues error message ADR772W and processing continues as normal. DFSMSDss still allows serialization to be bypassed and tolerates migrated volume serial number processing if those indicators are set.

DFSMSDss performs normal RACF and security processing if the UIM exit does not turn on the bypass RACF indicator.

- If the Tolerate-Migrated-Volser indicator is set on by the UIM, DFSMSDss takes special action to support the restoration of a non-VSAM data set with a volume serial number of MIGRAT for a logical data set restore operation in an SMS-managed environment or when the CATALOG option has been specified. DFSMSDss ignores the EI22BMIG indicator for copy and dump operations and VSAM data sets.

When DFSMSDss is restoring a non-VSAM data set and the Tolerate-Migrated-Volser indicator is set on, a catalog LOCATE is issued to determine the status of the data set being restored. Based on the result of that LOCATE, different actions are taken.

No catalog entry is found:

The data set being restored is not considered to be migrated. A normal DFSMSDss logical restore is performed. The Tolerate-Migrated-Volser indicator is ignored.

A catalog entry is found but the VOLSER is not MIGRAT:

The data set being restored is not considered to be migrated. Normal processing continues as though the indicator had not been set.

A catalog entry is found and the VOLSER is MIGRAT:

The data set is migrated and requires special processing. Instead of cataloging the data set after allocating it, DFSMSDss alters the existing MIGRAT volume serial number entry to the actual volume the data set was restored to. Once the catalog entry has been changed from MIGRAT to the new data set's volume serial numbers, the restore continues as normal.

Note: Because the migrated data set is not recalled during this restore, the user of this interface is responsible for deleting the migrated copy of the data set and updating the necessary control files.

DFSMSDss performs normal non-VSAM data set cataloging if the UIM exit does not turn on the tolerate-migrated-volser indicator.

- If the Extent Reduction bit (EI22EXTR) is set on by the UIM for a given data set and DFSMSDss is running in an SMS-managed environment, DFSMSDss tries to allocate the original volume where the data set was dumped.
- If the Restore-To-Like-Device bit (EI22LIKE) is set on by the UIM for a given data set and DFSMSDss is running in an SMS-managed environment, DFSMSDss tries to allocate the target data set on a device whose unit is the same as that of the source data set. If DFSMSDss can not allocate the target data set on a device whose unit type matches that of the source, the data set is not processed.
- Serialization by enqueueing on the major name of SYSDSN for a data set is bypassed if the UIM exit turns on the bypass SYSDSN indicator, EI22NSYS; other enqueues for a data set (for example, SYSVSAM) are not bypassed. The SYSDSN-level of enqueue serialization is bypassed for the source data set for DUMP and the target data set for RESTORE. DFSMSDss performs normal serialization if both EI22BSER and EI22NSYS are turned off.

DFSMSdss does not delete or uncatalog data sets if EI22NSYS is set, even if the DELETE or UNCATALOG keywords are specified. If EI22NSYS is on and the preallocated data set is too small, the data set is not scratched and reallocated, and the RESTORE fails.

If the UIM has set the "set reconnect" flag on, DFSMSdss then sets the reconnect flag in the catalog on. DFSMSdss ignores the EI22SFSM flag for VSAM data sets.

If the UIM exit sets the return code to EIRC16 but fails to turn on any of the bypass verification indicators, DFSMSdss ignores all bypass options and performs normally. If the UIM exit sets the return code to EIRC16 for a VSAM data set and turns on the tolerate-migrated-volser indicator, DFSMSdss does not treat it as an error condition and ignores the tolerate-migrated-volser indicator.

If a list of storage groups is passed by the UIM (EI22SGP) then the list is used to allocate SMS managed data sets during logical data set copy and restore. Users must take care to ensure that the storage groups passed do not conflict with the storage groups that would be assigned based on the storage class of the data set. A user may pass a maximum of 15 storage groups in the list. If the target data set is not SMS-managed then the storage groups passed are ignored.

If an ACS environment is passed by the UIM (EI22ACSEN), it is used in place of the DFSMSdss default environments (ALLOC for logical data set copy or RECOVER for logical data set restore). The ACS environment passed in is used when calling the management class and storage class ACS routines unless BYPASSACS indicates that one or both of the ACS routines should not be called. It is also passed to SMS on the allocation request. If EI22ACSEN specifies SPMGCLTR and a list of storage groups was not passed, DFSMSdss continues to process the data set in the default environments (ALLOC for logical data set copy or RECOVER for logical data set restore).

An application may set EI22DSSRL to either EISRLDB2, EISRLZFS, EISRLCICS or EISRLEXIT during a logical data set copy. If DFSMSdss is unable to obtain access to the data set, the service specified by EI22DSSRL is used to close the data set so that DFSMSdss is able to obtain access to process the data set. After the data set is processed the corresponding service is used to reopen the data set.

Note:

- When EI22DSSRL is set to either EISRLDB2, EISRLZFS, EISRLCICS or EISRLEXIT then RLS processing is bypassed.
- If EISRLZFS is set on by an application and the zFS is mounted then it must be mounted by only a single system. Other configurations are not supported.
- If EISRLDB2 or EISRLCICS is set on by the application and the data set being processed is open to CICS or DB2 then the system that is running the space management class transition must be a system that has the data set open. The restriction applies to CICS data sets that are opened for non-RLS access and DB2 subsystems that are not members of a data sharing group.

20

Disconnect exit

32

End function

36

End data set

Note: See the "Note for Eioptions 21, 22, and 23" at the end of the description for eioption 23.

Data set processed notification exit (Eioption 23)

This exit indicates to the UIM whether or not the logical copy, dump, or restore processing for individual data sets was successful. This exit is given control through the UIM at the conclusion of logical copy, dump, and restore processing for each data set. DFSMSdss provides the UIM with information through the EIREC23 structure within the exit identification block, ADREID0. EIREC23 is shown in ["ADREID0 data area" on page 609](#).

DFSMSdss provides the UIM with the following information at the conclusion of processing for logical dump of each data set:

- The data set name that was dumped.
- The RLS time stamps associated with the dump.
- A flag indicating whether or not the data set is marked "recovery required".
- A return code for the data set that indicates whether processing was successful:

0
Data set completely successful (informational)

4
Data set partially successful (warning)

8
Data set unsuccessful (error)

12
Ending error

16
Ending error

- The source SMS flag (DS1SMSFG) from the Format 1 DSCB
- The source data set organization (DS1DSORG) from the Format 1 DSCB for a non-VSAM data set
- Flags to indicate data set type for a VSAM data set
- The number of bytes processed in the source data set. When a backup is directed to a data set on DASD or tape, the number of bytes processed reflects the actual number of bytes read by DFSMSdss for the data set. When a backup is directed to an object storage cloud, the number of bytes reflects the number of tracks processed in the source data set times the number of bytes in a track. The reason for this difference is because when a backup is directed to an object storage cloud, the actual user data is not read into the host memory and the byte count is not available to DFSMSdss.

DFSMSdss provides the UIM with the following information at the conclusion of processing for a logical copy and a logical restore of each data set:

- The original data set name that was copied or restored. This is the data set name prior to rename processing, if any.
- The new data set name being copied or restored if rename processing was performed.
- A return code for the data set that indicates whether processing succeeded:

0
Data set completely successful (informational)

4
Data set partially successful (warning)

8
Data set unsuccessful (error)

12
Ending error

16
Ending error

- The source and target SMS flags (DS1SMSFG) from the Format 1 DSCB.
- The source and target data set organization (DS1DSORG) from the Format 1 DSCB for a non-VSAM data set.
- Flags to indicate data set type for a VSAM data set.
- A count of the volumes that the data set was copied or restored to.
- A list of the volume serial numbers (VOLSERs) that the data set was copied or restored to.

The UIM returns to DFSMSdss with a return code in the exit identification block. The valid return codes are:

0
Continue normal processing

20
Disconnect exit

32
End function

36
End data set

It does not undo any successful processing, but deletes the data set from the successfully-processed message list (if applicable) and includes it in the unsuccessfully-processed message list. It is the responsibility of the user of this interface to delete the copied, dumped, or restored data set from the target volumes. DFSMSDss continues processing with the next data set, if any. Refer to EIRC36 in [“Data set verification \(Eioption 21\)”](#) on page 596 for details on ending spheres.

Note for Eioptions 21, 22, and 23

If Eioptions 21, 22, and 23 end the function (UIM passes back a return code 32 to DFSMSDss) during a data set copy operation, DFSMSDss can be processing more than one data set at the time. This can happen if utilities are needed to process some of the data sets. DFSMSDss does the following before ending:

- No unprocessed data sets are scheduled for processing.
- All data sets in utility processing are allowed to complete normally.
- No new calls are made to exits 21, 22, or 23. This includes data sets in utility processing that are allowed to complete.
- Spheres that are being processed because of the SPHERE keyword and that are not complete at the end of the function have the target parts deleted to preserve sphere integrity. Preallocated spheres are left partially copied.
- Spheres being processed without the SPHERE keyword or individual sphere components being copied are left partially completed.

Concurrent copy initialization complete (Eioption 24)

This section contains intended programming interfaces.

DFSMSDss calls the UIM with option code 24 to inform it that the initialization of the concurrent copy session for a given data set or volume has completed. For full-volume or tracks operation, there is only one call (because there is only one input volume). For a physical data set operation, there is one call for each input volume. For a logical data set operation, there is one call for every data set. DFSMSDss does not call the UIM with this option code if the CONCURRENT keyword is not specified. DFSMSDss provides the UIM with information through the EIREC24 structure within the Exit Identification Block, ADREIB (in the ADREID0 macro).

DFSMSDss provides the UIM with the following information:

- A return code indicating whether or not the concurrent copy session initialization succeeded:

0
Concurrent copy session initialization was successful and any serialization on the data has been released.

4
Concurrent copy session initialization failed. If this is a logical data set operation, this data set is not processed using concurrent copy, but other data sets may be. If this is a full-volume, tracks, or physical data set operation, concurrent copy is not used. In either case, serialization is obtained and released as if CONCURRENT were not specified. When this exit is called, DFSMSDss may not be holding any serialization.

- A reason code providing further details about the status of the concurrent copy session initialization (always valid, even for a zero return code):

0

The concurrent copy operation is logically complete at this time (data movement has not been done yet).

4

All parts of the data being dumped or copied were not on hardware supporting concurrent copy.

8

Hardware limits exceeded.

12

System data mover failed.

16

Host limits exceeded.

20

System data mover not available.

24

Other host error.

28

Data set type not supported for concurrent copy.

32

The concurrent copy operation is logically and physically complete at this time (data movement has been done).

36

The data being processed requires the use of SnapShot, but the SnapShot software support is not available on the system.

- The volume serial of the volume on which the concurrent copy initialization was attempted (valid only if the reason code is not 16).
- The name of the data set on which the concurrent copy initialization was attempted (valid only if a logical data set operation is being performed).
- A flag indicating whether or not DFSMSDss has reset the data-set-changed flag in the VTOC for the data set (valid only if a logical data set dump operation is being performed).

The UIM returns to DFSMSDss with a return code in the Exit Identification Block. DFSMSDss continues to process based on that return code, as follows:

00

Continue normal processing

20

Disconnect this exit

32

The DFSMSDss function is ended (not valid for a physical data set dump or logical data set copy)

36

End data set (not valid for full or tracks operations)

Error handling is the same as described under [User Return Code](#).

Backspace physical tape record (Eioption 25)

This exit point is called when DFSMSDss requires the input to be repositioned to the previous tape block (tape or DASD). The EIRECPT is zero. The valid return codes are:

0

Continue normal processing

20

Disconnect exit

32

End function

Dump volume output notification (Eioption 26)

Exit 26 of the DFSMSDss UIM may be used to determine the following:

- When a new dump output volume has been added for each DDNAME being processed. (DFSMSDss sets EI26VOL with EI26DDN and EI26VSER.)
- When a dump output volume associated with a specific DDNAME receives a terminating error condition. (DFSMSDss sets EI26TERM with EI26DDN and EI26VSER. EI26VTRC contains the failing function return code.)
- When EI22IMS is on and an R0 count mismatch occurs during copy or dump processing for BWO(TYPEIMS) KSDS data sets. (DFSMSDss sets EI26ROCE. The data set name is located by EI26DSN.)
- When an application requests host-based encryption through the DFSMSDss application programming interface (API), and DFSMSDss overrides the request in favor of tape device encryption, DFSMSDss uses the Volume Output Notification Exit (Exit 26) to notify the caller. To indicate that the output tape volume used tape device encryption, DFSMSDss sets the EI26TWHE bit to one in the ADREID0 mapping.
- When a dump output volume associated with a specific DDNAME is closed. (DFSMSDss sets EI26VCLO with EI26DDN and EI26VSER. EI26VTRC contains the return code from CLOSE.)

Exit 26 is invoked with EI26VCLO set only during logical dump operations and only for output data sets residing on DASD devices.

If Exit 26 is not driven for a given DDNAME or is driven with EI26VTRC greater than zero, then the output data set associated with the DDNAME was not closed successfully and should not be depended upon for subsequent restore operations.

The UIM presents a return code to DFSMSDss in the Exit Identification Block. DFSMSDss continues to process based on that return code, as follows:

00

DFSMSDss continues normal processing

20

Disconnect this exit

32

DFSMSDss ends the function (valid only for EI26TERM)

Physical data set processed notification exit (Eioption 27)

This exit indicates to the UIM whether or not the physical data set copy for individual data sets was successful. This exit is given control through the UIM at the conclusion of physical data set copy processing for each data set. DFSMSDss provides the UIM with information through the EIREC27 structure within the exit identification block, ADREID0. EIREC27 is shown in [“ADREID0 data area” on page 609](#).

- The data set name.
- The new name of the data set.
- The new names of the VSAM data and index components.
- A return code indicating the success of processing the data set.
- The source data set organization (DS1DSORG).
- The source data set SMS flag (DS1SMSFG).
- Flags indicating the source VSAM data set type (KSDS, ESDS, LDS).
- Flags indicating the target VSAM data set type.

- A count of the number of tracks processed for this data set.
- The last volume indicator of the data set from the source data set. This will be valid for non-VSAM data sets only.
- If the first volume of a VSAM data set was processed, DFSMSDss will pass the total stripe count for the data component when the data set processed was a striped VSAM data set. The High Allocated RBA count for the data and index components will be passed as well.
- When processing a VSAM data set, DFSMSDss will pass the stripe number of the data component on this volume. 0 will be passed when the data set is not striped.
- When processing a VSAM data set, DFSMSDss will pass the number of RBAs processed for this data component on this volume.
- When processing an indexed VSAM data set, DFSMSDss will also pass the number of RBAs processed for this index component on this volume.
- The target volume serial as well as the source volume serial.
- An indication of whether the target data set is protected by a discrete RACF profile.

Target data set allocation notification exit (Eioption 28)

This exit indicates to the UIM how to direct allocation of target data sets during logical and physical restore and physical data set copy. The exit is given control through the UIM just prior to volume selection for target data set allocation for each data set being processed for logical data set operations. For physical operations, the exit is given control through the UIM just prior to target data set allocation. DFSMSDss provide the UIM with information through the EIREC28 structure within the exit identification block, ADREID0. EIREC28 is shown in [“ADREID0 data area” on page 609](#). The actual call includes the following information:

- Source Data Set Name (input)
- Target Data Set Name (input)
- Extended Attribute Value (DS1EATTR) (output)
- Format-9 DSCB (output)
- Target Data Set Creation Date (output)
- CA Reclaim Attribute Value (output)
- Linear data set defined as a zFS attribute value (output for physical data set COPY and RESTORE only)

The UIM presents a return code to DFSMSDss in the Exit Identification Block. DFSMSDss continues to process based on that return code, as follows:

00

DFSMSDss continues normal processing.

16

DFSMSDss will examine the paramter block passed back to determine if any parameters returned should be used for target allocation. If the UIM sets return code EIRC16 but did not indicate any of the parameters should be used, DFSMSDss continues normal processing.

20

Disconnect this exit.

Physical data set volume allocation notification exit (Eioption 30)

This exit indicates to the UIM that allocation to the specified SMS volume failed during physical data set RESTORE or COPY. The exit is given control through the UIM, after the allocation attempt for each data set attempted. DFSMSDss provides the UIM with information through the EIREC30 structure within the exit identification block, ADREID0. EIREC30 is shown in [“ADREID0 data area” on page 609](#).

The actual call includes the following information:

- Tracks required

- Overall exit return code
- Pointer to list of volumes
- Number of volumes in array
- VOLSER
- Volume attempted flag
- Volume used flag
- Volume return code
- Volume reason code

The UIM presents a return code to DFSMSdss in the Exit Identification Block. DFSMSdss continues to process based on that return code, as follows:

00

DFSMSdss continues normal processing.

16

The record has been modified.

20

Disconnect exit.

32

End function.

Store application metadata object (Eioption 31)

This exit lets the UIM write an application metadata object to be stored in the cloud. This exit point is given control through the UIM:

Logical data set processing: At the conclusion of processing for each data set.

Full volume processing: At the beginning of processing.

The EIRECPTR points to the buffer that contains the data to be stored and EIRECLEN is the length of the meta data to be stored. When EIRC08 is passed, this instructs DSS to store the buffer passed back as an object. The maximum length cannot be greater than 4063. The valid return codes are:

0

Continue normal processing

8

Store object

32

End function

Note:

1. On a Store request (EIRC08), if the length is zero or greater than 4063 DFSMSdss sets the error flag, EIXERR, on in the exit identification block and invokes the UIM again.
2. Application metadata objects are named as follows:

Logical data set: object_prefix/dsname/APPMETA where object_prefix is the object prefix specified on the DFSMSdss control statements, dsname is the data set name processed, APPMETA is a fixed string.
Full volume: object_prefix/volser/APPMETA where object_prefix is the object prefix specified on the DFSMSdss control statements, volser is the volume serial id of the volume being processed, APPMETA is a fixed string.

Retrieve application metadata object (Eioption 32)

This exit lets the UIM read an application metadata object stored in the cloud. This exit point is given control through the UIM: Logical data set processing: At the beginning of logical data set restore processing for each data set. Full volume processing: At the beginning of full volume restore processing.

The EIRECPTTR points to the buffer that contains the object and EIRECLEN is the length of the metadata stored. The maximum length cannot be greater than 4063. The valid return codes are:

- 0**
Continue normal processing.
- 32**
End function.

Output object notification exit (Eioption 33)

This exit indicates to the UIM the object name that was written to the cloud during logical dump processing for meta data and data objects associated to individual data sets that were stored to the cloud whether they were successful or not, as indicated by the return code. DFSMSDss provides the UIM with information through the EIREC33 structure within the exit identification block, ADREID0. EIREC33 is shown in [“ADREID0 data area” on page 609](#).

DFSMSDss provides the UIM with the following information at the conclusion of processing for logical dump of each data set:

- The header or data set name that was dumped
- An indication of object type (meta object or data object)
- An indication if the data object was encrypted with TCT encryption. Not applicable for meta data objects.
- An indication if the data object was compressed with TCT compression. Not applicable for meta data objects.
- The track count for data objects.
- The uncompressed byte count for meta and data objects.
- The compressed byte count for data objects. Not applicable for meta data objects.
- A return code for the data set that indicates whether processing was successful:

- 0**
Object store successful (informational)
- 8**
Object store unsuccessful (error)

Cloud bypass verification (Eioption 35)

This exit lets the UIM override control the cloud operation using supplied options. This exit point is given control through the UIM at the beginning of dump operations when the output is cloud object storage prior to performing authentication with the cloud provider.

DFSMSDss provides the UIM with information through the EIREC35 structure within the exit identification block, ADREID0. EIREC35 is shown in the EIREC35 table.

- 0**
Continue normal processing
- 15**
Record modification

z/OS UNIX file path notification (Eioption 41)

This exit lets the UIM end dump and restore processing for individual z/OS UNIX files. This exit is given control through the UIM at the start of file dump and restore processing for each z/OS UNIX file. DFSMSDss provides the UIM with the relative path name (from the working directory) through the EIREC41 structure within the exit identification block, ADREID0.

The UIM returns to DFSMSDss with a return code in the exit identification block. The valid return codes are:

- 0** Continue normal processing
- 20** Disconnect exit
- 32** End function
- 36** End file DFSMSdss continues processing with the next z/OS UNIX file, if any. This return code has the same behavior as excluding the file from processing. If the file being processed is a directory, its children are also excluded.

z/OS UNIX file path bypass notification (Eioption 42)

This exit lets a user force DFSMSdss to bypass normal processing that is made available to application programmers. This exit lets the UIM end dump and restore processing for individual z/OS UNIX files. This exit is given control through the UIM at the start of file dump processing for each z/OS UNIX file. DFSMSdss provides the UIM with the relative path name (from the working directory) through the EIREC42 structure within the exit identification block, ADREID0.

The UIM returns to DFSMSdss with a return code in the exit identification block. The valid return codes are:

- 0** Continue normal processing
- 16** If the Reset indicator (EI42RSET) is set on by the UIM, DFSMSdss resets the last backup date (ATTRREFTIME64 in the files attributes) if the regular file is successfully processed
- 20** Disconnect exit
- 32** End function.
- 36** End file DFSMSdss continues processing with the next z/OS UNIX file, if any. This return code has the same behavior as excluding the file from processing. If the file being processed is a directory, its children are also excluded.

z/OS UNIX file path processed notification (Eioption 43)

This exit indicates to the UIM that a z/OS UNIX file has been processed. DFSMSdss provides the UIM with the relative path name (from the working directory), of the file that was processed along with some file attributes through the EIREC43 structure within the exit identification block, ADREID0.

DFSMSdss returns the following return codes indicating whether processing was successful:

- 0** Successfully processed
- 4** Partially successful (warning)
- 8** Unsuccessful (error)
- 12** Post processing error.

The UIM returns to DFSMSdss with a return code in the exit identification block. The valid return codes are:

- 0** Continue normal processing

20

Disconnect exit

32

End function.

z/OS UNIX file path clone initialization notification (Eioption 44)

This exit notifies the result of the clone initialization operation for regular files when CLONE(PREFFERED|REQUIRED) is specified. The exit lets the UIM end dump processing for of the file or terminate the task. DFSMSdss provides the UIM with the relative path name (from the working directory) of the file being processed through the EIREC44 structure within the exit identification block, ADREID0.

The return code and reason codes can be identified in the z/OS UNIX System services Messages and Codes Return codes reference.

The UIM returns to DFSMSdss with a return code in the exit identification block. The valid return codes are:

0

Continue normal processing

20

Disconnect exit

32

End function

36

End file DFSMSdss continues processing with the next z/OS UNIX file, if any. This return code has the same behavior as excluding the file from processing.

Avoiding lockout

Refer to [“Avoiding lockout” on page 562](#) for a description of the ENQ scheme to prevent lockout.

Application interface summary

For Record Processing, if the UIM makes any changes to the presented record (assuming it can validly do so), return code 16 must be returned or the changes are ignored. If the record is to be replaced in total, return code 4 must be returned or the original record is used. If a record is to be inserted before the current record, return code 8 must be returned or the record is ignored. If the current record is to be deleted, return code 12 must be returned or the current record is processed. If the exit is no longer interested in processing any records appearing at the current DFSMSdss exit point, return code 20 must be returned or the exit is called at the next visitation of DFSMSdss to the exit point. The current record is still processed in either event. If you want to receive only user statistical records at any specific exit, return code 24 must be returned. If you want to process a WTOR within the UIM, the response must be returned to DFSMSdss with a return code 28.

If a record is being returned that is longer than the original record, the exit must either replace the record in the area pointed to by the original record pointer (as long as it does not exceed the length in EIRECALN) or supply another area and store the address to the record in the original record pointer field. In either case, the length must be stored in the original record length field and a return code of 4 must be used.

If a record is being returned that is shorter than the original record, the exit can supply an area and store the address to the record in the original record pointer field or, if allowed, can replace the original record with the shorter one. In either case, the length of the new record must be placed in the original record length field. If the new area option was used, return code 4 must be used. If the new record overlaid the original, return code 16 must be used. [“ADREID0 data area” on page 609](#) describes the data area corresponding to the DFSMSdss exit identification block.

For Data Set Processing, Eioptions 21, 22, and 23 are three exits that give you added control over data set processing during a logical data set copy, dump, and restore operations. These exits are called

immediately before and immediately after processing each data set, on a data set by data set basis, and allow you to make a number of processing changes.

Eioptions 21 and 22 are called, consecutively, at the start of processing of each data set. DFSMSdss passes the name of the data set being processed to each exit. Based on that name, each exit permits the following:

- End processing of that one data set (return code 36).
- End processing of the entire functional task (return code 32).
- Disconnect the exit so that it will not be called before subsequent data sets task processing (return code 20).
- Perform nothing (return code 0).
- Eioption 22 only: Modify how the data set will be processed (return code 16). See [“Bypass verification exit \(Eioption 22\)” on page 596](#) for details.

Eioption 23 is called immediately after a data set is processed. Eioption 23 lets you end processing of both the data set and the task, disconnect the exit, or do nothing.

For z/OS UNIX file processing, Eioptions 41, 42, 43, and 44 are exits that give you added control over file processing during file dump and restore operations. These exits (41-43) are called immediately before and after each file and allows you to make a number of changes. Each path that is processed can consist of many files (parent directories).

DFSMSdss passes the relative path (from the working directory) of the file being processed to each exit.

Eioptions 41 and 42 are called consecutively at the beginning of every z/OS UNIX file during dump operations. Eioption 41 is also called for Restore at the beginning of every z/OS UNIX file. When processing from a CLONE, see description for Eioption 44. Based on the path name, each exit permits the following:

- End processing of that one file (return code 36)
 - **Note:** If the file being processed is a directory, any child files are excluded from processing.
- End processing of the entire functional task (return code 32)
- Disconnect the exit so that it is not called before subsequent file task processing (return code 20)
- Perform nothing (return code 0)
- Eioption 42 only: Modify how the file is processed (return code 16).

Eioption 44 is only called for regular files during dump operations when CLONE(PREFERRED|REQUIRED) is specified. DFSMSdss attempts to clone all regular files prior to backing them up so that any selected regular files can be released to the application. This means that Eioptions 41 and 42 are called for regular files before they are called for any of its parent directories when processing from a clone. Once all clones are created, processing (backing up) the file does not occur until after its parents directories are processed. Eioption 43 for the regular file is called after its parent directories. Eioption 44 lets you end processing of both the file and the task.

Eioption 43 is called immediately after a file is processed. Eioption 43 lets you end processing of both the file and the task, disconnect the exit, or do nothing.

ADREID0 data area

1	ADREID0					
	OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
	=====	=====	=====	=====	=====	=====
	0	(0)	STRUCTURE	42	ADREIB	
	0	(0)	SIGNED	2	EIDLLEN	LENGTH OF ADREIB - 2
	2	(2)	CHARACTER	4	EIID	BLOCK IDENTIFIER EBCDIC "EIDB"
	6	(6)	SIGNED	4	EITSKID	TASK ID NUMBER
	10	(A)	BIT(32)	4	EIXALLOW	USER EXIT ALLOWANCE OPTIONS
	10	(A)	BIT(8)	1	EIXALLOW0	ALLOWANCE OPTION BYTE 1
		1			FIXREP	ALLOW REPLACE OF RECORD

		.1..		EIXINS	ALLOW INSERTION OF RECORD
		..1.		EIXDEL	ALLOW DELETION OF RECORD
		...1		EIXMOD	ALLOW MODIFICATION OF RECORD
	 1...		EIXDIS	ALLOW DISCONNECT OF EXIT
	1..		EIXWTOR	ALLOW WTOR RESPONSE
	1.		EIXSTAT	ALLOW SELECTION OF USER STATS
	1		EIXTERM	ALLOW FUNCTION TERMINATION
11	(B)	BIT(8)	1	EIXALOW1	ALLOWANCE OPTION BYTE 2
		1...		EIXTDSET	ALLOW DATA SET TERMINATION
		.111 1111		*	UNUSED
12	(C)	BIT(8)	1	EIXALOW2	ALLOWANCE OPTION BYTE 3
12	(C)	BIT(8)	1	*	RESERVED
13	(D)	BIT(8)	1	EIXALOW3	ALLOWANCE OPTION BYTE 4
		1...		EIXERR	DISALLOWED OPTION ATTEMPTED
		.111 1111		*	RESERVED
14	(E)	SIGNED	2	EIOPTION	PROCESSING OPTION
16	(10)	SIGNED	2	EIRETCOD	EXIT RETURN CODE
18	(12)	SIGNED	4	EIRECALN	RECORD AREA LENGTH
22	(16)	SIGNED	4	EIRECLEN	ORIGINAL RECORD LENGTH
26	(1A)	ADDRESS	4	EIRECPTR	ORIGINAL RECORD ADDRESS
30	(1E)	ADDRESS	4	EIUSEPTR	USER DATA AREA ADDRESS
34	(22)	ADDRESS	4	EIDDDID	EIOP06 DDNAME/VOLID PTR
34	(22)	ADDRESS	4	EIXMWPL	EIOP11 WTO PLIST PTR
38	(26)	BIT(32)	4	EIXFLAGS	OTHER FLAGS
38	(26)	BIT(8)	1	EIXFLAG0	FLAG BYTE 0
		1...		EIXABEND	FOR EIOP02 ONLY, 1=MESSAGE IS ADR013 INDICATING AN ABEND CONDITION
		.1..		EIXNTERR	FOR EIOP02 ONLY, 1=MESSAGE IS TYPE 'E' AND IS NOT 324 OR 347
		..1.		EIXWNGOK	FOR EIOP14 ONLY, 1=WARNING MSGS WERE ISSUED AND NONE ARE FATAL
		...1		EIXTRKER	TRACK IS IN ERROR
	 1111		*	RESERVED
39	(27)	BIT(8)	1	EIXFLAG1	FLAG BYTE 1
		1...		EIXTRC32	UIM SET EIRETCOD 32
		.1..		*	RESERVED
		..1.		EIXTDUMS	DUMMY SOURCE TAPE
		...1		EIXTDUMT	DUMMY TARGET TAPE
	 1...		EIXTISXM	APPL USING XMAPI
	1..		EIXDASYS	XMAPI ALLOCATION ERR
	1.		EIXTWERR	XMAPI TAPE WRITE ERR
	1		EIXTRERR	XMAPI TAPE READ ERR
40	(28)	BIT(8)	1	EIXFLAG2	FLAG BYTE 2
40	(28)	BIT(8)	1	*	RESERVED
41	(29)	BIT(8)	1	EIXFLAG3	FLAG BYTE 3
41	(29)	BIT(8)	1	*	RESERVED
42	(2A)	CHARACTER	0	*	FORCE SIZE OF CONTROL BLOCK TO WORD BOUDARY

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	24	EIDDDINFO	UIM-06 INFO
0	(0)	CHARACTER	8	EIDDDNAME	O/P DEVICE DDNAME
8	(8)	CHARACTER	6	EIVOLID	O/P VOLSER
14	(E)	UNSIGNED	1	EIRETC	RESERVED FOR RETCODE
15	(F)	UNSIGNED	1	EI06FLGS	EXIT 06 FLAGS
		1...		EI06FTLR	RESERVED FOR HSM USE
		.1..		EI06THWE	RESERVED FOR HSM USE
		..11 1111		*	OUTPUT DEVICE ENCRYPTS
				*	RESERVED
16	(10)	ADDRESS	4	EI06DCB@	RESERVED FOR HSM USE
20	(14)	CHARACTER	4	*	RESERVED

RECORD PRESENTED BY EXIT 00 (FUNCTION STARTUP):

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	20	EIREC00	
0	(0)	UNSIGNED	1	EI00SBPL	SHARED SUBPOOL NUMBER
1	(1)	BIT(8)	1	EI00FLGS	startup flags
		1...		EI00SENG	short VTOC enqueue
		.1..		EI00NONF	1 = Don't allow IGWNOTIF call 0 = Allow IGWNOTIF call
		..1.		EI00NOLK	1= NO ADRLOCK ENQ/DEQ
		...1		EI00NENQ	1= NO INPUT VOLUME
					SERIALIZATION. VALID DURING PHYSICAL COPY AND DUMP

 1...	EI00BSEC	FUNCTION STARTUPS 1= BYPASS SECURITY VERIFICATION. VALID DURING COPY FULL/TRACKS, DUMP FULL/TRACKS, AND RESTORE FULL/TRACKS FUNCTION STARTUPS, UNIX PATH DUMP/RESTORE
1..	EI00MVOLRECOV	1=CALLER IS PERFORMING A MULTI VOLUME RECOVERY USING PHYSICAL DATA SET RESTORE AND IT IS ACCEPTABLE FOR DFSMSDSS TO SCRATCH PREALLOCATED TARGET DATA SETS THAT ARE NOT THE CORRECT SIZE
1.	EI00BSEB	1=BYPASS SERIALIZATION OF THE DATA SETS BEING RESTORED DURING A PHYSICAL DATA SET RESTORE OPERATION
1	EI00_NOEXCL_NSPLOCK	1=BYPASS EXCLUSIVE ADRLCK NONSPEC ENQ
2	(2) CHARACTER	6 EI00SVOL	SOURCE VOLSER. VALID DURING COPY FULL/TRACKS, DUMP FULL/ TRACKS, AND RESTORE FULL/TRACKS FUNCTION STARTUPS. FOR RESTORE, THIS FIELD CONTAINS VOLSER OF THE INPUT VOLUME WHERE THE DUMP DATA SET RESIDES
8	(8) CHARACTER 1...	1 EI00FLG2 EI00SWNCRYPT	2ND FLAG AREA SOFTWARE ENCRYPTION WAS REQUESTED (RSA OR KEYPASSWORD WAS SPECIFIED)
	.1..1.	EI00CANSFE EI00MIXDEV	CANCEL SOFTWARE ENCRYPT MIXTURE OF DEVICES PERFORMING AND NOT PERFORMING HARDWARE ENCRYPTION
	...1 1111	EI00ZCOMP *	ZCOMPRESSION ATTEMPTED RESERVED
9	(9) CHARACTER	3 *	RESERVED
12	(C) UNSIGNED	4 EI00CIRBA_DA	RBA OF VVR CI DATA/NOVSM
16	(10) UNSIGNED	4 EI00CIRBA_IX	RBA OF VVR CI INDEX

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 14:
DSS MESSAGE NUMBER ASSOCIATED WITH NON ZERO RETCODE

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	32	EIREC14	
0	(0)	CHARACTER	4	EI14RC	DSS RETURN CODE
4	(4)	CHARACTER	4	EI14MESS	DSS MESSAGE
4	(4)	CHARACTER	3	EI14MNUM	MESSAGE NUMBER
7	(7)	CHARACTER	1	EI14MTYP	MESSAGE TYPE
8	(8)	CHARACTER	8	EI14CPUT	DSS CPU TIME
16	(10)	UNSIGNED	1	EI14ZCSV	ZCOMPRESS % SAVED
16	(10)	UNSIGNED	1	EI14TCT_CSAVE	TCT COMPRESS % SAVED
17	(11)	CHARACTER	15	*	UNUSED

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 20: VOLUME
NOTIFICATION EXIT.

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	68	EIREC20	
0	(0)	CHARACTER	44	EI20DSN	DATA SET NAME/CLUSTER NAME
44	(2C)	BIT(8)	1	EI20FLGS	SOME FLAGS:
	1...			EI20VSAM	1=DATA SET IS VSAM 0=DATA SET IS NONVSAM
	.1..			EI20RACF	1=DATA SET IS PROTECTED BY A DISCRETE RACF PROFILE
	..1.			EI20LVOL	1=LAST VOLUME INDICATOR ON FOR NON-VSAM DS
	...1			E20F9ATT	F9 ATTRS LOST FLAG
 1111			*	RESERVED
45	(2D) CHARACTER	1 *			RESERVED
46	(2E) UNSIGNED	1		EI20DA#	NUMBER OF DATA COMPONENTS

47	(2F) UNSIGNED	1	EI20IX#	NUMBER OF INDEX COMPONENTS (0 IF NONVSAM)
48	(30) ADDRESS	4	EI20DA@	POINTER TO DATA COMPONENT INFO FOR VSAM/DATA SET INFO FOR NONVSAM
52	(34) ADDRESS	4	EI20IX@	POINTER TO INDEX COMPONENT INFO FOR VSAM/0 FOR NONVSAM
56	(38) SIGNED	2	*	PADDING (UNUSED) THE EI20SCNT, EI20DSHA, AND EI20ISHA FIELDS ARE PROVIDED WHEN PROCESSING THE FIRST VOLUME OF A MULTI VOLUME DATA SET
58	(3A) SIGNED	2	EI20SCNT	TOTAL STRIPE COUNT FOR THIS VSAM DATA SET
60	(3C) UNSIGNED	4	EI20DSHA	VSAM DATA COMPONENT HIGH ALLOCATED RBA CNT
64	(40) UNSIGNED	4	EI20ISHA	VSAM INDEX COMPONENT HIGH ALLOCATED RBA CNT

FOR A NONVSAM DATA SET, EI20DA@ WILL POINT TO A SINGLE EI20DSI STRUCTURE. FOR A VSAM CLUSTER, EI20DA@ POINTS TO AN ARRAY OF STRUCTURES (ONE FOR EACH DATA COMPONENT), AND EI20IX@ POINTS TO A SIMILAR ARRAY FOR THE INDEX COMPONENTS

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	68	EI20DSI	DATA SET INFO
0	(0)	CHARACTER	44	EI20CON	COMPONENT NAME (BLANKS FOR NONVSAM)
44	(2C)	UNSIGNED	2	EI20NVOL	# OF VOLUMES IN DS
46	(2E)	CHARACTER	2	*	RESERVED
48	(30)	CHARACTER	20	EI20VLI	VOLUME INFORMATION
48	(30)	CHARACTER	6	EI20VOL	VOLSER
54	(36)	SIGNED	2	E20DSTP#	STRIPE NUMBER FOR THE DATA COMPONENT. 0 FOR NON-STRIPED DATA SETS
56	(38)	UNSIGNED	4	EI20RBA	RBA TOKEN THE EI20DRBA FIELD IS CALCULATED BY SUBTRACTING THE HIGH RBA FIELDS FROM THE LOW RBA FLDS.
60	(3C)	UNSIGNED	4	EI20DRBA	TOTAL NUMBER OF RBAS PROCESSED FOR THIS COMPONENT
64	(40)	SIGNED	2	EI20VLSQ	VOLUME SEQUENCE NUMBER FOR THE DATA SET
66	(42)	SIGNED	2	*	UNUSED

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 21:
DATA SET VERIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	44	EIREC21	
0	(0)	CHARACTER	44	EI21DSN	DATA SET/CLUSTER NAME

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 22:
BYPASS VERIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	124	EIREC22	
0	(0)	CHARACTER	44	EI22DSN	DATA SET/CLUSTER NAME
44	(2C)	BIT(8)	1	EI22FLGS	EXIT 22 FLAGS:
	1... ..			EI22BSER	1=BYPASS SERIALIZATION
	.1..			EI22BSEC	1=BYPASS DATASET LEVEL, STORCLAS & MGMTCLAS SECURITY CHECKS. ALSO BYPASS JES3 INTEGRITY CHECKS AND DO NOT CREATE DISCRETE DATASET PROFILES.
	..1.			EI22BMIG	1=TOLERATE MIGRATED VOLSER FOR NONVSAM DSETS (RESTORE ONLY)
	...1			EI22NSYS	@DS007 BYPASS SYSDSN ENQ SWAPPED W/ EI22SIRS

.... 1...	EI22LIKE	RESTORE/COPY DATA SET TO A LIKE DEVICE TYPE
.... .1..	EI22EXTR	RESTORE DATA SET TO EXTENT REDUCTION VOLUME
.... ..1.	EI22SIRS	SET INCMPLT RECALL SWAPPED W/ EI22NSYS
.... ...1	EI22DB2	SET DB2 FLAG IN A RENAME

THE FOLLOWING WERE REMOVED FROM THE MAP OF THE RECORD
PRESENTED TO EXIT 22 BUT ARE NOW USED

=====					
45	(2D)	CHARACTER	5	EI22SFLG	SOURCE DATA SET FLGS
45	(2D)	CHARACTER	1	E22SSMSF	SOURCE SMS FLAGS
46	(2E)	CHARACTER	2	E22SDSRG	SOURCE DATA SET ORG
48	(30)	BIT(16)	2	E22SVFLG	SRCE VSAM DSET FLAGS
		1... ..		E22SESDS	1=ESDS DATA SET
		.1.. ..		E22SKSDS	1=KSDS DATA SET
		..1.		E22SKRDS	1=KRDS DATA SET
		...1		E22SLDS	1=LINEAR DATA SET
	 1...		E22SRRDS	1=RRDS DATA SET
	1..		E22SPSSI	1=PAGE/SWAP/STGINDEX, ETC.
					UNSUPRTED DATA SETS
	1.		E22SVVDS	1=VVDS DATA SET
	1		E22SBCS	1=BCS DATA SET
49	(31)	1... ..		E22SAIX	1=AIX DATA SET
		.1.. ..		E22SVRRD	VRRDS DATASET
		..11 1111		*	RESERVED
50	(32)	BIT(8)	1	EI22FLG2	MORE EXIT 22 FLAGS
		1... ..		EI22RSET	RESET DS CHANGED FLAG
		.1.. ..		EI22SSYS	GET SHARED SYSDSN ENQ
		..1.		EI22RRB	MARK TGT RECOVERY REQ'D
		...1		EI22LINF	LOG INFO PASSED, SEE EI22LPRM AND EI22LSID
	 1...		EI22BWOP	BWO_ALLOWED PASSED, SEE EI22BWOA
	1..		EI22IMS	TYPE=TYPEIMS PROCESS
	1.		EI22BWOE	ENQ ON BWODSN ONLY, NO ADR OR SYS ENQ'S
	1		EI22_BYPASS_SOURCE_SER	BYPASS SOURCE SERIALIZAT - EI22BSER HAS PRECEDENCE
51	(33)	CHARACTER	1	EI22LPRM	LOG PARAMETER FOR TGT
52	(34)	CHARACTER	26	EI22LSID	LOG STREAM ID FOR TGT
78	(4E)	BIT(8)	1	EI22FLG3	MORE EXIT 22 FLAGS
		1111		EI22BWOA	BWO_ALLOWED
		... 1...		EI22BRLS	bypass RLS quiesce
	1..		EI22SFSM	set reconnectable flag
	1.		EI22LREP	set LOGREPLICATE flag
	1		EI22BRBK	bypass reblocking
79	(4F)	UNSIGNED	1	EI22DSSRL	DATA SET SERIALIZATION ERROR ACTION
80	(50)	CHARACTER	8	EI22ACSEN	ACS ENVIRONMENT
88	(58)	ADDRESS	4	EI22SGP	STORAGE GROUP LIST PTR
92	(5C)	CHARACTER	32	*	AVAILABLE

RECORD PRESENTED BY EXIT 22 (BYPASS VERIFICATION):
EISGLIST IS BASED ON (EI22SGP).

=====					
OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
=====					
0	(0)	STRUCTURE	484	EISGLIST	UIM-22 storage group list
0	(0)	CHARACTER	4	EISGHDR	Storage group list header
0	(0)	UNSIGNED	1	EISGCNT	Number of storage group
list entries					
=====					
1	(1)	CHARACTER	3	*	For alignment
4	(4)	CHARACTER	32	EISGENT(15)	Storage group entry array
4	(4)	SIGNED	2	EISGLEN	length of storage group
6	(6)	CHARACTER	30	EISGNAM	Name of storage group

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 23:
DATA SET PROCESSED NOTIFICATION EXIT

=====					
OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
=====					
0	(0)	STRUCTURE	*	EIREC23	

0	(0)	CHARACTER	141	EI23CNST	CONSTANT LENGTH PORTION OF CONTROL BLK
0	(0)	CHARACTER	44	EI23DSN	DATA SET/CLUSTER NAME
44	(2C)	CHARACTER	44	EI23NEWN	NEW DSET/CLUSTER NAME
88	(58)	SIGNED	2	EI23DSRC	RETURN CODE FOR DATA SET PROCESSING
90	(5A)	CHARACTER	5	EI23SFLG	SOURCE DATA SET FLAGS
90	(5A)	CHARACTER	1	E23SSMSF	SOURCE SMS FLAGS
91	(5B)	CHARACTER	2	E23SDSRG	SOURCE DATA SET ORG
93	(5D)	BIT(16)	2	E23SVFLG	SOURCE VSAM DSET FLAGS
		1... ..		E23SESDS	1=ESDS DATA SET
		.1.. ..		E23SKSDS	1=KSDS DATA SET
		..1.		E23SKRDS	1=KRDS DATA SET
		...1		E23SLDS	1=LINEAR DATA SET
	 1...		E23SRRDS	1=RRDS DATA SET
	1..		E23SPSSI	1=PAGE/SWAP/STGINDEX, ETC.
	1.		E23SVVDS	1=VVDS DATA SET
	1		E23SBCS	1=BCS DATA SET
94	(5E)	1... ..		E23SAIX	1=AIX DATA SET
		.1.. ..		E23BCSEL	1=BASE CLUSTER SELECTED
		..1.		E23SVRRD	VRRDS DATASET
		...1		EI23RRB	RECOVERY REQUIRED
	 1111		*	UNUSED
95	(5F)	CHARACTER	5	EI23TFLG	TARGET DATA SET FLAGS
95	(5F)	CHARACTER	1	E23TSMSF	TARGET SMS FLAGS
96	(60)	CHARACTER	2	E23TDSRG	TARGET DATA SET ORG
98	(62)	BIT(16)	2	E23TVFLG	TARGET VSAM DSET FLAGS
		1... ..		E23TESDS	1=ESDS DATA SET
		.1.. ..		E23TKSDS	1=KSDS DATA SET
		..1.		E23TKRDS	1=KRDS DATA SET
		...1		E23TLDS	1=LINEAR DATA SET
	 1...		E23TRRDS	1=RRDS DATA SET
	1..		E23TPSSI	1=PAGE/SWAP/STGINDEX, ETC.
	1.		E23TVVDS	1=VVDS DATA SET
	1		E23TBCS	1=BCS DATA SET
99	(63)	1... ..		E23TAIX	1=AIX DATA SET
		.1.. ..		E23TVRRD	VRRDS DATASET
		..11 1111		*	RESERVED
100	(64)	CHARACTER	16	EI23RLST	RLS TIME STAMPS
100	(64)	CHARACTER	8	EI23GMT	RLS GMT TIME STAMP
108	(6C)	CHARACTER	8	EI23LOC	RLS LOCAL TIME STAMP
116	(74)	CHARACTER	8	E23BYTES	TOTAL TRACK BYTES RESTORED, COPIED, OR DUMPED
124	(7C)	CHARACTER	1	E23FLGS	MISC FLAGS
		1... ..		E23BSET	1=E23BYTES SET
		.1.. ..		E23BPDS	1=BROKEN PDS
		..1.		E23BPSE	E23BYTES IS FOR PSE@0W25343
		...1		E23F9ATT	F9 ATTRIBS NOT RESTD
	 1111		*	AVAILABLE FOR USE
125	(7D)	CHARACTER	6	EI23DNAM	NAME OF DEVICE NEEDED TO DO NOPACKING FOR THE BROKEN PDS (I.E. A LIKE DEVICE)
131	(83)	CHARACTER	9	*	RESERVED
140	(8C)	UNSIGNED	1	EI23VOL#	NUMBER OF VOLUMES
141	(8D)	CHARACTER	6	EI23VSR(*)	VOLSER ARRAY

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 24:
CONCURRENT COPY INITIALIZATION COMPLETE

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	83	EIREC24	
0	(0)	UNSIGNED	2	EI24RTCD	RETURN CODE
2	(2)	UNSIGNED	2	EI24RSCD	REASON CODE
4	(4)	UNSIGNED	4	*	RESERVED
8	(8)	CHARACTER	6	EI24VOL	VOLUME SERIAL
14	(E)	CHARACTER	44	EI24DSN	DATA SET NAME
58	(3A)	BIT(8)	1	EI24FLGS	FLAGS
		1... ..		EI24RSET	DS CHANGE FLAG RESET
		.1.. ..		E24SAIX	1=AIX DATA SET
		..1.		E24BCSEL	1=BASE CLUSTER SELECTD
		...1 1111		*	UNUSED
59	(3B)	CHARACTER	24	*	AVAILABLE

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 26:
DUMP OUTPUT VOLUME MOUNT NOTIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	188	EIREC26	
0	(0)	BIT(32)	4	EI26TYPE	EXIT TYPE
		1... ..		EI26VOL	OUTPUT VOLUME NOTIFICATION
		.1.. ..		EI26TERM	OUTPUT VOLUME TERMINATED
		..1.		EI26R0CE	BWO R0 COUNT ERROR
		...1		EI26VCLO	output volume close - only for DASD outputs during logical dump operations
	 1...		EI26THWE	TAPE DEVICE WILL BE ENCRYPTING DATA IN HARDWARE
0	(0)	BIT(27) POS(6)	4	*	RESERVED FOR EXPANSION
4	(4)	CHARACTER	64	EI26DSN	DSNAME IF EI26R0CE = '1'B
68	(44)	UNSIGNED	1	EI26DSNL	LENGTH OF DSNAME
69	(45)	UNSIGNED	3	*	RESERVED FOR ALIGNMENT
72	(48)	CHARACTER	8	EI26DDN	output dd name if EI26VOL, EI26TERM, or EI26VCLO set
80	(50)	CHARACTER	6	EI26VSER	VOLSER - present if EI26VOL, EI26TERM, or EI26VCLO set
86	(56)	CHARACTER	2	*	RESERVED FOR ALIGNMENT
88	(58)	UNSIGNED	4	EI26VTRC	return code for volume term and volume close
92	(5C)	CHARACTER	96	*	RESERVED FOR EXPANSION

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 27:
PHYSICAL DATA SET PROCESSED NOTIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	264	EIREC27	
0	(0)	CHARACTER	264	EI27CNST	CONSTANT LENGTH PORTION OF CONTROL BLK
0	(0)	CHARACTER	44	EI27DSN	DATA SET/CLUSTER NAME
44	(2C)	CHARACTER	44	EI27NEWN	NEW DSET/CLUSTER NAME
88	(58)	SIGNED	2	EI27DSRC	RETURN CODE FOR DATA SET PROCESSING
90	(5A)	CHARACTER	5	EI27SFLG	SOURCE DATA SET FLAGS
90	(5A)	CHARACTER	1	E27SSMSF	SOURCE SMS FLAGS
91	(5B)	CHARACTER	2	E27SDSRG	SOURCE DATA SET ORG
93	(5D)	BIT(16)	2	E27SVFLG	SOURCE VSAM DSET FLAGS
		1... ..		E27SESDS	1=ESDS DATA SET FOR PHYSICAL DATA SET COPY IF THE SOURCE DATA SET IS A KSDS AND THE VOLUME BEING PROCESSED IS NOT THE PRIMARY VOLUME AND THE INDEX FOR THAT KSDS DOES NOT RESIDE ON THE CURRENT VOLUME BEING PROCESSED, THIS BIT MAY BE ON BECAUSE IT APPEARS TO BE A ESDS
		.1.. ..		E27SKSDS	1=KSDS DATA SET
		..1.		E27SKRDS	1=KRDS DATA SET
		...1		E27SLDS	1=LINEAR DATA SET
	 1...		E27SRRDS	1=RRDS DATA SET
	1..		E27SPSSI	1=PAGE/SWAP/STGINDEX, ETC. UNSUPPORTED DATA SETS
	1.		E27SVVDS	1=VVDS DATA SET
	1		E27SBCS	1=BCS DATA SET
94	(5E)	1... ..		E27SAIX	1=AIX DATA SET
		.1..		E27SVRRD	VRRDS DATASET
		..1.		E27BCSEL	1=BASE CLUSTER SELECTED
		...1		EI27RRB	RECOVERY REQUIRED
	 1111		*	UNUSED
95	(5F)	CHARACTER	5	EI27TFLG	TARGET DATA SET FLAGS
95	(5F)	CHARACTER	1	E27TSMF	TARGET SMS FLAGS
96	(60)	CHARACTER	2	E27TDSRG	TARGET DATA SET ORG
98	(62)	BIT(16)	2	E27TVFLG	TARGET VSAM DSET FLAGS
		1... ..		E27TESDS	1=ESDS DATA SET FOR PHYSICAL DATA SET COPY IF THE SOURCE DATA SET IS A KSDS AND THE VOLUME BEING PROCESSED IS NOT THE PRIMARY VOLUME AND THE INDEX FOR THAT KSDS DOES NOT RESIDE ON THE CURRENT VOLUME BEING PROCESSED, THIS BIT MAY BE ON BECAUSE IT APPEARS TO BE A ESDS
		.1..		E27TKSDS	1=KSDS DATA SET

		..1.		E27TKRDS	1=KRDS DATA SET
		...1		E27TLDS	1=LINEAR DATA SET
	 1...		E27TRRDS	1=RRDS DATA SET
	1..		E27TPSSI	1=PAGE/SWAP/STGINDEX, ETC. UNSUPPORTED DATA SETS
	1.		E27TVVDS	1=VVDS DATA SET
	1		E27TBCS	1=BCS DATA SET
99	(63)	1...		E27TAIX	1=AIX DATA SET
		.1..		E27TVRRD	VRRDS DATASET
		..1.		E27TRACF	1=DATA SET IS PROTECTED BY DISCRETE RACF PROFILE
		...1 1111		*	RESERVED
100	(64)	CHARACTER	8	EI27TRKS	TOTAL TRACKS COPIED
108	(6C)	CHARACTER	6	EI27SVOL	SOURCE VOLUME SERIAL
114	(72)	CHARACTER	6	EI27TVOL	TARGET VOLUME SERIAL
120	(78)	CHARACTER	1	E27FLGS	MISC FLAGS
		1...		EI27LVOL	LAST VOLUME INDICATOR
		.1..		E27F9ATT	FROM SRC DATA SET F9 ATTRS LOST
		..11 1111		*	AVAILABLE
121	(79)	CHARACTER	1	*	RESERVED
122	(7A)	SIGNED	2	EI27VLSQ	VOLUME SEQUENCE NUMBER FROM THE SOURCE DATA SET
124	(7C)	SIGNED	4	EI27DSHA	VSAM DATA COMPONENT ALLOCATED RBA COUNT
128	(80)	SIGNED	4	EI27ISHA	VSAM INDEX COMPONENT HIGH ALLOC RBA COUNT
132	(84)	SIGNED	2	EI27SCNT	VSAM DATA SET STRIPE COUNT
134	(86)	SIGNED	2	E27DSTP#	DATA COMPONENT STRIPE COUNT

THE EI27DRBA AND EI27IRBA FIELDS ARE CALCULATED BY
SUBTRACTING THE STARTING RBA FROM THE ENDING RBA

136	(88)	UNSIGNED	4	EI27DRBA	RBAS PROCESSED FOR THE DATA COMPONENT
140	(8C)	UNSIGNED	4	EI27IRBA	RBAS PROCESSED FOR THE INDEX COMPONENT
144	(90)	CHARACTER	44	EI27DANM	NEW DATA COMP NAME
188	(BC)	CHARACTER	44	EI27IXNM	NEW INDEX COMP NAME
232	(E8)	CHARACTER	32	*	RESERVED

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 28:
TARGET DATA SET ALLOCATION NOTIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	332	EIREC28	
0	(0)	CHARACTER	4	EI28PARMS1	EXIT 28 PARMS FIELD 1 ANY FIELD SET ON INDICATES THAT FIELD HAS BEEN SPECIFIED BE USED FOR THIS EXIT
		1...		EI28EATTRP	EATTR VALUE PASSED
		.1..		EI28F9DSCBP	F9 DSCB PASSED
		..1.		EI28CREDT	CREATION DATE PASSED
		...1		EI28RCAP	CA RECLAIM ATTR PASSED
	 1...		EI28ZFSP	LDS defined zFS passed
0	(0)	BIT(27) POS(6)	4	*	UNUSED
4	(4)	CHARACTER	4	EI28PARMS2	EXIT 28 PARMS FIELD 2
4	(4)	BIT(32)	4	*	UNUSED
8	(8)	CHARACTER	44	EI28TGDSN	TARGET DATA SET NAME
52	(34)	CHARACTER	44	EI28SRCDSN	SOURCE DATA SET NAME
96	(60)	CHARACTER	140	EI28F9DSCB	FORMAT 9 DSCB
236	(EC)	CHARACTER	1	EI28FIELDS1	EXIT 28 FIELDS 1 FLAG BYTE FOR BIT FIELDS
		11..		EI28EATTR	EATTR VALUE TO BE USED FOR TARGET DATA SET ALLOCATION
		..1.		EI28RCA	'00'B = NS (DEFAULT VALUE USED) '01'B = NO '10'B = OPT CA RECLAIM ATTRIBUTE 0 = CA RECLAIM NOT DISABLED 1 = CA RECLAIM DISABLED
		...1		EI28ZFS	LDS defined as zFS 0 = not defined as zFS 1 = LDS defined as zFS
	 1111		*	UNUSED
237	(ED)	CHARACTER	3	EI28CREDT	CREATION DATE(DS1CREDT)
240	(F0)	CHARACTER	92	*	SPACE FOR EXPANSION

PARAMETER BLOCK FOR UIM EXIT 30 NVA

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	12	EIREC30	
0	(0)	UNSIGNED	4	EI30TRKS	TRACKS REQUIRED
4	(4)	SIGNED	2	EI30RC	OVERALL RETURN CODE
6	(6)	CHARACTER	2	*	UNUSED
8	(8)	ADDRESS	4	EI30VSR@	PTR TO VOLUME STRUCTURE

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	260	EI30VLST	VOLUME LIST
0	(0)	SIGNED	2	EI30VOL#	NUMBER OF VOLUME ENTRIES
2	(2)	CHARACTER	2	*	UNUSED
4	(4)	CHARACTER	16	EI30VLENT(16)	VOLSER ENTRY ARRAY
4	(4)	CHARACTER	6	EI30VSR	VOLSER
10	(A)	CHARACTER	1	EI30VFLG	VOLUME FLAGS
		1... ..		EI30ATT	ALLOCATION ATTEMPTED
		.1.. ..		EI30USED	VOLUME USED
		..11 1111		*	UNUSED
11	(B)	CHARACTER	1	*	UNUSED
12	(C)	SIGNED	4	EI30VRC	VOLUME RETURN CODE
16	(10)	SIGNED	4	EI30VRSN	VOLUME REASON CODE

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 33:
OBJECT NAME NOTIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	184	EIREC33	
0	(0)	CHARACTER	154	EI33CNST	CONSTANT LENGTH PORTION OF CONTROL BLK
0	(0)	CHARACTER	2	EI33FLGS	OBJECT FLAGS
		1... ..		EI33META	ON=META OBJECT
		.1.. ..		EI33DATA	ON=USER DATA OBJECT
		..1.		EI33TCT_ENCRYPTED	ON=TCT ENCRYPTED
		...1		EI33TCT_COMPRESSED	ON=TCT COMPRESSED
0	(0)	BIT(14) POS(3)	2	*	UNUSED
2	(2)	CHARACTER	128	EI330BJN	OBJECT NAME
130	(82)	UNSIGNED	2	EI330BJL	OBJECT NAME LENGTH
132	(84)	UNSIGNED	2	*	UNUSED
134	(86)	CHARACTER	8	EI33TRKC	TOTAL TRACK STORED
142	(8E)	CHARACTER	8	EI33BYTES	TOTAL TKS/BYTES STORED
150	(96)	UNSIGNED	2	*	RESERVED
152	(98)	SIGNED	2	EI33RTC	RC FOR STORING OBJ
154	(9A)	CHARACTER	8	EI33BYTES_COMPRESSED	Compressed size-only set
when					EI33TCT_COMPRESSION=ON
162	(A2)	CHARACTER	22	*	RESERVED

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 35:
Cloud bypass verification

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	32	EIREC35	
0	(0)	CHARACTER	32	EI35CNST	CONSTANT LENGTH PORTION OF CONTROL BLK
0	(0)	CHARACTER	2	EI35FLG1	OPTION FLAGS
		1... ..		EI35BYPASS	ON=META OBJECT
				CREATE CONTAINER	
2	(2)	CHARACTER	30	*	RESERVED

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 41:
UNIX FILE PATH NOTIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	1027	EIREC41	

0	(0)	CHARACTER	1027	EI41CNST	CONSTANT LENGTH PORTION OF CONTROL BLK
0	(0)	UNSIGNED	4	EI41PATHL	LENGTH OF PATH NAME
4	(4)	CHARACTER	1023	EI41PATHN	PATH NAME

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 42:
UNIX FILE PATH BYPASS VERIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	1032	EIREC42	
0	(0)	CHARACTER	1032	EI42CNST	CONSTANT LENGTH PORTION OF CONTROL BLK
0	(0)	UNSIGNED	4	EI42PATHL	LENGTH OF PATH NAME
4	(4)	UNSIGNED	4	EI42MODE	FILE MODE (BPXYMODE)
8	(8)	UNSIGNED	1	EI42FLAG1	USER OPTIONS 1
		1... ..		EI42RSET	1=RESET LAST BACK UP
		.111 1111		*	AVAILABLE
9	(9)	CHARACTER	1023	EI42PATHN	PATH NAME

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 43:
UNIX FILE PATH PROCESSED NOTIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	1103	EIREC43	
0	(0)	CHARACTER	1103	EI43CNST	CONSTANT LENGTH PORTION OF CONTROL BLK
0	(0)	UNSIGNED	4	EI43PATHL	LENGTH OF PATH NAME
4	(4)	SIGNED	4	EI43RC	PATH RETURN CODE
8	(8)	CHARACTER	24	EI43STAT	STAT INFO
8	(8)	UNSIGNED	4	EI43MODE	FILE MODE (BPXYMODE)
12	(C)	UNSIGNED	4	EI43LINK	NUMBER OF LINKS
16	(10)	SIGNED	4	EI43UID	USER ID OF OWNER
20	(14)	SIGNED	4	EI43GID	GROUP ID OF GROUP
24	(18)	CHARACTER	8	EI43SIZE	FILE SIZE IN BYTES
32	(20)	CHARACTER	48	EI43STAT64	STAT 64BIT TIME FIELDS
32	(20)	CHARACTER	8	EI43ATIME64	LAST ACCESS TIME 64
40	(28)	CHARACTER	8	EI43MTIME64	LAST DATA-MOD TIME 64
48	(30)	CHARACTER	8	EI43CTIME64	LAST ATTR-MOD TIME 64
56	(38)	CHARACTER	8	EI43CREAT64	CREATION TIME 64
64	(40)	CHARACTER	8	EI43REF64	REFERENCE TIME 64
72	(48)	CHARACTER	8	*	STAT64 UNUSED
80	(50)	CHARACTER	1023	EI43PATHN	PATH NAME

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 44:
UNIX FILE CLONE INITIALIZATION NOTIFICATION EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	1035	EIREC44	
0	(0)	CHARACTER	1035	EI44CNST	CONSTANT LENGTH PORTION OF CONTROL BLK
0	(0)	UNSIGNED	4	EI44PATHL	LENGTH OF PATH NAME
4	(4)	UNSIGNED	4	EI44RETC	RETURN CODE
8	(8)	UNSIGNED	4	EI44RSNCD	REASON CODE
12	(C)	CHARACTER	1023	EI44PATHN	PATH NAME

THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 45:
UNIX FILE PRESENT FILE ATTRIBUTES EXIT

OFFSET DECIMAL	OFFSET HEX	TYPE	LENGTH	NAME (DIM)	DESCRIPTION
0	(0)	STRUCTURE	1035	EIREC45	
0	(0)	CHARACTER	1035	EI45CNST	CONSTANT LENGTH PORTION OF CONTROL BLK
0	(0)	UNSIGNED	4	EI45PATHL	LENGTH OF PATH NAME
4	(4)	UNSIGNED	4	EI45ATTRLEN	LENGTH OF BPXYATTR AREA
8	(8)	ADDRESS	4	EI45ATTR@	ADDRESS OF BPXYATTR
12	(C)	CHARACTER	1023	EI45PATHN	PATH NAME

Constants for ADREID0

The following shows the length, type, value, name and description for each constant:

```

4 CHARACTER EIDB ADREIBID BLOCK IDENTIFIER
2 DECIMAL 0 EIRC00 CONTINUE NORMAL PROCESS
2 DECIMAL 4 EIRC04 RECORD REPLACED
2 DECIMAL 8 EIRC08 INSERT RECORD
2 DECIMAL 12 EIRC12 DELETE RECORD
2 DECIMAL 16 EIRC16 RECORD MODIFIED
2 DECIMAL 20 EIRC20 DISCONNECT EXIT
2 DECIMAL 24 EIRC24 SELECT USER STATS RECS
2 DECIMAL 28 EIRC28 WTOR RESPONSE
2 DECIMAL 32 EIRC32 TERMINATE FUNCTION
2 DECIMAL 36 EIRC36 TERMINATE DSET ONLY
2 DECIMAL 0 EIOP00 FUNCTION STARTUP ENTRY
2 DECIMAL 1 EIOP01 READING SYSIN RECORD
2 DECIMAL 2 EIOP02 PRINTING SYSPRINT RECORD
2 DECIMAL 3 EIOP03 READING PHYSICAL TAPE
2 DECIMAL 4 EIOP04 READING LOGICAL TAPE
2 DECIMAL 5 EIOP05 WRITING LOGICAL TAPE
2 DECIMAL 6 EIOP06 WRITING PHYSICAL TAPE
2 DECIMAL 7 EIOP07 READING DISK TRACK
2 DECIMAL 8 EIOP08 WRITING DISK TRACK
2 DECIMAL 9 EIOP09 READING UTILITY SYSPRINT
2 DECIMAL 10 EIOP10 WRITING UTILITY SYSPRINT
2 DECIMAL 11 EIOP11 WRITING WTO MESSAGE
2 DECIMAL 12 EIOP12 WRITING WTOR MESSAGE
2 DECIMAL 13 EIOP13 PRESENTING ADRUFO REC
2 DECIMAL 14 EIOP14 FUNCTION TERMINATION
2 DECIMAL 15 EIOP15 PRESENTING WTOR RESPONSE
2 DECIMAL 16 EIOP16 TAPE VOL SECURITY
2 DECIMAL 17 EIOP17 TAPE MOUNT(NON-SPEC)
2 DECIMAL 18 EIOP18 INSERT LOGICAL REC
2 DECIMAL 19 EIOP19 TAPE OUTPUT ERROR
2 DECIMAL 20 EIOP20 VOLUME NOTIFICATION
2 DECIMAL 21 EIOP21 DSET VERIFICATION
2 DECIMAL 22 EIOP22 BYPASS VERIFICATION
2 DECIMAL 23 EIOP23 DS PROC NOTIFICATION
2 DECIMAL 708 VOLSLST MAX SIZE OF VOLSER LIST FOR EIREC23 (118
* 6)
2 DECIMAL 24 EIOP24 CC INIT DONE
2 DECIMAL 25 EIOP25 BACKSPACE TAPEIN
2 DECIMAL 26 EIOP26 VOLUME OPEN FOR O/P
2 DECIMAL 27 EIOP27 PHYSICAL DS PROCESSED
2 DECIMAL 28 EIOP28 TARGET ALLOCATION EXIT
2 DECIMAL 29 EIOP29 RESERVED
2 DECIMAL 30 EIOP30 RESERVED
2 DECIMAL 31 EIOP31 APPLICATION STORE META DATA
2 DECIMAL 32 EIOP32 APPLICATION RETRIEVE META DATA
2 DECIMAL 33 EIOP33 OBJECT NAME NOTIFY
2 DECIMAL 33 EIMAXOP MAXIMUM VALID EXIT NUMBER
2 DECIMAL 35 EIOP35 CLOUD BYPASS VERIFICATION
2 DECIMAL 45 EIMAXOP MAX VAL EXIT NUMBER

2 DECIMAL 48 EXITRECL EXIT RECORD LEN

CONSTANTS FOR SERIALIZATION ERROR ACTIONS
=====
1 DECIMAL 0 EISRLNONE USE NONE
1 DECIMAL 1 EISRLDB2 USE DB2
1 DECIMAL 2 EISRLZFS USE ZFS
1 DECIMAL 3 EISRLCICS USE CICS
1 DECIMAL 4 EISRLEXIT USE USER EXIT

4 DECIMAL 42 EIDBLEN LENGTH OF BLOCK
2 BIT 00 EI28EATTR_NS EATTR NOT SPECIFIED CONSTANT
2 BIT 01 EI28EATTR_NO EATTR NO CONSTANT
2 BIT 10 EI28EATTR_OPT EATTR NO CONSTANT

```

Cross reference for ADREID0

1 CROSS REFERENCE

NAME	HEX OFFSET	HEX VALUE	LEVEL
=====	=====	=====	=====
ADREIB	0		1

	EIDDDID	22		2
	EIDDDINFO	0		1
	EIDDDNAME	0		2
	EIDLLEN	0		2
	EIID	2		2
	EIOPTION	E		2
	EIRECALN	12		2
	EIRECLEN	16		2
	EIRECPTR	1A		2
	EIREC00	0		1
	EIREC14	0		1
	EIREC20	0		1
	EIREC21	0		1
	EIREC22	0		1
	EIREC23	0		1
	EIREC24	0		1
	EIREC26	0		1
	EIREC27	0		1
	EIREC28	0		1
	EIREC30	0		1
	EIREC33	0		1
	EIREC35	0		1
	EIREC41	0		1
	EIREC42	0		1
	EIREC43	0		1
	EIREC44	0		1
	EIREC45	0		1
	EIRETC	E		2
	EIRETCOD	10		2
	EISGCNT	0		3
	EISGENT	4		2
	EISGHDR	0		2
	EISGLEN	4		3
	EISGLIST	0		1
	EISGNAM	6		3
	EITSKID	6		2
	EIUSEPTR	1E		2
	EIVOLID	8		2
	EIXABEND	26	80	4
	EIXALLOW	A		2
	EIXALOW0	A		3
	EIXALOW1	B		3
	EIXALOW2	C		3
	EIXALOW3	D		3
	EIXDASYS	27	04	4
	EIXDEL	A	20	4
	EIXDIS	A	08	4
	EIXERR	D	80	4
	EIXFLAGS	26		2
	EIXFLAG0	26		3
	EIXFLAG1	27		3
	EIXFLAG2	28		3
	EIXFLAG3	29		3
	EIXINS	A	40	4
	EIXMOD	A	10	4
1	CROSS REFERENCE			
	NAME	HEX OFFSET	HEX VALUE	LEVEL
	=====	=====	=====	=====
	EIXMWPL	22		3
	EIXNTERR	26	40	4
	EIXREP	A	80	4
	EIXSTAT	A	02	4
	EIXTDSET	B	80	4
	EIXTDUMS	27	20	4
	EIXTDUMT	27	10	4
	EIXTERM	A	01	4
	EIXTISXM	27	08	4
	EIXTRC32	27	80	4
	EIXTRERR	27	01	4
	EIXTRKER	26	10	4
	EIXTWERR	27	02	4
	EIXWNGOK	26	20	4
	EIXWTOR	A	04	4
	EI00_NOEXCL_NSPLOCK	1	01	3
	EI00BSEC	1	08	3
	EI00BSER	1	02	3
	EI00CANSFE	8	40	3
	EI00CIRBA_DA	C		2
	EI00CIRBA_IX	10		2
	EI00FLGS	1		2

	EI00FLG2	8		2
	EI00MIXDEV	8	20	3
	EI00MVOLRECOV	1	04	3
	EI00NENQ	1	10	3
	EI00NOLK	1	20	3
	EI00NONF	1	40	3
	EI00SBPL	0		2
	EI00SENO	1	80	3
	EI00SVOL	2		2
	EI00SWNCRYPT	8	80	3
	EI00ZCOMP	8	10	3
	EI06DCB@	10		2
	EI06FLGS	F		2
	EI06FTLR	F	80	3
	EI06THWE	F	40	3
	EI14CPUT	8		2
	EI14MESS	4		2
	EI14MNUM	4		3
	EI14MTYP	7		3
	EI14RC	0		2
	EI14TCT_CSAVE	10		3
	EI14ZCSV	10		2
	EI20CON	0		2
	EI20DA#	2E		2
	EI20DA@	30		2
	EI20DRBA	3C		3
	EI20DSHA	3C		2
	EI20DSI	0		1
	EI20DSN	0		2
	EI20FLGS	2C		2
	EI20ISHA	40		2
	EI20IX#	2F		2
	EI20IX@	34		2
	EI20LVOL	2C	20	3
1	CROSS REFERENCE			
	NAME	HEX OFFSET	HEX VALUE	LEVEL
	=====	=====	=====	=====
	EI20NVOL	2C		2
	EI20RACF	2C	40	3
	EI20RBA	38		3
	EI20SCNT	3A		2
	EI20VLI	30		2
	EI20VLSQ	40		3
	EI20VOL	30		3
	EI20VSAM	2C	80	3
	EI21DSN	0		2
	EI22_BYPASS_SOURCE_SER	32	01	3
	EI22ACSEN	50		2
	EI22BMIG	2C	20	3
	EI22BRBK	4E	01	3
	EI22BRLS	4E	08	3
	EI22BSEC	2C	40	3
	EI22BSER	2C	80	3
	EI22BWOA	4E	F0	3
	EI22BWOE	32	02	3
	EI22BWOP	32	08	3
	EI22DB2	2C	01	3
	EI22DSN	0		2
	EI22DSSRL	4F		2
	EI22EXTR	2C	04	3
	EI22FLGS	2C		2
	EI22FLG2	32		2
	EI22FLG3	4E		2
	EI22IMS	32	04	3
	EI22LIKE	2C	08	3
	EI22LINF	32	10	3
	EI22LPRM	33		2
	EI22LREP	4E	02	3
	EI22LSID	34		2
	EI22NSYS	2C	10	3
	EI22RRB	32	20	3
	EI22RSET	32	80	3
	EI22SFLG	2D		2
	EI22SFSM	4E	04	3
	EI22SGP	58		2
	EI22SIRS	2C	02	3
	EI22SSYS	32	40	3
	EI23CNST	0		2
	EI23DNAM	7D		3
	EI23DSN	0		3

	EI23DSRC	58		3
	EI23GMT	64		4
	EI23LOC	6C		4
	EI23NEWN	2C		3
	EI23RLST	64		3
	EI23RRB	5E	10	5
	EI23SFLG	5A		3
	EI23TFLG	5F		3
	EI23VOL#	8C		3
	EI23VSER	8D		2
	EI24DSN	E		2
	EI24FLGS	3A		2
	EI24RSCD	2		2
1	CROSS REFERENCE			
		HEX OFFSET	HEX VALUE	LEVEL
	=====	=====	=====	=====
	EI24RSET	3A	80	3
	EI24RTCD	0		2
	EI24VOL	8		2
	EI26DDN	48		2
	EI26DSN	4		2
	EI26DSNL	44		2
	EI26R0CE	0	20	3
	EI26TERM	0	40	3
	EI26THWE	0	08	3
	EI26TYPE	0		2
	EI26VCLO	0	10	3
	EI26VOL	0	80	3
	EI26VSER	50		2
	EI26VTRC	58		2
	EI27CNST	0		2
	EI27DANM	90		3
	EI27DRBA	88		3
	EI27DSHA	7C		3
	EI27DSN	0		3
	EI27DSRC	58		3
	EI27IRBA	8C		3
	EI27ISHA	80		3
	EI27IXNM	BC		3
	EI27LVOL	78	80	4
	EI27NEWN	2C		3
	EI27RRB	5E	10	5
	EI27SCNT	84		3
	EI27SFLG	5A		3
	EI27SVOL	6C		3
	EI27TFLG	5F		3
	EI27TRKS	64		3
	EI27TVOL	72		3
	EI27VLSQ	7A		3
	EI28CREDT	ED		2
	EI28CREDTP	0	20	3
	EI28EATTR	EC	C0	3
	EI28EATTRP	0	80	3
	EI28FIELDS1	EC		2
	EI28F9DSCB	60		2
	EI28F9DSCBP	0	40	3
	EI28PARMS1	0		2
	EI28PARMS2	4		2
	EI28RCA	EC	20	3
	EI28RCAP	0	10	3
	EI28SRCDSN	34		2
	EI28TGTDN	8		2
	EI28ZFS	EC	10	3
	EI28ZFSP	0	08	3
	EI30ATT	A	80	4
	EI30RC	4		2
	EI30TRKS	0		2
	EI30USED	A	40	4
	EI30VFLG	A		3
	EI30VLIST	0		1
	EI30VOL#	0		2
1	CROSS REFERENCE			
		HEX OFFSET	HEX VALUE	LEVEL
	=====	=====	=====	=====
	EI30VOLENT	4		2
	EI30VRC	C		3
	EI30VRSN	10		3
	EI30VSER	4		3

EI30VSER@	8		2
EI33BYTES	8E		3
EI33BYTES_COMPRESSED	9A		3
EI33CNST	0		2
EI33DATA	0	40	4
EI33FLGS	0		3
EI33META	0	80	4
EI33OBJL	82		3
EI33OBJN	2		3
EI33RTC	98		3
EI33TRKC	86		3
EI41CNST	0		2
EI41PATHL	0		3
EI41PATHN	4		3
EI42CNST	0		2
EI42FLAG1	8		3
EI42MODE	4		3
EI42PATHL	0		3
EI42PATHN	9		3
EI42RSET	8	80	4
EI43ATIME64	20		4
EI43CNST	0		2
EI43CREAT64	38		4
EI43CTIME64	30		4
EI43GID	14		4
EI43LINK	C		4
EI43MODE	8		4
EI43MTIME64	28		4
EI43PATHL	0		3
EI43PATHN	50		3
EI43RC	4		3
EI43REF64	40		4
EI43SIZE	18		4
EI43STAT	8		3
EI43STAT64	20		3
EI43UID	10		4
EI44CNST	0		2
EI44PATHL	0		3
EI44PATHN	C		3
EI44RETC	4		3
EI44RSNCD	8		3
EI45ATTR@	8		3
EI45ATTRLEN	4		3
EI45CNST	0		2
EI45PATHL	0		3
EI45PATHN	C		3
E20DSTP#	36		3
E20F9ATT	2C	10	3
E22SAIX	31	80	4
E22SBCS	30	01	4
E22SDSRG	2E		3
E22SESDS	30	80	4
1 CROSS REFERENCE			

NAME	HEX OFFSET	HEX VALUE	LEVEL
=====	=====	=====	=====
E22SKRDS	30	20	4
E22SKSDS	30	40	4
E22SLDS	30	10	4
E22SPSSI	30	04	4
E22SRRDS	30	08	4
E22SSMSF	2D		3
E22SVFLG	30		3
E22SVRRD	31	40	4
E22SVVDS	30	02	4
E23BCSEL	5E	40	5
E23BPDS	7C	40	4
E23BPSE	7C	20	4
E23BSET	7C	80	4
E23BYTES	74		3
E23FLGS	7C		3
E23F9ATT	7C	10	4
E23SAIX	5E	80	5
E23SBCS	5D	01	5
E23SDSRG	5B		4
E23SESDS	5D	80	5
E23SKRDS	5D	20	5
E23SKSDS	5D	40	5
E23SLDS	5D	10	5
E23SPSSI	5D	04	5
E23SRRDS	5D	08	5

	E23SSMSF	5A		4
	E23SVFLG	5D		4
	E23SVRRD	5E	20	5
	E23SVVDS	5D	02	5
	E23TAIX	63	80	5
	E23TBCS	62	01	5
	E23TDSRG	60		4
	E23TESDS	62	80	5
	E23TKRDS	62	20	5
	E23TKSDS	62	40	5
	E23TLDS	62	10	5
	E23TPSSI	62	04	5
	E23TRRDS	62	08	5
	E23TSMSF	5F		4
	E23TVFLG	62		4
	E23TVRRD	63	40	5
	E23TVVDS	62	02	5
	E24BCSEL	3A	20	3
	E24SAIX	3A	40	3
	E27BCSEL	5E	20	5
	E27DSTP#	86		3
	E27FLGS	78		3
	E27F9ATT	78	40	4
	E27SAIX	5E	80	5
	E27SBCS	5D	01	5
	E27SDSRG	5B		4
	E27SESDS	5D	80	5
	E27SKRDS	5D	20	5
	E27SKSDS	5D	40	5
	E27SLDS	5D	10	5
1	CROSS REFERENCE			
	NAME	HEX OFFSET	HEX VALUE	LEVEL
	=====	=====	=====	=====
	E27SPSSI	5D	04	5
	E27SRRDS	5D	08	5
	E27SSMSF	5A		4
	E27SVFLG	5D		4
	E27SVRRD	5E	40	5
	E27SVVDS	5D	02	5
	E27TAIX	63	80	5
	E27TBCS	62	01	5
	E27TDSRG	60		4
	E27TESDS	62	80	5
	E27TKRDS	62	20	5
	E27TKSDS	62	40	5
	E27TLDS	62	10	5
	E27TPSSI	62	04	5
	E27TRACF	63	20	5
	E27TRRDS	62	08	5
	E27TSMSF	5F		4
	E27TVFLG	62		4
	E27TVRRD	63	40	5
	E27TVVDS	62	02	5

Example: invoking DFSMSdss by using an application program

The following example shows that a full DASD volume is to be dumped to a tape volume or volumes. DFSMSdss is LINKed somewhere in MYJOB to perform the dump, and, conditionally, the DEFRAG functions.

```
//JOB1      JOB      accounting information,REGION=nnnnK
//STEP1     EXEC     PGM=MYJOB
//STEPLIB   DD       DSN=MY.LINKLIB,DISP=SHR
//SYSPRINT  DD       SYSOUT=A
//DASD      DD       UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//TAPE      DD       UNIT=3480,VOL=SER=(TAPE01,TAPE02),
//          LABEL=(1,NL),DISP=(NEW,KEEP)
//SYSIN     DD       *
              DUMP  INDD(DASD) OUTDD(TAPE)
              IF LASTCC = 0 -
                THEN DEFR DDN(DASD)
/*
```

The preceding example does not show how the invocation of DFSMSdss was brought about, but does show that the user program, MYJOB, was run. At some point MYJOB needs to run DFSMSdss to perform the functions specified in the SYSIN data set. The next example shows the code needed at that point to LINK to DFSMSdss. Because no EXEC PARMs were specified and the standard SYSIN and SYSPRINT data set names are to be used, there is no need to pass special parameters.

```

LINK EP=ADDRSSU,PARAM=(OPTPTR),VL=1
      .
      .
      .
      .
OPTPTR  CNOP 2,4
        DC   H(0)

```

Related reading: For additional information about the Application Interface, see Chapter 24, “Examples of the application program with the user interaction module (UIM),” on page 627.

How to determine DFSSdss version, release, and modification level

Subsystems that invoke DFSMSdss dynamically must determine if DFSMSdss is installed on the system, and if it is, its version, release, and modification level, and features supported. A DFSMSdss-provided macro tries to determine the DFSMSdss version, release, and modification level and features supported and pass the requested information in a register.

ADRMCLVL (in SYS1.MACLIB) is an inline executable assembler-language macro that can be invoked by a caller. The caller can be in problem program state and can have a user key. The caller must save registers 0, 1, 14, and 15 before invoking the macro. No other registers are disturbed. The caller can determine the installed level and features of DFSMSdds from the information that is returned in registers 1 and 14.

On return, register 1 contains information as follows:

- If the release level of ADRDSSU cannot be determined, register 1 contains X'04000000'.
- Otherwise, register 1 contains:

Byte 0

Product number, in binary:

- 0 = DFDSS
2 = MVS or OS/390® DFSMSdss
3 = z/OS DFSMSdss

Byte 1

Version number, in binary:

- 1 = Version 1
2 = Version 2

Byte 2

When byte 0 is 0 or 2:

- Release number, in binary:

- 1 = Release 1
2 = Release 2
3 = Release 3
4 = Release 4
5 = Release 5
A = Release 10

When byte 0 is 3:

- Release number, in decimal:

01 = Release 1
02 = Release 2
03 = Release 3
04 = Release 4
05 = Release 5
10 = Release 10
11 = Release 11
12 = Release 12

Byte 3

Modification level, in binary:

0 = Modification level 0
1 = Modification level 1
2 = Modification level 2

On return, register 14 contains the following information:

- If the release level of ADRDSSU is less than DFSMSdss Version 1, Release 4, Modification level 0, then the contents of register 14 are unpredictable.
- Otherwise, register 14 contains the following:

Byte 0

Feature Flags:

Bit 0, when set to 1, means DFSMSdss cross-memory Application Programming Interface support for concurrent copy is available.

Bit 1, when set to 1, means the PTF that introduced the ZCOMPRESS keyword support (OA42238) is applied.

Bit 2, when set to 1, means the software support that introduced the Bypass verification exit (Eioption 22), Bypass Source Serialization (EI22_BYPASS_SOURCE_SER) during logical copy is applied (OA49854).

Bit 3, when set to 1, means DFSMSdss has the software support for the FCTOXRCPPRIMARY function (APAR OA44701).

Bit 4, when set to 1, means the PTF that introduced transparent cloud tiering cloud/archive (OA48365) is applied.

Bit 5, when set to 1, means that the PTF that introduced full volume support for transparent cloud tiering (OA57526) is applied.

Bits 6–7 are reserved.

Bytes 1–3

Reserved.

Chapter 24. Examples of the application program with the user interaction module (UIM)

This topic contains General-use Programming Interface and Associated Guidance information, and Product sensitive Programming Interface information.

Figure 29 on page 627 shows the process by which DFSMSdss can call user interaction module (UIM) functions.

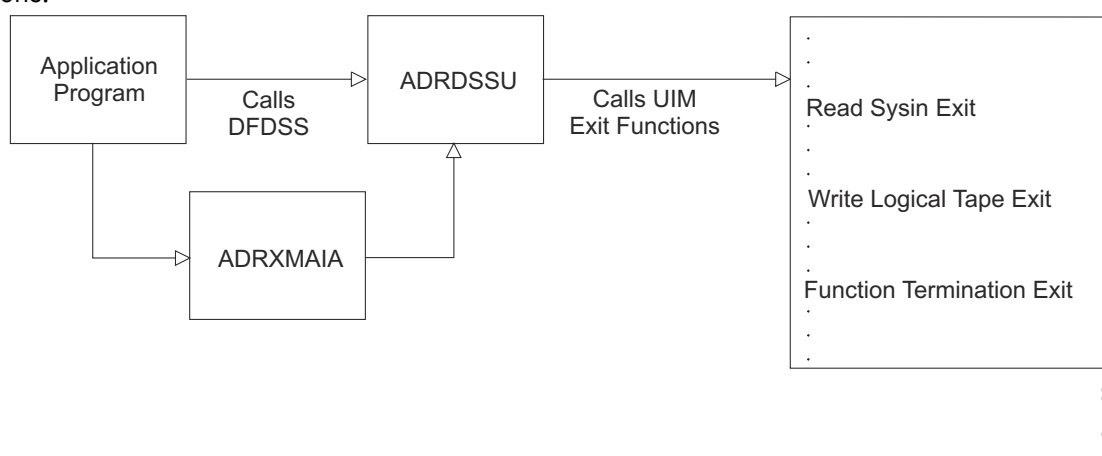


Figure 29. The application program process

This topic contains the following examples:

- The JCL to invoke an application program that invokes DFSMSdss
- A complete sample program listing showing how a user can use all of the UIM exit functions to receive control from DFSMSdss ([“Application interface program example” on page 628](#) and [“User interaction module example” on page 629](#))
- An output listing ([“Output resulting from use of the UIM exits” on page 639](#)) resulting from the sample program in [“Application interface program example” on page 628](#) and [“User interaction module example” on page 629](#)

Note: The example shown is not written in reentrant code. If you are planning to share a UIM between tasks, you should code the module in reentrant code.

The example has not been submitted to any formal test and is distributed as it is, without any warranty either expressed or implied. The use of this example or the implementation of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer’s operational environment. Although each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

The following JCL was used to invoke an application program, which then called DFSMSdss.

```

//STEPS014 EXEC PGM=USRAIPGM
//STEPLIB DD DISP=SHR,DSN=SYS1.LINKLIB
//SYSPRINT DD SYSOUT=*
//DASD DD UNIT=SYSDA,VOL=SER=D9S060,DISP=SHR
//TAPE DD DD DISP=(,CATLG),DSN=PUBSEXMP.DUMP,
// UNIT=3390,VOL=SER=DAVIS4,
// SPACE=(CYL,(10,10),RLSE)
//MYDATA DD *
EOJ
/*
  
```

Application interface program example

```

*****
*
* Module Name           = USRAIPGM
*
* Descriptive Name = DFSMSdss Application Interface Program Example
*
* Function             = Invoke DFSMSdss supplying alternate UIM: USRUIM
*                       or
*                       = Invoke DFSMSdss Cross Memory Application
*                       Interface UIM: USRUIM
*
* Operation            = Place SYSIN records in the user area.  The UIM
*                       will pass these records to DFSMSdss.
*
*****
USRAIPGM CSECT
USRAIPGM AMODE 31
USRAIPGM RMODE ANY
*****
* Standard entry linkage
*****
      STM 14,12,12(13)
      BALR 12,0
      USING *,12
      LA 3,SAVEAREA
      ST 3,8(13)
      ST 13,4(3)
      LR 13,3
*****
* Load and call DFSMSdss
*****
      LOAD EP=ADRDSSU
+      CNOP 0,4 @YA29363 01-LOAD
+      LA 0,++8 LOAD PARAMETER INTO REGISTER ZERO 01-LOAD
+      B ++12 BRANCH AROUND CONSTANT(S) 01-LOAD
+      DC CL8'ADRDSSU' ENTRY POINT NAME 01-LOAD
+      SR 1,1 SHOW NO DCB PRESENT 01-LOAD
+      SVC 8 01-LOAD
      LR 15,0
      LA 3,USERAREA
      ST 3,UA@
      CALL (15),(OPTPTR,DDNPTR,PAGEPTR,UIMPTR,UAPTR),VL
+      DS 0H 01-CALL
+      CNOP 0,4 02-IHBOP
+      LA 1,IHB0003 LIST ADDRESS @L1C 02-IHBOP
+      B IHB0003A BYPASS LIST @ZMC3742 02-IHBOP
+IHB0003 EQU * 02-IHBOP
+      DC A(OPTPTR) PROB.PROG.PARAMETER 02-IHBOP
+      DC A(DDNPTR) PROB.PROG.PARAMETER 02-IHBOP
+      DC A(PAGEPTR) PROB.PROG.PARAMETER 02-IHBOP
+      DC A(UIMPTR) PROB.PROG.PARAMETER 02-IHBOP
+      DC A(UAPTR+'X'80000000') @G860P40 02-IHBOP
+IHB0003A EQU * 02-IHBOP
+      BALR 14,15 BRANCH TO ENTRY POINT 01-CALL
*****
* Load and call DFSMSdss Cross Memory Application Interface
*****
      LOAD EP=ADRXMAIA
      LR 15,0
      LA 3,USERAREA
      ST 3,UA@
      CALL (15),(OPTPTR,DDNPTR,PAGEPTR,UIMPTR,UAPTR,ASNPT),VL
*****
* Standard exit linkage
*****
      L 13,4(13)
      RETURN (14,12),,RC=(15)
+      L 14,12(0,13) RESTORE REG 14 @L1C 01-RETUR
+      LM 0,12,20(13) RESTORE THE REGISTERS 01-RETUR
+      BR 14 RETURN 01-RETUR
*****
* Option area (same format as EXEC parameters)
*****
      CNOP 2,4
      OPTPTR DC AL2(OPTLEN)
      OPTIONS DC C'SIZE=4096K,TRACE=YES'
      OPTLEN EQU *-OPTIONS
*****
* DDNAME area (SYSIN is replaced by MYDATA)
*

```



```

*****
      CNOP  2,4
DDNPTR  DC   AL2(DDNLEN)
DDNAMES DC   XL8'00'
      DC   XL8'00'
      DC   XL8'00'
      DC   XL8'00'
      DC   CL8'MYDATA'
      DC   CL8'SYSPRINT'
DDNLEN  EQU   *-DDNAMES
*****
* Page number area (first page will be 1) *
*****
      CNOP  2,4
PAGEPTR DC   AL2(PAGELEN)
PAGENO  DC   CL4'0001'
PAGELEN EQU   *-PAGENO
*****
* User Interaction Module area (UIM is called USRUIM) *
*****
      CNOP  2,4
UIMPTR  DC   AL2(UIMLEN)
UIM      DC   CL8'USRUIM'
UIMLEN  EQU   *-UIM
*****
* User area pointer area *
*****
      CNOP  2,4
UAPTR   DC   AL2(UALEN)
UA@     DS    A
UALEN   EQU   *-UA@
*****
* User area *
*****
USERAREA DC   AL1(COM1)           Length of first command
      DC   C' WTO 'USRAIPGM INVOKING DFSMSdss''
COM1     EQU   *-USERAREA-1
SECNDCOM DC   AL1(COM2)           Length of first command
      DC   C' DUMP DS(EXC(SYS1.**)) LOGINDD(DASD) OUTDD(TAPE)'
COM2     EQU   *-SECNDCOM-1
THIRDCOM DC   AL1(COM3)           Length of third command
      DC   C' IF LASTCC=0 - '
COM3     EQU   *-THIRDCOM-1
FOURCOM  DC   AL1(COM4)           Length of fourth command
      DC   C' THEN DEFRAG DDN(DASD)'
COM4     EQU   *-FOURCOM-1
ENDCOMM  DC   X'00'              End of command flag
*****
* Register save area *
*****
SAVEAREA DS    18F
*
      END

```

User interaction module example

```

*****
*                                                                 *
* Module Name           = USRUIM                                *
*                                                                 *
* Descriptive Name      = DFSMSdss User Interaction Module Example *
*                                                                 *
* Function              = Perform various functions at different exit *
*                        points.                                   *
*                                                                 *
* Exit Point           Operation                                  *
* -----             - - - - - - - - - - - - - - - - - - - *
* 0                   Initialize counters for task               *
* 1                   Insert SYSIN records from user area         *
* 2                   Insert table of accounting data             *
* 3                   Count Tape Blocks Read                     *
* 4                   Count Logical Tape Blocks Read             *
* 5                   Count Logical Tape Blocks Written           *
* 6                   Count Tape Blocks Written                 *
* 7                   Count DASD Tracks Read                     *
*                                                                 *

```

```

*          8          Count DASD Tracks Written          *
*          9          Nothing                            *
*         10          Nothing                            *
*         11          Insert a WTO message               *
*         12          Never allow ADR369D                *
*         13          Force Reblocking                   *
*         14          Convert counts to printable characters *
*         15          Nothing                            *
*         16          Always bypass password protection for tape *
*         17          Supply TAPE01 as a tape volser      *
*         18          Nothing                            *
*         19          Save Tape DDNAME and volser which had error *
*         20          Save Data Set name                  *
*         21          Do not process temporary data sets  *
*         22          Bypass serialization of SYS1 data sets *
*         23          End function if processing errors    *
*         24          End function if initialization fails  *
*
*         25          Nothing                            *
*         26          Nothing                            *
*****
*
USRUIM  CSECT
USRUIM  AMODE 31
USRUIM  RMODE ANY
*****
* Registers with special uses          *
*****
*
EIDBASE EQU 2          Base reg for ADREIB.
OPTION  EQU 3          Reg to test option
WORKREG EQU 4          Work register
LENGTH  EQU 5          Reg to get length of records
TEMPBASE EQU 6         Temporary base reg for exits
RC       EQU 7          Register for return code
SYSPBASE EQU 8         Register for sysprint recs
MSGOFF   EQU 9         Register for offsets to table
*****
* Save Registers and Establish Addressability.          *
*****
        USING *,15          Initial Addressability
        STM 14,12,12(13)    Save regs
        BALR 12,0           New base addressability
        USING *,12
        DROP 15
        B BEGIN
        DC CL16'CSECT - USRUIM'
*****
* Establish Addressability to EIDB          *
*****
BEGIN   LR 15,13          Save caller SA pointer
        LA 13,SAVEAREA
        ST 15,SAVEAREA+4  Backward Chain
        B BEGIN
        ST 13,8(15)       Forward Chain
        USING ADREIB,EIDBASE Addressability to EIDB
        L EIDBASE,0(,1)   Address of EIDB
        LH OPTION,EIOPTION Get DFSMSdss processing option
        SLL OPTION,2      Multiply times four
        LA 15,VECTABLE    Point to vector table start
        L 15,0(OPTION,15) Point to processor routine
        BALR 14,15        Go to processor routine
        L 13,SAVEAREA+4   Restore SA pointer
        RETURN (14,12)    Return to DFSMSdss
+       LM 14,12,12(13)   RESTORE THE REGISTERS 01-RETURN
+       BR 14            RETURN 01-RETURN
*****
* Processing Routines          *
*****
*****
* AIOPT00: Function Startup          *
* Initialize counters for tape and dasd accounting.          *

```

```

*   Insert message identifying task into table   *
*****
AIOPT00  XC   RTBCNT,RTBCNT           Init Tape Blocks Read
          XC   WTBCNT,WTBCNT           Init Tape Blocks Written
          XC   RLTBCNT,RLTBCNT         Init Log Tape Blocks Read
          XC   WLTBCNT,WLTBCNT         Init Log Tape Blocks Written
          XC   RDTCNT,RDTCNT           Init Disk Tracks Read
          XC   WDTCNT,WDTCNT           Init Disk Tracks Written
          ICM   WORKREG,15,EITSKID      Get current task identifier
          CVD   WORKREG,CVDWORK        Convert to decimal
          UNPK  OPT00TSK(2),CVDWORK+6(2) Unpack into message insert
          OI    OPT00TSK+1,X'F0'       Make last character printable
          L     MSGOFF,MSGCOUNT        Get current MSGCOUNT
          MH     MSGOFF,MSGLEN          Multiply by MSGLEN to get
          LA     WORKREG,MSGTABLE       offset into message table
          LA     TEMPBASE,0(MSGOFF,WORKREG)
          USING TABLEMAP,TEMPBASE
          MVC    TABENTRY(133),OPT00MSG Move into msgtable
          DROP   TEMPBASE
          L      WORKREG,MSGCOUNT      Increment MSGCOUNT
          LA     WORKREG,1(WORKREG)
          ST     WORKREG,MSGCOUNT      Save new count
          LH     RC,EIRC00              Set normal process retcode
          STH    RC,EIRETCOD           Store retcode in EIDB
          BR     14                    Return to intercept processor
*****
* AIOPT01: Read SYSIN Record *
*   Get SYSIN records from the user area and give them to DFSMSdss *
*   using the Insert Record return code. *
*****
AIOPT01  TM     SYSINFL,SYSIFRST       Q. First entry to SYSIN exit?
          BO     NXTSYS                A. No, get next input
          OI     SYSINFL,SYSIFRST      Indicate first time taken
          ICM    TEMPBASE,15,EIUSEPTR  Get address of UA
          ST     TEMPBASE,CURCOMM      Save pointer to first record
          L      TEMPBASE,CURCOMM      Get current command pointer
          LA     WORKREG,1(,TEMPBASE)  Get past length
          STCM   WORKREG,15,EIRECPTR   Initialize the pointer
          SR     LENGTH,LENGTH         Clear workreg
          IC     LENGTH,0(,TEMPBASE)    Get length of record
          LTR    LENGTH,LENGTH         Q. End of SYSIN data?
          BZ     SYSDEL                A. Yes, bypass SYSIN DS
          STCM   LENGTH,15,EIRECLEN    Set record length
          LA     TEMPBASE,1(LENGTH,TEMPBASE) Point to new record
          ST     TEMPBASE,CURCOMM      Save new pointer
          LH     RC,EIRC08              Set insert record retcode
          B      SYSEXIT               Continue setup
          ICM    LENGTH,15,EIRECLEN    Get current record length
          LTR    LENGTH,LENGTH         Q. EOF condition ? (reclen=0)
          BZ     SYSRC00               A. Yes, process it.
          LH     RC,EIRC12              Set delete record retcode
          B      SYSEXIT               Continue Setup
          SYSDEL ICM    LENGTH,15,EIRECLEN
          LTR    LENGTH,LENGTH
          BZ     SYSRC00
          LH     RC,EIRC00
          SYSEXIT STH    RC,EIRETCOD    Store retcode in EIDB
          BR     14                    Return to intercept processor
*****
* AIOPT02: Write SYSPRINT *
*   Insert SYSPRINT records before message ADR012I *
*****
AIOPT02  ICM    SYSPBASE,15,EIRECPTR
          USING  SYSPMAP,SYSPBASE
          CLC    MSGID(8),ADR012I      Q. Last SYSPRINT rcd?
          BNE    NOTLAST               A. No, don't insert records
          TM     SYSPRFL,INSDONE        Q. Are we done inserting?
          BO     NOTLAST               A. Yes, don't insert records
          L      WORKREG,INSERTCT
          LR     MSGOFF,WORKREG         Save INSERTCT for offset
          LA     WORKREG,1(WORKREG)     Increment INSERTCT and
          C      WORKREG,MSGCOUNT      Q. Have we printed all rcds?
          BNH    INSERT                A. No, insert another record
          OI     SYSPRFL,INSDONE        Indicate we are done insert
          LA     WORKREG,TRAILER        Point to trailer record

```

```

        STCM  WORKREG,15,EIRECPTR      Store into EIRECPTR
        LH    WORKREG,MSGLEN           Update EIRECLEN
        STCM  WORKREG,15,EIRECLEN
        LH    RC,EIRC08                Set insert record retcode
        B     SYSPEXIT
INSERT  MH    MSGOFF,MSGLEN           Multiply by MSGLEN to get
        LA    WORKREG,MSGTABLE         offset into message table
        LA    WORKREG,0(MSGOFF,WORKREG)
        STCM  WORKREG,15,EIRECPTR     Store address in EIRECPTR
        LH    WORKREG,MSGLEN
        STCM  WORKREG,15,EIRECLEN     Store length in EIRECLEN
        L     WORKREG,INSERTCT        Increment MSGCOUNT
        LA    WORKREG,1(WORKREG)
        ST    WORKREG,INSERTCT        Save new count
        LH    RC,EIRC08                Set insert record retcode
        B     SYSPEXIT
NOTLAST LH    RC,EIRC00                Set normal process retcode
SYSPEXIT STH  RC,EIRETCOD             Store retcode in EIDB
        BR    14                     Return to intercept processor
*****
* AIOPT03: Read Tape Block Exit *
*   Count Tape Block records read *
*****
AIOPT03 LH    WORKREG,RTBCNT          Increment Read Tape Block
        LA    WORKREG,1(WORKREG)      count
        STH  WORKREG,RTBCNT
        LH    RC,EIRC00                Set normal process retcode
        STH  RC,EIRETCOD             Store retcode in EIDB
        BR    14                     Return to intercept processor
*****
* AIOPT04: Read Logical Tape Block Exit *
*   Count Logical Tape Block records read *
*****
AIOPT04 LH    WORKREG,RLTBCNT        Increment Read logical Tape
        LA    WORKREG,1(WORKREG)      Block count
        STH  WORKREG,RLTBCNT
        LH    RC,EIRC00                Set normal process retcode
        STH  RC,EIRETCOD             Store retcode in EIDB
        BR    14                     Return to intercept processor
*****
* AIOPT05: Write Logical Tape Block Exit *
*   Count Logical Tape Block records written *
*****
AIOPT05 LH    WORKREG,WLTBCNT        Increment Write logical Tape
        LA    WORKREG,1(WORKREG)      Block count
        STH  WORKREG,WLTBCNT
        LH    RC,EIRC00                Set normal process retcode
        STH  RC,EIRETCOD             Store retcode in EIDB
        BR    14                     Return to intercept processor
*****
* AIOPT06: Write Tape Block Exit *
*   Count Tape Block records written *
*****
AIOPT06 LH    WORKREG,WTBCNT        Increment Write Logical
        LA    WORKREG,1(WORKREG)      Tape Block count
        STH  WORKREG,WTBCNT
        LH    RC,EIRC00                Set normal process retcode
        STH  RC,EIRETCOD             Store retcode in EIDB
        BR    14                     Return to intercept processor
*
*****
* AIOPT07: Read DASD Track Exit *
*   Count DASD Tracks read *
*****
AIOPT07 LH    WORKREG,RDTCNT        Increment Read DASD Track
        LA    WORKREG,1(WORKREG)      count
        STH  WORKREG,RDTCNT
        LH    RC,EIRC00                Set normal process retcode
        STH  RC,EIRETCOD             Store retcode in EIDB
        BR    14                     Return to intercept processor
*****
* AIOPT08: Write DASD Track Exit *

```

```

*   Count DASD Tracks written
*****
AIOPT08  LH   WORKREG,WDTCNT      Increment Write DASD Track
        LA   WORKREG,1(WORKREG)  count
        STH  WORKREG,WDTCNT
        LH   RC,EIRC00           Set normal process retcode
        STH  RC,EIRETCOD         Store retcode in EIDB
        BR   14                 Return to intercept processor
*****
* AIOPT09: Read Utility SYSPRINT
*****
AIOPT09  LH   RC,EIRC00           Set normal process retcode
        STH  RC,EIRETCOD         Store retcode in EIDB
        BR   14                 Return to intercept processor
*****
* AIOPT10: Write SYSPRINT Record
*****
AIOPT10  LH   RC,EIRC00           Set normal process retcode
        STH  RC,EIRETCOD         Store retcode in EIDB
        BR   14                 Return to intercept processor
*****
* AIOPT11: Write WTO Message
*   Delete the WTO and insert my own
*****
AIOPT11  LH   WORKREG,WTOCT      Check WTO count
        LTR  WORKREG,WORKREG     Q. Is WTOCT 0?
        BZ   OPT11INS            A. Yes, insert one
OPT11DEL LH   RC,EIRC12           Set delete record retcode
        B    OPT11XIT            (delete all other WTOS)
OPT11INS LA   WORKREG,1(WORKREG)  Increment WTOCT
        STH  WORKREG,WTOCT
        LA   WORKREG,UIMWTO      Get address of WTO
        STCM WORKREG,15,EIRECPTR Store it in EIRECPTR
        LA   WORKREG,WTOLEN     Get length of WTO
        STCM WORKREG,15,EIRECLEN Store length it in EIRECLEN
        LH   RC,EIRC08           Set insert record retcode
        B    OPT11XIT
OPT11XIT STH  RC,EIRETCOD         Store retcode in EIDB
        BR   14                 Return to intercept processor
*****
* AIOPT12: Write WTOR Message
*   Check for ADR369D, never allow it to write over VTOC, etc
*****
AIOPT12  ICM  WORKREG,15,EIRECPTR
        CLC  12(8,WORKREG),ADR369D Q. Authorize request?
        BNE  OPT12RC0            A. No, let DFSMSdss do WTOR
        LA   WORKREG,UIMRESP     Get address of response
        STCM WORKREG,15,EIRECPTR Store in EIRECPTR
        LA   WORKREG,RESPLEN     Get length of response
        STCM WORKREG,15,EIRECLEN Store in EIRECLEN
        LH   RC,EIRC28           Set WTOR Response retcode
        B    OPT12XIT
OPT12RC0 LH   RC,EIRC00           Set normal process retcode
OPT12XIT STH  RC,EIRETCOD         Store retcode in EIDB
        BR   14                 Return to intercept processor
*****
* AIOPT13: Present ADRUFO Record
*   If parm call, set IO and AI buffers above 16M
*   If function call and Copy, force reblocking.
*****
AIOPT13  ICM  TEMPBASE,15,EIRECPTR
        USING ADRUFOB,TEMPBASE   Get addressability to UFO
        CLC  UFFUNCT,PARM        Q. Is this a PARM call?
        BE   PARMCALL            A. Yes, go to parm call
FUNCTION TM  UFFUNCT1,UFFUCOPY   Is this a copy?
        BNO  OPT13RC0            No, don't change options
        SR   WORKREG,WORKREG
        ICM  WORKREG,3,UFBDOFF   Get offset for PARM list
        AR   WORKREG,TEMPBASE    Add to address of UFO
        USING UFOFUNCT,WORKREG   Establish addressability
        OI   UFO3FLGS,UFOFRBLK  Force reblocking
        DROP WORKREG

```

```

      LH      RC,EIRC16          Set modify record retcode
      B       OPT13XIT
PARMCALL SR      WORKREG,WORKREG This is a parm call.
      ICM     WORKREG,3,UFBDOFF  Get offset for PARM list
      AR      WORKREG,TEMPBASE   Add to address of UFO
      USING   UFOFUNCT,WORKREG   Establish addressability
      OI      UFXAFLAG,UFXABUFF  Make sure IO buffers are >16M
      OI      UFXAFLAG,UFAI31B   Make sure AI buffers are >16M
      DROP    WORKREG
      LH      RC,EIRC16          Set modify record retcode
      LA      WORKREG,2          Initialize message table:
      ST      WORKREG,MSGCOUNT  Initialize msgcount
      DROP    TEMPBASE
      LA      TEMPBASE,MSGTABLE  Get address of table
      USING   TABLEMAP,TEMPBASE
      MVC     TABENTRY(133),HEADER Move header into msgtable
      LA      TEMPBASE,133(TEMPBASE) Move past first message
      MVC     TABENTRY(133),BLAN  Move blank line into table
      DROP    TEMPBASE
      B       OPT13XIT          Exit.
OPT13RC0 LH      RC,EIRC00       Set normal process retcode
OPT13XIT EQU     *
      STH     RC,EIRETCOD       Store retcode in EIDB
      BR      14               Return to intercept processor
*****
* AIOPT14: Function Ending *
*****
AIOPT14 LH      WORKREG,RTBCNT   Get Tape Blocks Read
      CVD     WORKREG,CVDWORK   Convert to decimal
      UNPK    RTBINS,CVDWORK    Unpack into message insert
      OI      RTBINS+7,X'F0'    Make last character printable
      LH      WORKREG,WTBCNT    Get Tape Blocks Written
      CVD     WORKREG,CVDWORK   Convert to decimal
      UNPK    WTBINS,CVDWORK    Unpack into message insert
      OI      WTBINS+7,X'F0'    Make last character printable
      LH      WORKREG,RLTBCNT   Get Tape Logical Blocks Read
      CVD     WORKREG,CVDWORK   Convert to decimal
      UNPK    RLTBINS,CVDWORK   Unpack into message insert
      OI      RLTBINS+7,X'F0'   Make last character printable
      LH      WORKREG,WLTBCNT   Get Tape Log Blocks Written
      CVD     WORKREG,CVDWORK   Convert to decimal
      UNPK    WLTBINS,CVDWORK   Unpack into message insert
      OI      WLTBINS+7,X'F0'   Make last character printable
      LH      WORKREG,RDTCNT    Get Disk Tracks Read
      CVD     WORKREG,CVDWORK   Convert to decimal
      UNPK    RDTINS,CVDWORK    Unpack into message insert
      OI      RDTINS+7,X'F0'    Make last character printable
      LH      WORKREG,WDTCNT    Get Disk Tracks Written
      CVD     WORKREG,CVDWORK   Convert to decimal
      UNPK    WDTINS,CVDWORK    Unpack into message insert
      OI      WDTINS+7,X'F0'    Make last character printable
      L       MSGOFF,MSGCOUNT  Get current MSGCOUNT
      MH      MSGOFF,MSGLEN     Multiply by MSGLEN to get
      LA      WORKREG,MSGTABLE  offset into message table
      LA      TEMPBASE,0(MSGOFF,WORKREG)
      USING   TABLEMAP,TEMPBASE
      MVC     TABENTRY(133),READMSG Move into msgtable
      LA      TEMPBASE,133(TEMPBASE) Increment pointer for 2nd msg
      MVC     TABENTRY(133),WRITEMSG Move into msgtable
      DROP    TEMPBASE
      L       WORKREG,MSGCOUNT  Increment MSGCOUNT
      LA      WORKREG,2(WORKREG)
      ST      WORKREG,MSGCOUNT  Save new count
      LH      RC,EIRC00       Set normal process retcode
      STH     RC,EIRETCOD     Store retcode in EIDB
      BR      14               Return to intercept processor
*****
* AIOPT15: Present WTOR Response *
*****
AIOPT15 LH      RC,EIRC00       Set normal process retcode
      STH     RC,EIRETCOD     Store retcode in EIDB
      BR      14               Return to intercept processor

```

```

*****
* AIOPT16: OPEN/EOF Tape Security and Verification *
* Always bypass password protection *
*****
AIOPT16 ICM TEMPBASE,15,EIRECPTR Get address of record
        LH RC,EIRC16 Bypass password protection
        ST RC,0(TEMPBASE) Store in EIRECPTR
        LH RC,EIRC16 Set modify record retcode
        STH RC,EIRETCOD Store retcode in EIDB
        BR 14 Return to intercept processor
*****
* AIOPT17: OPEN/EOF Nonspecific Tape Mount *
* Supply DFSMSdss with TAPE01 *
*****
AIOPT17 ICM TEMPBASE,15,EIRECPTR Get address of record
        MVC 0(6,TEMPBASE),TAPE01 Supply TAPE01 as volser
        LH RC,EIRC16 Set modify record retcode
        STH RC,EIRETCOD Store retcode in EIDB
        BR 14 Return to intercept processor
*****
* AIOPT18: Insert logical VSAM Record during Logical Restore *
*****
AIOPT18 LH RC,EIRC00 Set normal process retcode
        STH RC,EIRETCOD Store retcode in EIDB
        BR 14 Return to intercept processor
*****
* AIOPT19: Output Tape I/O Error *
* Save DDNAME, Volser, and Return code, print record. *
*****
AIOPT19 ICM TEMPBASE,15,EIRECPTR Get address of EIRECORD
        USING EIDINFO,TEMPBASE Est addressability to DDINFO
        MVC OPT19DD,EIDDDNAME Move DDNAME to message
        MVC OPT19VS,EIVOLID Move VOLSER to message
        SR WORKREG,WORKREG Clear out work register
        IC WORKREG,EIRETC Get return code
        CVD WORKREG,CVDWORK Convert to decimal
        UNPK OPT19RC(2),CVDWORK+6(2) Unpack into message insert
        OI OPT19RC+1,X'F0' Make last character printable
        DROP TEMPBASE
        L MSGOFF,MSGCOUNT Get current MSGCOUNT
        MH MSGOFF,MSGLEN Multiply by MSGLEN to get
        LA WORKREG,MSGTABLE offset into message table
        LA TEMPBASE,0(MSGOFF,WORKREG)
        USING TABLEMAP,TEMPBASE
        MVC TABENTRY(133),OPT19MSG Move into msgtable
        DROP TEMPBASE L WORKREG,MSGCOUNT Increment MSGCOUNT
        LA WORKREG,1(WORKREG)
        ST WORKREG,MSGCOUNT Save new count
        LH RC,EIRC00 Set normal process retcode
        STH RC,EIRETCOD Store retcode in EIDB
        BR 14 Return to intercept processor
*****
* AIOPT20: Volume Information Exit *
* Save Data Set name *
*****
AIOPT20 ICM TEMPBASE,15,EIRECPTR Establish addressability to
        USING EIREC20,TEMPBASE EIREC20
        MVC OPT20DSN(44),EI20DSN Move data set name into msg
        DROP TEMPBASE
        L MSGOFF,MSGCOUNT Get current MSGCOUNT
        MH MSGOFF,MSGLEN Multiply by MSGLEN to get
        LA WORKREG,MSGTABLE offset into message table
        LA TEMPBASE,0(MSGOFF,WORKREG)
        USING TABLEMAP,TEMPBASE
        MVC TABENTRY(133),OPT20MSG Move into msgtable
        DROP TEMPBASE
        L WORKREG,MSGCOUNT Increment MSGCOUNT
        LA WORKREG,1(WORKREG)
        ST WORKREG,MSGCOUNT Save new count
        LH RC,EIRC00 Set normal process retcode
        STH RC,EIRETCOD Store retcode in EIDB
        BR 14 Return to intercept processor

```

```

*****
* AIOPT21: Data Set Verification Exit *
* Do Not Process TEMP Data Sets *
*****
AIOPT21 EQU *
        ICM TEMPBASE,15,EIRECPTR Get record pointer
        USING EIREC21,TEMPBASE Get addressability
        CLC EI21DSN(4),TMPL If first 4 char of dsn='TEMP'
        DROP TEMPBASE Release addressability
        BNE LABEL1 Then
        LH RC,EIRC36 Do not process data set
        B LABEL2 Else
LABEL1 LH RC,EIRC00 Set normal return code
LABEL2 STH RC,EIRETCOD Store retcode in EIDB
        BR 14 Return to intercept processor
*****
* AIOPT22: Bypass Verification Exit *
* Bypass Serialization of SYS1 Data Sets *
*****
AIOPT22 EQU *
        ICM TEMPBASE,15,EIRECPTR Get record pointer
        USING EIREC22,TEMPBASE Get addressability
        CLC EI22DSN(5),SYSL If first 5 char of dsn='SYS1.'
        BNE LAB1 Then
        LH RC,EIRC16 Bypass serialization of dsn
        OI EI22FLGS,B'10000000' Set EI22BSER bit on
        B LAB2 Else
        DROP TEMPBASE Release addressability
LAB1 LH RC,EIRC00 Set normal return code
LAB2 STH RC,EIRETCOD Store retcode in EIDB
        BR 14 Return to intercept processor
*****
* AIOPT23: Data Set Processed Notification Exit *
* End Function If Errors in Processing *
*****
AIOPT23 EQU *
        ICM TEMPBASE,15,EIRECPTR Get record pointer
        USING EIREC23,TEMPBASE Get addressability
        LH WORKREG,EI23DSRC If data set processing RC^=0
        DROP TEMPBASE Release addressability
        LTR WORKREG,WORKREG Then
        BZ LAB1
        LH RC,EIRC32 End function
        B LAB2 Else
LAB1 LH RC,EIRC00 Set normal return code
LAB2 STH RC,EIRETCOD Store retcode in EIDB
        BR 14 Return to intercept processor
*****
* AIOPT24: Concurrent Copy initialization complete *
* End function if initialization is not successful *
*****
AIOPT24 EQU *
        LH RC,EIRC00 Start EIOP24 processing
        ICM TEMPBASE,15,EIRECPTR Assume goodness
        USING EIREC24,TEMPBASE Get record pointer
        LH WORKREG,EI24RTCD Get addressability
        DROP TEMPBASE Get the return code
        LTR WORKREG,WORKREG Release addressability
        BZ OPT24END If zero
        LH RC,EIRC32 End function
        OPT24END STH RC,EIRETCOD Else end the function
        BR 14 Store retcode in EIDB
        Return to intercept processor
*
*****
* AIOPT25: Backspace tape input *
*****
AIOPT25 EQU *
        LH RC,EIRC00 Start EIOP25 processing
        STH RC,EIRETCOD Set normal process retcode
        BR 14 Store retcode in EIDB
        Return to intercept processor
*****
* AIOPT26: Volume open for output *

```



```

*****
AIOPT26 EQU * Start EIOP26 processing
          LH RC,EIRC00 Set normal process retcode
          STH RC,EIRETCOD Store retcode in EIDB
          BR 14 Return to intercept processor
*****
* Start of local variables *
*****
*
*
CURCOMM DC F'0' Address of current command
SYSINFL DC X'00' SYSIN Flag
SYSIFRST EQU X'80' First entry SYSIN Performed
SYSPRFL DC X'00' SYSPRINT Flag
INSDONE EQU X'80' Done inserting SYSPRINT recs
TMPL DC C'TEMP' TEMP high-level qualifier
SYSL DC C'SYS1.' SYS1 high-level qualifier
*
SAVEAREA DC 18F'0' My save area
*
VECTABLE DC A(AIOPT00) Function Startup Exit
VEC01 DC A(AIOPT01) Read SYSIN Exit
VEC02 DC A(AIOPT02) Write SYSPRINT Exit
VEC03 DC A(AIOPT03) Read Tape Block Exit
VEC04 DC A(AIOPT04) Read Logical Tape Exit
VEC05 DC A(AIOPT05) Write Logical Tape Exit
VEC06 DC A(AIOPT06) Write Tape Block Exit
VEC07 DC A(AIOPT07) Read DASD Track Exit
VEC08 DC A(AIOPT08) Write DASD Track Exit
VEC09 DC A(AIOPT09) Read Utility Sysprint Exit
VEC10 DC A(AIOPT10) Write SYSPRINT Record
VEC11 DC A(AIOPT11) Write WTO Message Exit
VEC12 DC A(AIOPT12) Write WTOR Message Exit
VEC13 DC A(AIOPT13) Present ADRUFO Record Exit
VEC14 DC A(AIOPT14) Function Ending Exit
VEC15 DC A(AIOPT15) Present WTOR Response
VEC16 DC A(AIOPT16) OPEN/EOF Tape Sec/Ver Exit
VEC17 DC A(AIOPT17) OPEN/EOF NonSpec Tape Mount
VEC18 DC A(AIOPT18) Insert log VSAM Rcd -Restore
VEC19 DC A(AIOPT19) Output Tape I/O Error Exit
VEC20 DC A(AIOPT20) Volume Information Exit
VEC21 DC A(AIOPT21) Data Set Verification Exit
VEC22 DC A(AIOPT22) Bypass Verification Exit
VEC23 DC A(AIOPT23) Data Set Processed Exit
VEC24 DC A(AIOPT24) Concurrent Copy Init Complete
*
VEC25 DC A(AIOPT25) Backspace tape input
VEC26 DC A(AIOPT26) Volume open for output
*
UIMWTO DC C'>>> USRUIM is active <<<'
*
WTOLEN EQU *-UIMWTO Show that we are active
*
UIMRESP DC C'T' Reply T to ADR369
RESPLEN EQU *-UIMRESP
*
ADR369D DC CL8'ADR369D' DFSMSdss Authorization WTOR
ADR012I DC CL8'ADR012I' Last DFSMSdss message id
*
MSGLEN DC H'133' Length of messages
INSERTCT DC F'0' Count of msgs given to DFSMSdss
MSGCOUNT DC F'0' Count of messages in table
*
RTBCNT DC H'0' Count of Tape Blocks Read
WTBCNT DC H'0' Count of Tape Blocks Written
*
RLTBCNT DC H'0' Count Log Tape Blocks Read
WLTBCNT DC H'0' Count Log Tape Blocks Written
*
RDCNT DC H'0' Count Disk Tracks Read
WDCNT DC H'0' Count Disk Tracks Written

```

```

*
WTOCT    DC    H'0'                      Count of WTOs
*
PARM     DC    X'0000'                   Parm call test for Opt 13
*
TAPE01   DC    CL6'TAPE01'               Tape volser for Opt 17
*
*
HEADER    DC    C' ***** Begin USRUIM Messages '
          DC    CL(133-(*-HEADER))'*****C
          *****
*
TRAILER   DC    C' ***** End USRUIM Messages '
          DC    CL(133-(*-TRAILER))'*****C
          *****
*
BLANKS    DC    CL133' '                 Blank message
*
READMSG   DC    C'   Disk Tracks Read:   '
RDTINS    DC    C'XXXXXXXX'
          DC    C'   Tape Blocks Read:   '
RTBINS    DC    C'XXXXXXXX'
          DC    C'   Logical Tape Blocks Read: '
RLTBINS   DC    C'XXXXXXXX'
          DC    CL(133-(*-READMSG))' '
*
WRITEMSG  DC    C'   Disk Tracks Written: '
WDTINS    DC    C'XXXXXXXX'
          DC    C'   Tape Blocks Written: '
WTBINS    DC    C'XXXXXXXX'
          DC    C'   Logical Tape Blocks Written: '
WLTBINS   DC    C'XXXXXXXX'
          DC    CL(133-(*-WRITEMSG))' '
*
OPT00MSG  DC    C'   Messages follow for task: '
OPT00TSK  DC    C'XX'
          DC    CL(133-(*-OPT00MSG))' '
*
OPT19MSG  DC    C'   Permanent Tape Error for DDNAME: '
OPT19DD   DC    C'XXXXXXXX'
          DC    C' VOLSER: '
OPT19VS   DC    C'XXXXXX'
          DC    C' Return Code: '
OPT19RC   DC    C'XX'
          DC    CL(133-(*-OPT19MSG))' '
*
OPT20MSG  DC    C'   EI20DSN = '''
OPT20DSN  DC    C'XXXXXXXX.XXXXXXXX.XXXXXXXX.XXXXXXXX.XXXXXXXX'
          DC    C'''
          DC    CL(133-(*-OPT20MSG))' '
*
CVDWORK   DS    D                       Workarea for decimal conversion
*
          ADREID0                       Macro to map EIDB
*
          ADRUFO                         Macro to map UFO
*
USRUIM    CSECT
MSGTABLE  DS    CL13300                 Room in table for 100 msgs
*
TABLEMAP  DSECT
TABENTRY  DS    CL133                 DSECT to insert msgs into table
*
SYSPMAP   DSECT
          DS    CL1                   DSECT to find MSGID in DFSMSdss
MSGID     DS    CL7                   messages
*
          END

```

Output resulting from use of the UIM exits

```

PAGE 0001      5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES      1999.211 14:56

ADR010I (SCH)-PRIME(01), SIZE VALUE OF 4096K WILL BE USED FOR GETMAIN
WTO 'USRAIPGM INVOKING DFSMSdss'
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'WTO '
DUMP DS(EXC(SYS1.**)) LOGINDD(DASD) OUTDD(TAPE)
ADR101I (R/I)-RI01 (01), TASKID 002 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
IF LASTCC=0 -
ADR101I (R/I)-RI01 (01), TASKID 003 HAS BEEN ASSIGNED TO COMMAND 'IF '
THEN DEFrag DDN(DASD)
ADR101I (R/I)-RI01 (01), TASKID 004 HAS BEEN ASSIGNED TO COMMAND 'DEFrag '
ADR109I (R/I)-RI01 (01), 1999.211 14:56:05 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED.
ADR050I (002)-PRIME(01), DFSMSDSS INVOKED VIA APPLICATION INTERFACE
ADR016I (002)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (002)-STEND(01), 1999.211 14:56:05 EXECUTION BEGINS
ADR788I (002)-DIVSM(03), PROCESSING COMPLETED FOR CLUSTER PUBSEXMP.KSDS.S01, 100 RECORD(S)
PROCESSED, REASON 0
ADR801I (002)-DTDSC(01), DATA SET FILTERING IS COMPLETE. 4 OF 4 DATA SETS WERE SELECTED: 0
FAILED SERIALIZATION
AND 0 FAILED FOR OTHER REASONS.
ADR454I (002)-DTDSC(01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
PUBSEXMP.SAM.S01
PUBSEXMP.PDS.S01
CLUSTER NAME PUBSEXMP.ESDS.S01
CATALOG NAME SYS1.MVSRES.MASTCAT
COMPONENT NAME PUBSEXMP.ESDS.S01.DATA
CLUSTER NAME PUBSEXMP.KSDS.S01
CATALOG NAME SYS1.MVSRES.MASTCAT
COMPONENT NAME PUBSEXMP.KSDS.S01.DATA
COMPONENT NAME PUBSEXMP.KSDS.S01.INDEX
ADR006I (002)-STEND(02), 1999.211 14:56:06 EXECUTION ENDS
ADR013I (002)-CLTSK(01), 1999.211 14:56:06 TASK COMPLETED WITH RETURN CODE 0000
ADR050I (004)-PRIME(01), DFSMSDSS INVOKED VIA APPLICATION INTERFACE
ADR016I (004)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (004)-STEND(01), 1999.211 14:56:07 EXECUTION BEGINS
ADR208I (004)-EANAL(01), 1999.211 14:56:07 BEGINNING STATISTICS ON D9S060:
FREE CYLINDERS 000058
FREE TRACKS 000003
FREE EXTENTS 000002
LARGEST FREE EXTENT (CYL,TRK) 000053,0003
PAGE 0002      5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES      1999.211 14:56
FRAGMENTATION INDEX 0.050
PERCENT FREE SPACE 97
ADR220I (004)-EANAL(01), INTERVAL BEGINS AT CC:HH 00001:0000 AND ENDS AT CC:HH 00006:000C
ADR209I (004)-EFRAG(01), 1999.211 14:56:07 MOVED EXTENT 001 FROM 00006:0000-00006:0004 TO
00001:0000-00001:0004 FOR
PUBSEXMP.ESDS.S01.DATA
ADR209I (004)-EFRAG(01), 1999.211 14:56:07 MOVED EXTENT 001 FROM 00006:0005-00006:0006 TO
00001:0005-00001:0006 FOR
PUBSEXMP.KSDS.S01.DATA
ADR209I (004)-EFRAG(01), 1999.211 14:56:07 MOVED EXTENT 001 FROM 00006:0007-00006:0008 TO
00001:0007-00001:0008 FOR
PUBSEXMP.SAM.S01
ADR209I (004)-EFRAG(01), 1999.211 14:56:07 MOVED EXTENT 001 FROM 00006:0009-00006:000A TO
00001:0009-00001:000A FOR
PUBSEXMP.PDS.S01
ADR209I (004)-EFRAG(01), 1999.211 14:56:08 MOVED EXTENT 002 FROM 00006:000B-00006:000B TO
00001:000B-00001:000B FOR
PUBSEXMP.PDS.S01
ADR213I (004)-EANAL(01), 1999.211 14:56:08 ENDING STATISTICS ON D9S060:
DATA SET EXTENTS RELOCATED 000005
TRACKS RELOCATED 000012
FREE CYLINDERS 000058
FREE TRACKS 000003
FREE EXTENTS 000001
LARGEST FREE EXTENT (CYL,TRK) 000058,0003
FRAGMENTATION INDEX 0.000
PAGE 0003      5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES      1999.211 14:56
ADR212I (004)-EANAL(01), EXTENT DISTRIBUTION MAP FOR D9S060:
EXTENT *FREE SPACE BEFORE* *FREE SPACE AFTER* * ALLOCATED *
SIZE
IN NO. CUM. NO. CUM. NO. CUM.
TRACKS EXTS PCT/100 EXTS PCT/100 EXTS PCT/100
1 4 0.148
2 4 0.444
5 1 0.629
10 1 1.000

```

```

              75          1    0.085
            >499          1    1.000          1    1.000
ADR006I (004)-STEND(02), 1999.211 14:56:08 EXECUTION ENDS
ADR013I (004)-CLTSK(01), 1999.211 14:56:08 TASK COMPLETED WITH RETURN CODE 0000
***** Begin USRUIM Messages *****
*****
Messages follow for task: 00
Messages follow for task: 02
  Disk Tracks Read:    00000014    Tape Blocks Read:    00000000    Logical Tape Blocks
Read:    00000000
  Disk Tracks Written: 00000000    Tape Blocks Written: 00000036    Logical Tape Blocks
Written: 00000012
Messages follow for task: 04
  Disk Tracks Read:    00000052    Tape Blocks Read:    00000000    Logical Tape Blocks
Read:    00000000
  Disk Tracks Written: 00000012    Tape Blocks Written: 00000000    Logical Tape Blocks
Written: 00000000
***** End USRUIM Messages *****
*****
ADR012I (SCH)-DSSU (01), 1999.211 14:56:08 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS
0000

```

Chapter 25. Data set attributes

This topic lists various attributes that DFSMSdss can set or change for a given data set, and identifies where DFSMSdss gets the attribute information from.

Locate the data set attribute of interest in the first column of [Table 42 on page 641](#). The remaining columns indicate where and under which conditions DFSMSdss finds the attribute information. For example, the data set size is usually determined by the source data set, unless the preallocated target data set is larger. The ALLDATA(x) and ALLEXCP keywords have an effect on the data set size.

Table 42. Data Set Attributes and How They Are Determined.

Attribute	Is the Attribute Determined by...			
	...the Source Data Set?	...the Preallocated Target?	...a Keyword?	...another Factor?
Data set name	Yes, if no RENAME or RENAMEU	No	RENAME or RENAMEU	No
Data set size	Yes, unless the preallocated target data set is larger	Yes, if it is big enough	ALLDATA(x) or ALLEXCP	No
Volumes	Yes, if doing a RESTORE and no output volumes were specified, no target exists, and not SMS-managed	Yes	Yes, if not SMS, OUTDDNAME(x,...) or OUTDYNAM(x,...)	If SMS-managed, ACS routines and SMS allocation choose volumes with most available space; if not SMS-managed, DFSMSdss chooses volumes with most available space
Data set location on volume	Yes, if no target and either ABSTR, PSU, POU, or DAU	Yes	FORCE can override ABSTR, PSU, POU, and DAU	DFSMSdss locates wherever space is available; DEFrag may move extents
PDS directory size (blocks)	Yes	No	No	No
PDSE directory size (blocks)	Yes	No	No	No
SMS Storage Class or Management Class	Yes, if no target and BYPASSACS is specified	Yes	Yes, if STORCLAS(x) or MGMTCLAS(x), or both, are specified with BYPASSACS	ACS routines if no target and BYPASSACS is not specified
SMS Data Class	Yes, if no target	Yes	No	No
BLKSZ	Yes, if REBLOCK is not specified and data set is not system reblockable	No	If REBLOCK keyword is specified, DFSMSdss chooses a new optimal blocksize	If system reblockable, DFSMSdss chooses a new optimal blocksize, or else the user can change blocksize with the installation reblock exit and can specify REBLOCK with the installation options exit
LRECL	Yes	No	No	No
RECFM	Yes	No	No	No
DSORG	Yes	No	No	No
Number of stripes	Yes, source must be striped	Yes for non-VSAM. No for VSAM	No	For nonguaranteed-space, determined by sustained data rate (SDR) in STORCLAS. For guaranteed-space, must have a nonzero SDR, then determined by number of output volumes supplied

Table 42. Data Set Attributes and How They Are Determined. (continued)

Attribute	Is the Attribute Determined by...			
	...the Source Data Set?	...the Preallocated Target?	...a Keyword?	...another Factor?
Number of volumes (VOLCOUNT)	Yes	Yes	VOLCOUNT can make a single volume source into a multivolume target, or change the number of volumes for a multivolume data set	Yes (see the COPY and RESTORE command VOLCOUNT parameter descriptions for specific information)
Number of extents	Yes, for imbedded extended KSDSs during physical data set restore	Yes	No	DFSMSDss always tries to consolidate during COPY/RESTORE. RELEASE may reduce the number of extents
PDS/PDSE	Yes, if no target or if doing a RESTORE	Yes, if doing a COPY	CONVERT(PDS(x)) or CONVERT(PDSE(x))	No
Cataloged	Yes, if RECATALOG(*) is specified, no target, and the user is not doing a physical data set restore	Yes	RECATALOG(x), CATALOG, UNCATALOG (applies to source only)	If SMS or VSAM, cataloged by default; if physical data set restore, only single volume non-VSAM is cataloged (if CATALOG is specified)
Allocation unit	Yes, if no target and TGTALLOC (SOURCE) specified or defaulted	Yes	TGTALLOC(x)	No
Free space in VSAM	Yes, if going to like device, nonVALIDATE, no dummy blocks, and CI and CA sizes do not change	Yes, if doing VSAM I/O and uses values in target catalog entry	VALIDATE, NOVALIDATE, FREESPACE	No
Security (RACF)	If no target and source was generic or discrete, and no applicable profile protecting new target, a new discrete will be defined	Yes	MENTITY, MVOLSER	Full RACF profile information (access lists) are not preserved
AIX data sets on VSAM clusters	If SPHERE is specified during COPY or DUMP and RESTORE, sphere and connections are preserved	No	SPHERE	No
GDS state	Yes, if no target and TGTGDS (source) specified	Yes	TGTGDS(x)	No
RLS BWO field	Yes, if no preallocated target and no UIM input	Yes, if no UIM input	No	UIM can pass a value in Exit 22 during logical RESTORE
RLS timestamps	Yes, if not a logical restore	No	No	For logical restore, if dumped using RLS access, timestamps reflect the time of the dump; otherwise the timestamps are zero
RLS recovery required	Yes, if no UIM input	No	No	UIM can pass a value in Exit 22 during logical RESTORE
RLS log parameter	Yes, if no preallocated target and no UIM input	Yes, if no UIM input	No	UIM can pass a value in Exit 22 during logical RESTORE
RLS log stream ID	Yes, if no preallocated target and no UIM input	Yes, if no UIM input	No	UIM can pass a value in Exit 22 during logical RESTORE
BCS Quiesce enabled	Yes, unless preallocated	Yes	No	Only applies to logical restore

Table 42. Data Set Attributes and How They Are Determined. (continued)

Attribute	Is the Attribute Determined by...			
	...the Source Data Set?	...the Preallocated Target?	...a Keyword?	...another Factor?
BCS lock or suspend	Yes, unless preallocated	Yes	No	Only applies to logical restore. BCSRECOVER(LOCK SUSPEND) is ignored if not preallocated.
BCS RLS in use	No	No	No	Only applies to logical restore. BCSRECOVER(LOCK SUSPEND) is ignored if not preallocated.
Class transition information	Yes, if no preallocated target	Yes, unless it is unusable and needs to be scratched and reallocated	No	No
LOGREPLICATE parameter	Yes, if no preallocated target and no UIM input	Yes, if no UIM input	No	UIM can pass a value in EXIT 22 during logical restore
extended format sequential version number	Yes, if no preallocated target	Yes, if target has to be scratched and reallocated, the target version type will be preserved	No	No, if the target data set does not exist the target will be allocated with the source version type
z/OS data set encryption	Yes	No, must be a usable target with like encryption eligibility.	No	No

Chapter 26. z/OS UNIX file attributes

This topic lists various attributes that DFSMSdss can set or change for a given z/OS UNIX file, and identifies where DFSMSdss gets the attribute information.

Since DFSMSdss behaves as a Virtual File System (VFS) server when processing z/OS UNIX files, it leverages the attributes of a file as mapped by the BPXYATTR macro.

Dump

DFSMSdss only updates the last backup date (AttrRefTime64) for regular files when the RESET keyword is specified. Other attributes can be updated by z/OS UNIX System Services as a result of processing files.

Restore

The restored file's attributes depend on whether the file was returned to the same location from which it was dumped, or if it was restored to a different location. If the file is being restored to a different location, then DFSMSdss identifies this as introducing a new file to the system. The resulting attributes are described in the following table:

Table 43. Attributes of the restored files			
Attribute	Attribute determined by		
	ADMINISTRATOR keyword specified	No Administrator keyword	
		Same Location	Different Location
File name	Source value	Source value	Source value
Character set information (Attcharsetid)	Source value	Source value	Source value
Permissions (Attrmode)	Source value ¹	Source value ¹	DSS defaults ²
Owning UID (AttrUid)	Source value	Source value	Effective UID
Owning GID (AttrGid)	Source value	Source value	Effective GID
Sticky bit (AttrNoDelFiles)	Source value	Source value	Not set
Shared Library (AttrShareLibMask)	Source value	Not set	Not set
Do not run in shareas (AttrNoShareasMask)	Source value	System default	System default
APF authorized program (AttrApfAuthMask)	Source value	Not set	Not set
Program controlled (AttrProgCtlMask)	Source value	Not set	Not set
External symlink (AttrExtLinkMask)	Source value	Source value	Source value
File size (AttrSize)	Source value	Source value	Source value
Auditor audit (AttrAuditorAudit)	Source value	Source value	Not set

Table 43. Attributes of the restored files (continued)

Attribute	Attribute determined by		
	ADMINISTRATOR keyword specified	No Administrator keyword	
		Same Location	Different Location
Auditor user (AttUserAudit)	Source value	Source value	Not set
Last access time (AttrAtime64)	Source value	Source value	Current time
Last modification time (AttrMtime64)	Source value ³	Source value ³	Current time
Last file stat change time (AttrCtime64)	Source value	Source value	Current time
Last reference time ⁴ (ATTRREFTIME64)	Source value	Source value	Not set
Creation time (AttrCreateTime64)	Current time	Current time	Current time
File format (ATTRREFTIME64)	Source value	Source value	Source value
Security label (ATTRSECLABEL)	Source value	Source value	System default
Notes	¹ Base and extended ACLs, if any, are preserved. ² Permission octets result in 640 for files and 750 for directories ³ When restoring a directory, its modification and change time can reflect the time of the restore operation because DFSMSdss restores the directories first followed by any files (if any) to the directories. Such operation results in those time fields being updated. ⁴ The last reference time is used as the last backup date. Refer to the RESET keyword for further information.		

Appendix A. Coexistence considerations

This topic presents considerations for migrating to a newer version of DFSMSdss.

Restoring backups using DFSMSdss

Dumps created with an older version or release of DFDSS or DFSMSdss can be restored with a newer version or release.

Data sets with extended attribute DSCBs can be restored to volumes that do not support extended attribute DSCBs; however, the extended attributes are lost. To prevent losing the extended attributes, restore the data set to volumes that support extended attribute DSCBs.

Appendix B. Accessibility

Accessible publications for this product are offered through [IBM Documentation for z/OS \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS documentation see [How to Send Feedback to IBM](#) to leave documentation feedback.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

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

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

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

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

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

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

IBM may use or distribute any of the information you 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
Site Counsel
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 document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

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

COPYRIGHT LICENSE:

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

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: [IBM Lifecycle Support for z/OS \(www.ibm.com/software/support/systemsz/lifecycle\)](http://www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

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 Programming Interface Information

Trademarks

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

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

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

Glossary

This glossary defines technical terms and abbreviations that are used in DFSMSdss documentation. If you do not find the term that you are looking for, refer to the index of the appropriate DFSMSdss manual.

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.

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 (VVDSSs).

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.

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.

CREDIT

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 (DFSMS)

The complementary functions of DFSMSdftp, 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 FlashCopy

One of the FlashCopy Version 2 functions. See also *FlashCopy Version 2*.

data-set-changed indicator.

A bit that is set when a data set is opened for processing other than input.

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*.

DFSMSdftp

A DFSMS functional component that provides functions for storage management, data management, program management, device management, and distributed data access.

DFSMSdss

A DFSMS functional component used to copy, move, dump, and restore data sets and volumes. DFSMSdss is the primary data mover of DFSMS.

DFSMShsm

A DFSMS functional component used for backing up and recovering data, and managing space on volumes in the storage hierarchy.

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.

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 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 same ESS. DFSMS exploits data set FlashCopy.

FMID

Function modification identifier.

forward recovery

The CICSVR function that reapplies all changes to the VSAM sphere or RLS user catalog 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 or RLS user catalog 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

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 z/OS that permits online problem management, interactive problem diagnosis, problem tracking, and problem reporting.

Interactive Storage Management Facility (ISMF)

An interactive interface of z/OS 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.

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.

MSGADRNnt

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 as resource allocation, scheduling, input/output control, and data management.

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.

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 program

A DFSMSdss program that allows you to restore a full volume or particular tracks from a dump tape. This program can be used without a host system environment.

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)

A z/OS 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 z/OS 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 z/OS system.

System Modification Program Extended (SMP/E)

An IBM licensed program used to install software and software changes on the z/OS 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.

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.

Z**zFS**

See *z/OS File System*.

z/OS File System (zFS)

A z/OS UNIX file system. zFS stores files in VSAM linear data sets.

Index

Special Characters

- * (single asterisk) used in partially qualified data set names [28](#), [256](#)
- ** (double asterisk) used in partially qualified data set names [28](#), [256](#)
- % (percent sign) used in partially qualified data set names [28](#), [256](#)

Numerics

- 16 MB virtual storage, storage requirements per command [19](#), [20](#)
- 16 megabytes virtual storage
 - buffers above [251](#)
- 31-bit addressing mode [587](#)
- 3380 Direct Access Storage [93](#), [125](#)
- 3390 Direct Access Storage [93](#), [125](#)
- 3494 Tape Library [516](#)
- 3495 Tape Library [516](#)
- 64-bit addressing mode [251](#)
- 9345 DASD module [93](#), [125](#)

A

- ABEND keyword [249](#)
- ABOVE16 keyword [251](#)
- access authorization for DFSMSdss commands [548](#)
- accessibility
 - contact IBM [649](#)
- ACCESSVOL keyword
 - FlashCopy relationship, volume [129](#)
 - thawing the frozen, volume [130](#)
- ACL (automatic cartridge loader) [518](#)
- ACS information [165](#)
- ACS variables
 - CONVERTV [166](#)
 - COPY [165](#)
 - name/description [165](#)
 - passed in COPY command [165](#)
 - passed in RESTORE/CONVERTV [166](#)
 - RESTORE [166](#)
- ACTIVE state [88](#), [118](#)
- ACTIVE subkeyword of TGTGDS
 - COPY command [355](#)
 - RESTORE command [490](#)
- ADMINISTRATOR keyword
 - COMPRESS command [282](#)
 - CONSOLIDATE command [288](#)
 - COPY command [315](#)
 - DEFRAG command [374](#)
 - DUMP command [397](#)
 - FACILITY class profiles for [543](#)
 - PRINT command [434](#)
 - RELEASE command [444](#)
 - RESTORE command [462](#)
- ADMINISTRATOR keyword (*continued*)
 - with FlashCopy [126](#)
 - with native SnapShot [131](#)
- ADMINISTRATOR parameter [270](#)
- ADMINISTRATOR parameter (Stand-Alone BUILD command) [270](#)
- ADRDSU [577](#)
- ADRDSU program
 - using for Linux partition [171](#)
- ADREID0 data area [609](#)
- ADRMCLVL macro [625](#)
- ADRTAPB [180](#)
- ADRUFO, presenting record – eioption 13 [594](#)
- ADRUIM module [579](#)
- ADRXMAIA [580](#)
- ADSP (Automatic Data Set Protection) [537](#), [538](#)
- AIX (alternate index), recataloging [319](#)
- aliases of non-VSAM data sets [51](#), [81](#), [113](#)
- ALLDATA keyword
 - COPY command [315](#)
 - copying data set to like device [366](#), [367](#)
 - copying data set to unlike device [368](#), [369](#)
 - DUMP command [398](#)
 - dumping records past last-used-block pointer [52](#), [73](#)
 - PRINT command [434](#)
 - specified during a dump [73](#)
 - to control what DFSMSdss copies [98](#)
 - with preallocated target [123](#)
- ALLEXCP keyword
 - COPY command [316](#)
 - copying data set to like device [366](#), [367](#)
 - copying data set to unlike device [368](#), [369](#)
 - DUMP command [398](#)
 - dumping records past last-used-block pointer [52](#), [73](#), [172](#)
 - Linux dumps [172](#)
 - specified during a DUMP [73](#), [172](#)
 - to control what DFSMSdss copies [98](#), [172](#)
 - with preallocated target [123](#), [172](#)
- ALLOC keyword [30](#), [258](#)
- allocation space [354](#), [489](#)
- ALTER LOCK, IDCAMS command [80](#)
- alternate index, recataloging [319](#)
- always-call [537](#), [539](#)
- AMDSADMP (SADMP) service aid [523](#)
- AMSGCNT keyword [249](#)
- application interface
 - function [33](#)
 - invoking [33](#)
 - module names [36](#)
 - summary
 - data set processing [608](#)
 - record processing [608](#)
- application program process [627](#)
- archive [39](#), [42](#)
- ASIDPTR [578](#)

- assignment of class names using the RESTORE and COPY commands [496](#)
- assistive technologies [649](#)
- asterisks, in a data set qualifier [256](#)
- auditing information [33](#)
- authorization checking for EXPORT/IMPORT (IDCAMS commands) [105](#)
- authorization installation exit routine [4](#)
- authorization levels for BUILDSEA command [529](#)
- authorization structure [35](#)
- automatic cartridge loader [518](#)
- Automatic Data Set Protection [538](#)
- AUTORELBLOCKADDRESS keyword
 - COPY command [317](#)
 - RESTORE command [462](#)
- auxiliary commands
 - continuation errors [513](#)
 - EOJ [514](#)
 - IF-THEN-ELSE [510](#)
 - PARALLEL [508](#)
 - SERIAL [508](#)
 - SET [509](#)
 - WTO [507](#)
 - WTOR [507](#)
- availability [39](#)
- availability management [5](#)
- availability strategy, planning [39](#)
- avoiding lockout [563](#)

B

- backing up (dumping) data [386](#)
- backing up HFS data sets [49](#)
- backing up volumes
 - with incremental FlashCopy [64](#)
- backspace physical tape record – eioption 25 [602](#)
- backup
 - concurrent copy [7](#), [45](#), [106](#)
 - data set [5](#), [39](#), [42](#), [48](#)
 - DFSMSHsm and DFSMSdss [7](#)
 - disaster recovery [27](#)
 - file [5](#)
 - incremental [5](#)
 - integrated catalog facility user catalog [51](#)
 - reducing time
 - using volume dump and volume copy [57](#)
 - scenario [48](#)
 - SMS-managed data sets [53](#)
 - SMS-managed environment [40](#)
 - special requirements [48](#)
 - system volumes [56](#)
 - vital records [26](#), [39](#)
 - volume [5](#), [39](#), [55](#)
- backup copies, multiple [369](#)
- backup-while-open serialization
 - CICS [570](#)
 - concurrent copy [573](#)
 - DFSMSHsm [570](#)
 - general processing [572](#)
 - IMS data sets [574](#)
 - overview [570](#)
 - status definitions [572](#)
- BCSRECOVER keyword
 - RESTORE command [463](#)

- BELOW16 keyword [251](#)
- BLK subkeyword of TGTALLOCC
 - COPY command [354](#)
 - RESTORE command [489](#)
- BLKSIZE and LRECL keywords, specifying in a print operation [436](#)
- block size (DFSMSDss dump data set)
 - controlling at dump time [415](#)
 - default when dumping to tape or DASD [59](#), [415](#)
 - minimum [59](#)
 - with COPYDUMP command [369](#)
- block structure, calling [577](#)
- blocks, application interface [584](#)
- broken data sets [27](#), [263](#)
- buffers above 16 megabytes [251](#)
- buffers below 2 gigabyte real storage [251](#)
- building a Stand-Alone Services IPL-able core image [269](#)
- BUILDSEA command
 - for DFSMSDss [269](#)
- BY keyword
 - COMPRESS command [282](#)
 - CONSOLIDATE command [288](#)
 - COPY command [317](#)
 - criteria [27](#), [255](#)
 - DEFRAG command [374](#)
 - DUMP command [399](#)
 - filtering criteria [263](#)
 - operator meaning [29](#)
 - RELEASE command [444](#)
 - RESTORE command [463](#)
- bypass verification exit – eioption 22 [596](#)
- BYPASSACS keyword
 - COPY command [317](#)
 - RESTORE command [464](#)

C

- calling block structure [577](#)
- CANCELERROR keyword
 - CONSOLIDATE command [288](#)
 - COPY command [318](#)
 - DUMP command [399](#)
 - RESTORE command [464](#)
 - SPACEREL command [503](#)
- card readers supported [21](#)
- catalog
 - access authority [547](#)
 - during logical restore [76](#)
 - integrated catalog facility [80](#)
 - locking [80](#)
 - moving [112](#)
 - restoring [80](#)
 - standard search order [263](#)
 - temporary copied [24](#)
- CATALOG keyword
 - CONSOLIDATE command [288](#)
 - CONVERTV command [296](#)
 - COPY command [318](#)
 - during logical restore processing [76](#)
 - RESTORE command [465](#)
 - with preallocated target [123](#)
- catalog management services [533](#)
- cataloging non-VSAM data sets during restore [78](#)
- CATLG keyword [29](#), [258](#)

- CDA provider Transparent Cloud Tiering TCT [10](#)
- CDACOMPRESS keyword
 - RESTORE command [401](#)
- CGCREATED
 - creating consistent copies, volume [129](#)
 - FlashCopy relationship, volume [129](#)
 - thawing the frozen, volume [130](#)
 - verifying the consistency group, volume [130](#)
- CGCREATED command
 - for DFSMSdss [274](#)
- changing
 - management class with restore [87](#)
 - storage class with RESTORE [86](#)
- characteristics
 - data sets [255](#)
 - filtering by [258](#)
- characteristics of data sets [27](#)
- CHECKVTOC keyword
 - COPY command [319](#)
 - DUMP command [399](#)
- CHECKVTOC keyword, data integrity [27](#)
- CICSVR
 - CICSVRBACKUP keyword [47](#)
 - DFSMSdss [47](#)
- CICSVRBACKUP
 - COPY command [319](#)
 - DUMP command [399](#)
- CICSVRBACKUP keyword
 - COPY command [47](#)
 - DUMP command [47](#)
- class names
 - filter [53](#)
 - saved [53](#)
- class selection, process diagram [496](#)
- CLONE keyword
 - DUMP command [400](#)
- CLOUD
 - DUMP command [400](#)
 - RESTORE command [465](#)
- CLOUD keyword
 - DUMP command [47](#)
- CLOUDCREDENTIALS
 - DUMP command [402](#)
 - RESTORE command [466](#)
- CLOUDUTILS command
 - ALL keyword [279](#)
 - CDACREDSTORE keyword [277](#)
 - CDAPROVIDER keyword [277](#)
 - CLOUD keyword [276](#)
 - CLOUDCREDENTIALS keyword [277](#), [279](#)
 - CONTAINER keyword [277](#)
 - DEBUG keyword [279](#)
 - DELETE keyword [278](#)
 - examples [280](#)
 - explanation [275](#)
 - for DFSMSdss [275](#)
 - FORCE keyword [278](#)
 - LIST keyword [278](#)
 - sample operations [280](#)
 - syntax diagram [275](#)
- CLOUDUTILS, access authorization for [549](#)
- cluster, VSAM, restoring [84](#)
- coexistence considerations [647](#)
- combining
 - combining (*continued*)
 - data set extents [145](#)
 - volume copy and volume dump functions [57](#)
 - command
 - IF-THEN-ELSE [512](#)
 - null [512](#)
 - syntax, general instructions [246](#)
 - commands
 - auxiliary [507](#)
 - DFSMSdss BUILDSA command [517](#)
 - DFSMSdss commands
 - BUILDSA [269](#)
 - CGCREATED [274](#)
 - CLOUDUTILS [275](#)
 - COMPRESS [281](#)
 - CONSOLIDATE [286](#)
 - CONVERTV [295](#)
 - COPY [300](#)
 - COPYDUMP [369](#)
 - DEFRAG [371](#)
 - DUMP [386](#)
 - EOJ [514](#)
 - IF-THEN-ELSE [510](#)
 - PARALLEL [508](#)
 - PRINT [432](#)
 - RELEASE [441](#)
 - RESTORE [451](#)
 - SERIAL [508](#)
 - SET [509](#)
 - SPACEREL [502](#)
 - Writing to the Operator [507](#)
 - WTOR [507](#)
 - DFSMSdss COPYDUMP command [72](#)
 - DFSMSdss DUMP command [71](#)
 - DFSMSdss RESTORE command [72](#)
 - function [267](#)
 - overview [267](#)
- compaction [59](#), [403](#)
- compatibility between DFSMSdss versions [647](#)
- COMPRESS
 - access authorization for [549](#)
 - subkeyword (DUMP command) [403](#), [408](#)
- COMPRESS command
 - ADMINISTRATOR keyword [282](#)
 - BY keyword [282](#)
 - DDNAME keyword [283](#)
 - definition [17](#)
 - DYNALLOC keyword [283](#)
 - DYNAM keyword [283](#)
 - EXCLUDE keyword [283](#)
 - FILTERDD keyword [284](#)
 - for DFSMSdss [281](#)
 - INCLUDE keyword [284](#)
 - module name [36](#)
 - PASSWORD keyword [284](#)
 - PDS (partitioned data set) [144](#)
 - sample operations [285](#)
 - WAIT keyword [285](#)
- concurrent copy
 - backup [45](#), [106](#)
 - CONCURRENT keyword [46](#), [106](#), [404](#)
 - dynamic pacing [349](#)
 - full-volume dump [427](#)
 - initialization [431](#)

- concurrent copy (*continued*)
 - initialization complete – eioption 24 [601](#)
 - Moving Data Sets [106](#)
 - performance considerations [60](#)
 - processing considerations [7](#)
 - serialization handling [7](#)
- CONCURRENT keyword
 - COPY command [106](#), [320](#)
 - DUMP command [46](#), [404](#)
- condition codes
 - general description [508](#)
 - LASTCC [510](#)
 - MAXCC [510](#)
 - setting [508](#)
- conditioned volume
 - copying [99](#)
 - description [57](#)
- consoles supported [21](#)
- CONSOLIDATE command
 - access authorization for [549](#)
 - ADMINistrator keyword [288](#)
 - BY keyword [288](#)
 - CANCELERROR keyword [288](#)
 - DATASET keyword [289](#)
 - DEBUG keyword
 - FRMSG sub-keyword [289](#)
 - TRACE sub-keyword [289](#)
 - description [286](#)
 - diagnosing errors [155](#)
 - DYNALLOC keyword [290](#)
 - example [295](#)
 - EXclude keyword [290](#)
 - FASTREPLICATION keyword [290](#)
 - FCTOPPRCPPRIMARY keyword [291](#)
 - FILTERDD keyword [292](#)
 - for DFSMSdss [286](#)
 - FORCECP keyword [292](#)
 - INCLUDE keyword [292](#)
 - MAXTIME keyword [292](#)
 - PASSWORD keyword [293](#)
 - performance [146](#)
 - PHYSINDDNAME keyword [293](#)
 - PHYSINDYNAM keyword [294](#)
 - PROCESS keyword [294](#)
 - syntax diagram [286](#)
 - using [145](#)
 - WAIT keyword [294](#)
 - WRITECHECK keyword [295](#)
- CONSOLIDATE keyword (DEFRAG command)
 - obsolete [374](#)
- CONSOLIDATE option [146](#)
- contact
 - z/OS [649](#)
- CONTAINER
 - DUMP command [402](#)
 - RESTORE command [467](#)
- container serialization [575](#)
- continuation errors in auxiliary commands [513](#)
- control area [339](#), [473](#)
- control interval [339](#), [473](#)
- controlling DFSMSdss
 - using ISMF [33](#)
 - using JCL [33](#)
- controlling task processing [508](#)
- controlling what DFSMSdss copies [98](#)
- conversion
 - by data movement [17](#)
 - from SMS
 - by data movement [137](#)
 - special requirements [141](#)
 - without data movement [141](#)
 - GDG data set [140](#), [142](#)
 - in an SMS-managed environment [85](#)
 - ineligible data sets [135](#), [136](#)
 - multivolume [140](#), [141](#)
 - preparing a volume [138](#)
 - simulating [137](#)
 - to and from SMS management [17](#), [135](#)
 - to SMS
 - by data movement [136](#)
 - special requirements [139](#)
 - without data movement [138](#)
 - without data movement [17](#), [137](#)
- CONVERT keyword (COPY command) [322](#)
- CONVERTV command
 - CATALOG keyword [296](#)
 - DDNAME keyword [297](#)
 - DEBUG keyword [297](#)
 - DYNAM keyword [297](#)
 - explanation [295](#)
 - for DFSMSdss [295](#)
 - FORCECP keyword [297](#)
 - INCAT keyword [296](#)
 - NONSMS keyword [298](#)
 - PREPARE keyword [298](#)
 - REDETERMINE keyword [298](#)
 - sample operations [299](#)
 - SELECTMULTI keyword [298](#)
 - serialization [568](#), [569](#)
 - SMS keyword [299](#)
 - syntax diagram [295](#)
 - TEST keyword [299](#)
- CONVERTV processing, variables passed to ACS routines [166](#)
- COPY command
 - changing
 - management class [120](#)
 - storage class [120](#)
 - CICSVRBACKUP keyword [47](#)
 - CONCURRENT keyword [106](#)
 - CONVERT keyword
 - partitioned data set [322](#)
 - partitioned data set extended [322](#)
 - data mover selection matrix [104](#)
 - DEBUG keyword [323](#)
 - FILTERDD keyword [338](#)
 - for DFSMSdss [300](#)
 - like devices [15](#)
 - logical processing [25](#)
 - logical volume [125](#)
 - LOGINDYNAM keyword [342](#)
 - module name [36](#)
 - moving data [15](#), [98](#), [113](#)
 - physical volume [125](#)
 - restriction
 - copying an extended-format VSAM data set [115](#)
 - SOURCE subkeyword of TGTALLOC [375](#)
 - sphere restrictions [115](#)
 - unlike devices [15](#)

- COPY command (*continued*)
 - used to make CICSVR backups [47](#)
 - user catalog [112](#)
 - using volume copy
 - for reducing backup time [57](#), [64](#)
 - with volume dump [57](#), [64](#)
 - variables passed to ACS routines [165](#)
 - VSAM data sets [114](#)
- COPY DATASET, data mover selection matrix [104](#)
- COPY FULL
 - freeing the sources, volume [129](#)
- copy operation for logical data set [301](#)
- COPY TRACKS CPVOLUME
 - freeing the sources, volume [129](#)
- COPY, access authorization for [549](#), [554](#)
- COPYDUMP command
 - for DFSMSdss [369](#)
 - INDDNAME keyword [370](#), [410](#)
 - LOGICALVOLUME keyword [370](#)
 - OUTDDNAME keyword [370](#)
 - sample operations [370](#)
 - syntax diagram [369](#), [370](#)
- COPYDUMP command, Linux [177](#)
- COPYDUMP, access authorization for [554](#)
- copying multivolume data sets [113](#)
- copying the Stand-Alone Services core image (example) [273](#)
- COPYVOLID keyword
 - COPY command [322](#)
 - RESTORE command [468](#)
- CPVOLUME keyword
 - COPY command [323](#)
 - DUMP command [405](#)
 - PRINT command [434](#)
 - RESTORE command [468](#)
- CREDIT keyword [29](#), [258](#)
- criteria for filtering [27](#), [255](#)
- critical data sets [41](#)
- Cross-Memory Application Interface
 - controlling DFSMSdss
 - ASPACE parameter [583](#)
 - SNAPX parameter [583](#)
 - SRVRTIME parameter [582](#)
 - return codes [583](#)
- customer program, invocation [47](#), [63](#)
- CYL subkeyword of TGTALLOC
 - COPY command [354](#)
 - RESTORE command [489](#)

D

- damaged PDS, restoring [84](#)
- DASD (direct access storage device)
 - devices supported [21](#)
 - initialized [21](#)
 - reclaiming space [143](#)
 - space fragmentation [146](#)
 - space utilization [146](#)
- DASD (direct access storage device) space fragmentation [371](#)
- dasdfmt utility
 - formatting a Linux for IBM Z volume [170](#)
- DASDVOL
 - access authority [537](#), [542](#), [554](#)
 - authority [534](#), [545](#)
- data class [3](#)
- data compaction
 - hardware [59](#)
 - software [59](#)
- data integrity
 - considerations
 - COPY command [360](#)
 - DUMP command [426](#)
 - RESTORE command [453](#)
 - during data processing [27](#)
 - shared DASD considerations [63](#)
- data movement
 - conversion by [17](#)
 - criteria for [97](#)
 - preparing for [97](#)
 - with FlashCopy [126](#)
 - with native SnapShot [131](#)
- data mover selection matrix for Data Set Copy [104](#)
- data security [153](#)
- data set
 - absolute track [79](#), [81](#)
 - backup [5](#), [39](#), [42](#), [48](#)
 - broken [27](#)
 - changed flag
 - DUMP command [418](#)
 - characteristics (BY criteria) [27](#)
 - converting to multivolume in an SMS-managed environment [85](#)
 - critical [41](#)
 - DEFRAG, special [23](#)
 - direct access [81](#), [82](#), [117](#)
 - dummy [24](#)
 - expiration date handling [103](#)
 - extents, combining [145](#)
 - filtering [25](#), [27](#), [29](#)
 - GDG [88](#), [117](#)
 - HFS [49](#)
 - indexed sequential [81](#)
 - last-used-block pointer [52](#)
 - line operator module names [36](#)
 - logical dump [40](#), [43](#)
 - logical restore [73](#)
 - message [23](#)
 - moving [98](#)
 - multivolume [49](#), [113](#)
 - name
 - fully qualified [28](#), [256](#)
 - partially qualified [28](#), [256](#)
 - non-SMS-managed [89](#)
 - organizations [22](#)
 - partitioned (PDS) [22](#)
 - physical Dump [44](#)
 - physical restore [77](#)
 - preallocated [74](#), [121](#)
 - profiles
 - discrete [537](#), [538](#)
 - generic [537](#)
 - renaming [101](#)
 - restoring
 - in an SMS-managed environment [85](#)
 - multivolume [79](#)
 - to multiple target volumes [79](#)
 - sequential [22](#)
 - SMS-managed [53](#), [86](#), [88](#)

- data set (*continued*)
 - SMS-managed data sets
 - non-VSAM [495](#)
 - physical restore actions [495](#)
 - VSAM [495](#)
 - special requirements [48](#), [111](#)
 - SYS1 system [52](#)
 - system [112](#)
 - temporary copied data set [23](#)
 - temporary names [22](#), [141](#)
 - uncataloged [30](#)
 - undefined DSORG [83](#)
 - unmovable [81](#), [82](#), [116](#)
 - VSAM [52](#)
 - with phantom catalog entries [89](#)
- data set backup
 - CLOUD keyword [47](#)
- data set movement
 - concurrent copy [106](#)
- data-set-changed indicator
 - DFSMSdss handling for a restored data set [77](#), [93](#)
 - example of use [48](#)
 - using for data set backup [29](#), [30](#), [42](#)
 - using for volume backup [40](#)
- DATABASE 2 [22](#)
- DATACLAS keyword [29](#), [259](#)
- DALENGTH keyword (PRINT command) [435](#)
- DATASET keyword
 - CONSOLIDATE command [289](#)
 - COPY command [323](#)
 - data mover selection matrix for copy [104](#)
 - DUMP command [405](#)
 - PRINT command [435](#)
 - RESTORE command [468](#)
 - with logical processing [25](#)
 - with physical data set restore [77](#)
- DB2 (DATABASE 2) data sets [22](#)
- DCB keywords LRECL and BLKSIZE, specifying in a print operation [436](#)
- DDNAME keyword
 - COMPRESS command [283](#)
 - CONVERTV command [297](#)
 - DEFRAG command [374](#), [377](#)
 - RELEASE command [445](#)
 - SPACEREL command [504](#)
- DDNAME list [577](#)
- DDNAME/VOLID Pointer [587](#)
- DDPTR parameter [577](#)
- DEBUG
 - DUMP command [403](#)
- DEBUG keyword
 - CONSOLIDATE command
 - FRMSG sub-keyword [289](#)
 - TRACE sub-keyword [289](#)
 - CONVERTV command [297](#)
 - RESTORE command [469](#)
- DEBUG(FRMSG)
 - FlashCopy, data sets [109](#)
 - FlashCopy, volume [127](#)
 - SnapShot [111](#), [149](#)
 - SnapShot, volume [132](#)
- decrypting data
 - hardware considerations [70](#)
 - software considerations [71](#)
- default block size when dumping to tape or DASD [59](#)
- DEFERRED state [88](#), [118](#)
- DEFERRED subkeyword of TGTGDS
 - COPY command [355](#)
 - RESTORE command [490](#)
- DEFINE function of SAF (system authorization facility) [533](#)
- DEFRAG command
 - access authorization for [554](#)
 - ADMINISTRATOR keyword [374](#)
 - BY keyword [374](#)
 - CONSOLIDATE keyword [374](#)
 - data sets excluded [149](#)
 - DDNAME keyword [375](#)
 - DDNAME subkeyword [374](#)
 - definition [18](#)
 - DYNALOC keyword [376](#)
 - DYNAM keyword [377](#)
 - EXCLUDE keyword [377](#)
 - explanation [371](#)
 - FASTREPLICATION keyword [378](#)
 - for DFSMSdss [371](#)
 - FORCECP keyword [379](#)
 - FRAGMENTATIONINDEX keyword [379](#)
 - general hints [150](#)
 - LIST subkeyword [374](#)
 - MAXMOVE keyword [380](#)
 - MAXTIME keyword [381](#)
 - MMOVPCT keyword [381](#)
 - operation interrupted [23](#)
 - options [149](#)
 - PASSDELAY subkeyword [380](#)
 - PASSWORD keyword [382](#)
 - performance [146](#)
 - sample operations [383](#)
 - serialization [151](#)
 - syntax diagram [373](#)
 - VERIOIN1 keyword [383](#)
 - WAIT keyword [383](#)
 - when to run [146](#)
 - with SnapShot [146](#)
 - WRITECHECK keyword [383](#)
- DEFRAG data set, special [23](#)
- DEFRAG/RACF database [149](#)
- DELETE keyword
 - COPY command [327](#)
 - DUMP command [406](#)
- DELETCATALOGENTRY keyword
 - using [89](#)
- DELETCATALOGENTRY keyword (RESTORE command) [469](#)
- deleting unwanted data sets [144](#)
- description [128](#)
- determining version, release, modification level using the ADRMCLVL macro [625](#)
- Device Support Facilities utility [21](#)
- devices supported [21](#), [93](#)
- DEVTYPE parameter (Stand-Alone TAPECNTL command) [528](#)
- DFM [327](#)
- DFP segment [541](#)
- DFSMSdss
 - access authorization for commands [548](#)
 - backing up Linux for z/Series [169](#)
 - backup and DFSMSshm [7](#)
 - BUILDSA command [517](#)

DFSMSdss (continued)

CICSVR [47](#)

commands

BUILDSA [269](#)

CGCREATED [274](#)

CLOUDUTILS [275](#)

COMPRESS [281](#)

CONSOLIDATE [286](#)

CONVERTV [295](#)

COPY [300](#)

COPYDUMP [369](#)

DEFRAG [371](#)

DUMP [386](#)

EOJ [514](#)

IF-THEN-ELSE [510](#)

PARALLEL [508](#)

PRINT [432](#)

RELEASE [441](#)

RESTORE [451](#)

SERIAL [508](#)

SET [509](#)

SPACEREL [502](#)

Writing to the Operator [507](#)

WTOR [507](#)

compatibility with older versions [647](#)

control [33](#)

COPYDUMP command [72](#)

devices supported [21](#), [93](#)

double encryption [72](#)

filtering [27](#)

function protection [35](#)

interactive [33](#)

invoking [33](#)

invoking with application interface [33](#)

line operators [36](#)

module names [35](#)

module, main entry point [577](#)

overview [3](#)

pointer to address-space-identifier list [578](#)

processing option [585](#)

protecting modules [36](#)

Stand Alone Restore program [517](#)

storage requirements [19](#)

system requirements [19](#)

volume formats supported [21](#)

DFSMSdss commands

BUILDSA [269](#)

CGCREATED [274](#)

CLOUDUTILS [275](#)

COMPRESS [281](#)

CONSOLIDATE [286](#)

CONVERTV [295](#)

COPY [300](#)

COPYDUMP [369](#)

DEFRAG [371](#)

DUMP [386](#)

EOJ [514](#)

IF-THEN-ELSE [510](#)

PARALLEL [508](#)

PRINT [432](#)

RELEASE [441](#)

RESTORE [451](#)

SERIAL [508](#)

SET [509](#)

DFSMSdss commands (continued)

SPACEREL [502](#)

Writing to the Operator [507](#)

WTOR [507](#)

DFSMSdss DUMP NEWNAMEUNCONDITIONAL
description [44](#)

DFSMSshm and DFSMSdss, backup [7](#)

DFSMSstvs

backup-while-open data sets [570](#)

backup-while-open status definitions [572](#)

direct access

data set

moving [117](#)

restoring [82](#)

supported [22](#)

storage device [21](#)

direct access data set

copying [350](#)

restoring [481](#)

direct to cloud [11](#)

direct to cloud limitations [12](#)

disaster recovery [27](#), [39](#), [40](#)

disk track

reading – eioption 07 [592](#)

writing – eioption 08 [592](#)

DO-END group of commands

condition for processing [510](#)

syntax [510](#)

double encryption

processing requests for [71](#)

DSCHA keyword [29](#), [258](#)

DSORG keyword [29](#), [258](#)

DSS cloud solutions [8](#)

dummy data set [24](#), [541](#)

DUMP command

access authorization for [554](#)

ADMINISTRATOR keyword [397](#)

ALLDATA keyword [398](#)

ALLEXCP keyword [398](#)

BY keyword [399](#)

CANCELERROR keyword [399](#)

CDACREDSTORE keyword [401](#)

CDAPROVIDER keyword [401](#), [466](#)

CHECKVTOC keyword [399](#)

CICSVRBACKUP keyword [47](#), [399](#)

CLONE keyword [400](#)

CLOUD keyword [47](#), [400](#)

CLOUDCREDENTIALS keyword [402](#)

COMPRESS command [403](#), [408](#)

CONCURRENT keyword [46](#), [404](#)

CONTAINER keyword [402](#)

CPVOLUME keyword [405](#)

data set changed flag [418](#)

data set example [252](#)

DATASET keyword [405](#)

DEBUG keyword [403](#)

DELETE keyword [406](#)

disaster recovery of logical data sets [40](#)

DUMP PATH keyword [397](#)

DYNALLOC keyword [406](#)

ENQFAILURE subkeyword [421](#)

exceptions to

hardware data compaction [59](#)

software data compaction [59](#)

DUMP command (*continued*)

- EXCLUDE keyword [407](#)
- explanation [386](#)
- FCWITHDRAW keyword [407](#)
- FILTERDD keyword [408](#)
- for DFSMSdss [386](#)
- FORCECP keyword [408](#)
- FULL keyword [408](#)
- INCAT keyword [409](#)
- INCLUDE keyword [409](#)
- INDDNAME keyword [410](#)
- INDYNAM keyword [410](#)
- IOERROR subkeyword [421](#)
- logical data set [43](#)
- logical data set disaster recovery [40](#)
- logical volume [55](#)
- LOGINDDNAME keyword [412](#)
- LOGINDYNAM keyword [412](#)
- module name [36](#)
- NEWNAMEUNCONDITIONAL keyword [414](#)
- non-VSAM data sets [51](#)
- NOTIFYCLONE keyword [414](#)
- NOVALIDATE keyword [415](#), [424](#)
- OBJECTPREFIX keyword [402](#)
- ONLYINCAT keyword [409](#), [415](#)
- OPTIMIZE keyword [59](#), [415](#)
- OUTDDNAME keyword [415](#)
- PASSWORD keyword [416](#)
- PATH keyword [416](#)
- performance considerations [59](#)
- physical data set [44](#)
- physical volume [55](#)
- printed output
 - produced by integrated catalog facility user catalog [51](#)
- PROCESS keyword [417](#)
- PURGE keyword [418](#)
- READIOPACING keyword [418](#)
- RESET keyword [418](#)
- sample operations [426](#)
- SELECTMULTI keyword [342](#), [412](#)
- SHARE keyword [420](#)
- special considerations [387](#)
- SPHERE keyword [420](#)
- STORGRP keyword [421](#)
- syntax diagram [387](#), [390](#)
- SYS1 subkeyword [417](#)
- TOLERATE keyword [421](#)
- TRACKS keyword [422](#)
- UNCATALOG keyword [423](#)
- UNIX files [6](#)
- VALIDATE keyword [52](#), [424](#)
- WAIT keyword [424](#)
- with HFS data sets [564](#)
- WORKINGDIRECTORY keyword [425](#)
- ZCOMPRESS keyword [425](#)
- dump conditioned volume [328](#)
- dump data, multiple copies [369](#)
- DUMPCONDITIONING keyword
 - COPY command [328](#)
 - dump conditioned volume [328](#)
 - using [57](#), [99](#)
- dumping
 - data sets [52](#)

dumping (*continued*)

- efficiently [56](#)
- HFS data sets [49](#)
- indexed VSAM data sets [52](#)
- integrated catalog facility user catalog [51](#)
- multivolume data sets [49](#)
- non-VSAM data sets [51](#)
- SYS1 system data sets [52](#)
- VSAM spheres [52](#)
- DYNALLOC keyword
 - COMPRESS command [283](#)
 - CONSOLIDATE command [290](#)
 - COPY command [328](#)
 - DEFRAG command [376](#)
 - DUMP command [406](#)
 - PRINT command [435](#)
 - RELEASE command [445](#)
 - RESTORE command [471](#)
- DYNALLOC option [149](#)
- DYNAM keyword
 - COMPRESS command [283](#)
 - CONVERTV command [297](#)
 - DEFRAG command [377](#)
 - RELEASE command [445](#)
 - SPACEREL command [505](#)
- dynamic allocation [566](#)

E

- eioption 00 – function startup [589](#)
- eioption 01 – reading SYSIN record [590](#)
- eioption 02 – printing SYSPRINT record [590](#)
- eioption 03 – reading physical tape record [591](#)
- eioption 04 – reading logical tape record [591](#)
- eioption 05 – writing logical tape record [591](#)
- eioption 06 – writing physical tape record [592](#)
- eioption 07 – reading disk track [592](#)
- eioption 08 – writing disk track [592](#)
- eioption 09 –reading utility SYSPRINT [592](#)
- eioption 10 – writing SYSPRINT record [593](#)
- eioption 11 – writing WTO message [593](#)
- eioption 12 – writing WTOR message [593](#)
- eioption 13 – presenting ADRUFO record [594](#)
- eioption 14 – function ending [594](#)
- eioption 15 – presenting WTOR response [594](#)
- eioption 16 – OPEN/EOV tape volume security and verification exit [594](#)
- eioption 17 – OPEN/EOV nonspecific tape volume mount [595](#)
- eioption 18 – insert logical VSAM record during restore [595](#)
- eioption 19 – output tape I/O error [595](#)
- eioption 20 – volume notification [595](#)
- eioption 21– data set verification [596](#)
- eioption 22 – bypass verification exit [596](#)
- eioption 23 – data set processed notification exit [599](#)
- eioption 24 – concurrent copy initialization complete [601](#)
- eioption 25 – backspace physical tape record [602](#)
- eioption 26 – IMS volume notification user exit [603](#)
- eioption 31 – store application metadata object [605](#)
- eioption 32 – retrieve application metadata object exit [605](#)
- eioption 33 – output object notification exit [606](#)
- eioption 35 – cloud bypass verification [606](#)
- eioption 41 – unix file path notification [606](#)
- eioption 42 – UNIX file path bypass notification [607](#)

- eioption 43 – UNIX file path processed notification [607](#)
- eioption 44 – UNIX file path initialization notification [608](#)
- eligibility for conversion [136](#)
- ELSE command [512](#)
- empty non-VSAM data sets [111](#)
- encryption
 - for tape backups [66](#)
 - host-based, for tape backups [67](#)
- END command [512](#)
- ending
 - function – eioption 14 [594](#)
 - your DFSMSdss step [514](#)
- ENDTRK parameter (Stand-Alone RESTORE command) [526](#)
- ENQ keyword [563](#)
- ENQFAILURE subkeyword
 - COPY command [355](#)
 - DUMP command [421](#)
 - PRINT command [437](#)
 - RESTORE command [490](#)
- enqueue
 - compared to DYNALLOC [566](#)
 - exit routine options for non-VSAM data sets [564](#)
 - scheme [563](#)
- enqueue installation exit routine [4](#)
- environment, system [19](#)
- EOJ command
 - for DFSMSdss [514](#)
- EQ operator [29](#), [259](#)
- erase-on-scratch [541](#)
- ERRORTRACKS keyword (PRINT command) [435](#)
- ESDS data sets, supported [22](#)
- EXCLUDE criteria [27](#)
- EXCLUDE keyword
 - CONSOLIDATE command [290](#)
- EXCLUDE keyword
 - COMPRESS command [283](#)
 - COPY command [328](#)
 - criteria [255](#)
 - DEFRAG command [377](#)
 - DUMP command [407](#)
 - RELEASE command [445](#)
 - RESTORE command [471](#)
- EXCP data sets, supported [22](#)
- EXEC statement [248](#)
- EXEC statement PARM information [249](#)
- exit
 - identification block [585](#)
 - interface, structure [584](#)
- exit functions, User Interaction Module [34](#)
- exits, installation [4](#)
- EXPDT keyword [29](#), [258](#)
- expiration date handling [103](#)
- EXPORT/IMPORT (IDCAMS commands), authorization checking [105](#)
- extended-addressable VSAM KSDS [22](#)
- extended-format VSAM data set
 - COPY restriction [115](#)
 - restoring [83](#)
- extended-sequential data set [360](#), [442](#), [494](#)
- extents, combining [145](#)
- EXTNT keyword [29](#), [259](#)

F

- FACILITY class profiles for DFSMSdss keywords [535](#), [542](#)
- FAILRELATION sub-keyword
 - specifying for FCSETGTOK keyword [128](#), [335](#)
- FASTREPLICATION keyword
 - CONSOLIDATE command [290](#)
 - COPY command [329](#)
 - DEFRAG command [378](#)
 - FlashCopy, data sets [108](#)
 - FlashCopy, volume [126](#)
 - native SnapShot [111](#)
 - native SnapShot, volume [131](#)
- FCCGFREEZE keyword
 - COPY command [329](#)
 - FlashCopy relationship, volume [129](#)
 - freezing the source, volume [129](#)
- FCCGVERIFY keyword
 - FlashCopy relationship, volume [129](#)
 - verifying the consistency group, volume [130](#)
- FCFASTREVERSERESTORE keyword
 - COPY command [330](#)
- FCFULLVOLUMERELATION keyword
 - COPY command [331](#)
- FCINCREMENTAL keyword
 - using [64](#)
- FCINCREMENTALLAST keyword
 - using [64](#)
- FCINCRVERIFY keyword
 - using [64](#)
- FCNOCOPY keyword
 - COPY command [331](#), [334](#), [336](#), [338](#)
 - FlashCopy relationship [109](#)
 - FlashCopy relationship, volume [127](#), [129](#)
 - using [57](#), [58](#)
- FCSETGTOK keyword
 - description [335](#)
 - FACILITY class profile [535](#)
 - using space efficient FlashCopy [128](#)
- FCTOPPRCPRIMARY keyword
 - CONSOLIDATE command [291](#)
- FCTOXRCPRIMARY keyword
 - COPY command [337](#)
- FCWAIT keyword
 - using [64](#)
- FCWITHDRAW keyword
 - DUMP command [407](#)
 - freeing subsystem resources [109](#)
 - freeing subsystem resources, volume [127](#)
 - initializing the source volume [129](#)
 - using [57](#), [58](#)
 - withdrawing FlashCopy relationship [109](#)
 - withdrawing FlashCopy relationship, volume [127](#)
- fdasd utility
 - creating Linux partitions [170](#)
- file backup [5](#)
- FILE parameter (Stand-Alone RESTORE command) [527](#)
- filter DD, JCL statement [249](#)
- filter, class names [53](#)
- FILTERDD keyword
 - COMPRESS command [284](#)
 - CONSOLIDATE command [292](#)
 - COPY command [338](#)
 - DUMP command [408](#)

FILTERDD keyword (*continued*)

RELEASE command [446](#)

RESTORE command [472](#)

filtering

BY [263](#)

characteristics [258](#)

COMPRESS [27](#)

data set characteristics [29](#)

examples [30](#)

FILTERDD keyword [30](#)

general description [27](#), [255](#)

logical DUMP [27](#)

logical RESTORE [27](#)

physical DUMP [27](#)

physical RESTORE [27](#)

relative generation [257](#)

RELEASE [27](#)

RESTORE command [73](#)

VSAM data sets [255](#)

FlashCopy

authorization checking [126](#)

backing up and restoring volumes [64](#)

backing up, volume [129](#)

creating consistent copies, volume [129](#)

DEBUG(FRMSG keyword [109](#)

DEBUG(FRMSG keyword, volume [127](#)

FCNOCOPY keyword relationship [109](#)

FCNOCOPY keyword relationship, volume [127](#)

freeing subsystem resources [109](#)

freeing subsystem resources, volume [127](#)

freezing the source, volume [129](#)

in conjunction with

physical full volume copy [57](#), [64](#)

moving data sets with FlashCopy [107](#)

moving data with FlashCopy [16](#)

moving volumes with FlashCopy [126](#)

problem solving [109](#)

problem solving, volume [127](#)

reducing backup time [57](#), [64](#)

thawing the frozen, volume [129](#)

verifying the consistency group, volume [129](#)

withdrawing the system relationship [109](#)

withdrawing the system relationship, volume [127](#)

FORCE keyword

COPY command [338](#)

RESTORE command [472](#)

FORCECP keyword

CONSOLIDATE command [292](#)

CONVERTV command [297](#)

COPY command [339](#)

DEFRAG command [379](#)

DUMP command [408](#)

RESTORE command [472](#)

FRAGI keyword [150](#), [151](#)

fragmentation index [151](#)

FRAGMENTATIONINDEX keyword (DEFRAG command) [379](#)

free-space fragmentation [146](#), [371](#)

FREESPACE keyword

COPY command [339](#)

RESTORE command [472](#)

FROMADDR parameter (Stand-Alone RESTORE command) [525](#)

FROMDEV parameter (Stand--Alone RESTORE command) [525](#)

FSIZE keyword [29](#), [259](#)

FULL keyword

COPY command [339](#)

DUMP command [408](#)

Linux for IBM Z dumps [172](#)

Linux for IBM Z restore [172](#)

physical processing [26](#)

RESTORE command [473](#)

FULL parameter (Stand-Alone RESTORE command) [526](#)

full volume copy

FCNOCOPY keyword [58](#)

FCWITHDRAW keyword [58](#)

process [57](#), [64](#)

using the DUMPCONDITIONING keyword [57](#), [99](#)

fully qualified data set names [28](#), [256](#)

function

ending – eioption 14 [594](#)

startup – eioption 00 [589](#)

G

GDG (generation data group)

data set status assignment [355](#), [490](#)

filtering [257](#)

GDG (generation data group) data set

conversion from SMS [142](#)

conversion to SMS [140](#)

moving [117](#)

restoring [88](#)

GE operator [29](#), [259](#)

generating the Stand-Alone Services program [269](#)

generation data group [257](#)

generation data sets, moving to non-SMS-managed volumes [118](#)

global access checking table [537](#), [538](#)

GT operator [29](#), [259](#)

H

HFS data set

DUMP command considerations [420](#)

logical data set COPY [301](#)

logical dump and serialization [564](#)

physical dumb and serialization [565](#)

read/write serialization scheme [566](#)

RELEASE command consideration [442](#)

HFS data set, dumping [49](#)

host-based encryption

considerations [67](#), [68](#)

double encryption requests

allowing [72](#)

processing [71](#)

examples [67](#), [69](#)

hardware considerations [67](#), [70](#)

ICSF Callable Services

DFSMSdss [67](#)

key management considerations [67](#)

keys and DFSMSdss [68](#)

managing keys [68](#)

performance and hardware types [67](#), [70](#)

securing tape backups [67](#)

software requirements [67](#), [71](#)

types [67](#)

host-based encryption (*continued*)
using compression with [67](#), [69](#)

I

I/O error, output tape – eioption 19 [595](#)
ICKDSF REFORMAT command [271](#)
ICKDSF, initialize DASD volumes with [21](#)
IDCAMS utility [104](#)
IEBCOPY utility [104](#)
IF command, nesting [513](#)
IF-THEN-ELSE command
for DFSMSdss [510](#)
IF-THEN-ELSE group of
commands
command sequencing [512](#)
description [508](#)
examples [513](#)
syntax [510](#)
IGWFAMS utility [104](#)
IMPORT keyword [90](#)
IMPORT keyword (RESTORE command) [473](#), [559](#)
IMS volume notification user exit – eioption 26 [603](#)
INCAT keyword
CONVERTV command [296](#), [298](#)
COPY command [340](#)
DUMP command [409](#)
RELEASE command [446](#)
INCLUDE criteria [27](#)
INCLUDE keyword
COMPRESS command [284](#)
CONSOLIDATE command [292](#)
COPY command [340](#)
criteria [255](#)
DUMP command [409](#)
RELEASE command [446](#)
RESTORE command [474](#)
incremental backup [5](#), [43](#)
incremental FlashCopy
FCINCREMENTAL keyword [64](#)
FCINCREMENTALLAST keyword [64](#)
FCINCRVERIFY keyword [64](#)
FCWAIT keyword [64](#)
INDDNAME keyword
COPY command [340](#)
COPYDUMP command [370](#)
DUMP command [410](#)
physical processing [26](#)
PRINT command [435](#)
RESTORE command [475](#)
INDDNAME parameter (Stand-Alone BUILD SA command)
[270](#)
index, fragmentation [151](#)
indexed sequential data set
restoring [81](#)
indexed VSAM data sets, logical data set dump of [52](#)
indexed VTOCs [21](#)
INDYNAM keyword
COPY command [341](#)
DUMP command [410](#)
physical processing [26](#)
PRINT command [436](#)
INITIAL state [138](#)
INITIAL status [88](#)

initialization
concurrent copy complete – eioption 24 [601](#)
from concurrent copy failure [427](#), [431](#)
initialize all DASD volumes [21](#)
input error toleration
COPY command [355](#)
DUMP command [421](#)
PRINT command [437](#)
input to DFSMSdss (in DD statement) [248](#)
input volumes, specifying [98](#), [100](#)
insert logical VSAM record during restore – eioption 18 [595](#)
installation exit routines [4](#)
integrated catalog facility user catalog
backing up a user catalog, example of [51](#)
dumping [51](#)
printed output for dumping a user catalog [51](#)
printed output for restoring a user catalog [80](#)
restore [80](#)
integrity, data serialization [561](#)
Interactive Storage Management Facility [33](#)
invocation
customer program [47](#)
invocation, from a customer program [63](#)
invoking DFSMSdss
application interface [33](#)
by ATTACH, LINK, or CALL macro [579](#)
from an application program [33](#)
JCL examples [252](#)
using ISMF [33](#)
using JCL [33](#)
invoking ISMF [33](#)
IOERROR subkeyword
COPY command [355](#)
DUMP command [421](#)
PRINT command [437](#)
IPL
tape
rewinding and unloading [528](#)
IPLing Stand-Alone Services on a stand-alone system [520](#)
ISMF (Interactive Storage Management Facility)
display panels [33](#)
function protection [35](#)
invoking [33](#)
line operators [36](#)
logging on [33](#)
menu-driven panels [33](#)
module names [35](#)
online panels [33](#)
PERMIT command [36](#)
protecting modules [36](#)
RDEFINE command [36](#)
use and examples [33](#)
volume list [97](#)

J

JCL
examples [18](#)
JCL (job control language)
concurrent copy [427](#), [431](#)
EXEC statement [248](#)
filter DD statement [249](#)
input DD statement [248](#)
invoking DFSMSdss [33](#), [252](#)

- JCL (job control language) (*continued*)
 - JOB statement [248](#)
 - output DD statement [249](#)
 - password DD statement [249](#)
 - requirements for DFSMSdss [248](#)
 - restore integrated catalog facility user catalog [80](#)
 - SYSIN DD statement [248](#)
 - SYSPRINT DD statement [248](#)
 - volume count subparameter [249](#)
- job control language [33](#), [248](#)
- JOB statement JCL [248](#)
- JOBCAT DD statement, JCL [78](#)

K

- key sequenced data sets, supported [22](#)
- keyboard
 - navigation [649](#)
 - PF keys [649](#)
 - shortcut keys [649](#)
- KEYLENGTH keyword (PRINT command) [436](#)
- Keys [13](#)
- keyword
 - module protection [37](#)
 - preallocated targets [123](#)
 - profile names [37](#)
- keywords [14](#)
- KSDS, supported [22](#)

L

- LASTCC
 - definition [509](#)
 - in IF-THEN-ELSE command sequence [511](#)
 - in SET command [510](#)
- LDS, supported [22](#)
- LE operator [29](#), [259](#)
- like devices, moving volumes to [132](#)
- line operators, DFSMSdss/ISMF [36](#)
- LINECNT keyword [250](#)
- Linux copy
 - COPYDUMP command [177](#)
 - full volume [177](#)
- Linux dumps
 - ALLEXCP keyword [172](#)
 - DATASET keyword [172](#)
 - FULL keyword [172](#)
 - using concurrent copy [172](#)
- Linux for IBM Z
 - authorization [171](#)
 - backing up with partitions [169](#)
 - DFSMSdss command
 - BUILD STAND-ALONE [169](#)
 - COPY FULL [169](#)
 - COPY FULL COPYVOLID ALLEXCP [169](#)
 - COPYDUMP [169](#)
 - DUMP DATASET [169](#)
 - DUMP FULL [169](#)
 - DUMP FULL with CONCURRENT COPY [169](#)
 - RESTORE DATASET [169](#)
 - RESTORE FULL [169](#)
 - disk utility

- Linux for IBM Z (*continued*)
 - disk utility (*continued*)
 - dasdfmt [170](#)
 - fdasd [170](#)
 - hardware environment [169](#)
 - restoring partitions [169](#)
 - submitting JCL batch jobs to z/OS
 - using FTP [177](#)
 - volume serial rules [170](#)
 - LIST subkeyword
 - DEFRAG command [374](#), [377](#)
 - location-dependent data [74](#), [82](#)
 - locking a user catalog [80](#)
 - lockout, avoiding [563](#)
 - LOGDDNAME keyword (RELEASE command) [447](#)
 - logging on to ISMF [33](#)
 - logical
 - tape record, reading – eioption 04 [591](#)
 - tape record, writing – eioption 05 [591](#)
 - VSAM record, insert during restore – eioption 18 [595](#)
 - logical COPY [97](#), [125](#)
 - logical data set dump of indexed VSAM data sets [52](#)
 - logical data set restore [73](#)
 - logical dump volume [55](#)
 - logical processing [25](#)
 - logical restore
 - cataloging data sets [76](#)
 - data sets with phantom catalog entries [89](#)
 - preformatted empty VSAM data set [90](#)
 - renaming data sets [76](#)
 - user catalog aliases [80](#)
 - without preallocated targets [82](#)
 - LOGICALVOLUME keyword
 - COPYDUMP command [370](#)
 - RESTORE command [475](#)
 - LOGINDDNAME keyword
 - COPY command [341](#)
 - DUMP command [412](#)
 - logical processing [25](#)
 - LOGINDYNAM keyword
 - COPY command [342](#)
 - DUMP command [412](#)
 - logical processing [25](#)
 - LRECL and BLKSIZE keywords, specifying in a print operation [436](#)
 - LT operator [29](#), [259](#)

M

- magnetic tape devices supported [21](#)
- maintenance [580](#)
- MAKEMULTI keyword
 - COPY command [343](#)
 - RESTORE command [475](#)
- management class
 - changing with copy [120](#)
 - changing with RESTORE [87](#)
- Managing backups with CLOUDUTILS [15](#)
- mapping macro, ADREIDO [609](#)
- MAXCC keyword
 - definition [509](#)
 - in IF-THEN-ELSE command sequence [511](#)

MAXCC keyword (*continued*)
 in SET command [510](#)
 maximize track utilization by reblocking [153](#)
 MAXMOVE keyword [150](#)
 MAXMOVE keyword (DEFRAG command) [380](#)
 MAXSIZE variable [167](#)
 MAXTIME keyword
 CONSOLIDATE command [292](#)
 MAXTIME keyword (DEFRAG command) [381](#)
 MENTITY keyword
 COPY command [343](#)
 RESTORE command [476](#)
 message data set [23](#), [539](#)
 MGMTCLAS keyword
 COPY command [344](#)
 RESTORE command [476](#)
 MGMTCLAS profile [259](#), [541](#)
 MINCYLs keyword
 SPACEREL command [505](#)
 minivolumes [21](#)
 MINSECQTY keyword [143](#)
 MINSECQTY keyword (RELEASE command) [448](#)
 MINTRACKSUNUSED keyword [143](#)
 MINTRACKSUNUSED keyword (RELEASE command) [448](#)
 MMOVPCT keyword (DEFRAG command) [381](#)
 MMOVPCT option [150](#)
 mode switching [508](#)
 module names for data set application commands [36](#)
 moving
 catalogs [112](#)
 damaged PDS [116](#)
 data in an SMS-managed environment [16](#)
 data sets
 preformatted empty VSAM [124](#)
 special requirements [111](#)
 utilities [104](#)
 data sets to unlike devices [117](#)
 direct access data set [117](#)
 empty non-VSAM data sets [111](#)
 generation data sets
 to non-SMS-managed volumes [118](#)
 to SMS-managed volumes [118](#)
 multivolume data set [113](#)
 non-VSAM data sets that have aliases [113](#)
 PDSE [116](#)
 SMS-managed data sets [119](#)
 system data sets [112](#)
 to preallocated data set [121](#)
 undefined DSORG data sets [111](#)
 unmovable data sets [116](#)
 volumes
 logical volumes [125](#)
 physical volumes [125](#)
 to devices of equal capacity [132](#)
 to devices of greater capacity [132](#)
 to unlike devices [132](#)
 VM-format volumes [133](#)
 VTOC considerations [124](#)
 moving data
 with concurrent copy [16](#)
 with FlashCopy [16](#)
 with SnapShot [16](#)
 moving data sets [252](#), [327](#)
 moving volumes
 with FlashCopy [126](#)
 with native SnapShot [131](#)
 MULTI keyword [29](#), [259](#)
 multiple backup copies [369](#), [416](#)
 multiple subkeywords [246](#)
 multiple target volumes
 restoring [79](#)
 specifying [123](#)
 multiple tracks read, on dump [415](#)
 multivolume data set
 conversion [140](#), [141](#)
 copying [114](#)
 discrete profile [541](#)
 dumping [49](#)
 restore [79](#)
 restoring [114](#)
 multivolume VSAM data sets [85](#), [114](#)
 MVOLSER keyword
 COPY command [343](#)
 RESTORE command [476](#)
 MVS environments supported [19](#)

N

names, data set [28](#)
 navigation
 keyboard [649](#)
 NE operator [29](#), [259](#)
 nested IF commands [513](#)
 NEWNAMEUNCONDITIONAL keyword
 description [44](#)
 DUMP command [414](#)
 FACILITY class profile [535](#)
 non-SMS authorization [547](#)
 non-SMS-managed data sets, restoring [89](#)
 non-SMS-managed targets, special considerations [142](#)
 non-SMS-managed volumes, moving generation data sets to [118](#)
 Non-VSAM Data Set Erase Table [153](#)
 non-VSAM data sets that have aliases [51](#), [81](#), [113](#)
 non-VSAM data sets, converting to multivolume [85](#)
 non-VSAM data sets, restore actions [495](#)
 non-VSAM preallocation [122](#)
 nonindexed VTOC [21](#)
 NONSMS keyword (CONVERTV command) [298](#)
 NOPACKING keyword
 COPY command [344](#)
 RESTORE command [476](#)
 restoring to preallocated target data sets [75](#)
 to restore damaged partitioned data sets [85](#)
 with preallocated target [123](#)
 NOREADCHECK parameter (Stand-Alone RESTORE command) [526](#)
 NORUN keyword [250](#)
 notification
 data set processed exit – eioption 23 [599](#)
 data set processed exit – eioption 31 [605](#)
 data set processed exit – eioption 32 [605](#)
 data set processed exit – eioption 33 [606](#)
 IMS volume user exit – eioption 26 [603](#)
 volume – eioption 20 [595](#)
 NOTIFYCLONE keyword

NOTIFYCLONE keyword (*continued*)

DUMP command [414](#)

NOTIFYCONCURRENT keyword [301](#), [320](#)

NOVALIDATE keyword (DUMP command) [415](#), [424](#)

NOVERIFY parameter (Stand-Alone RESTORE command) [526](#)

null command [512](#)

NULLMGMTCLAS keyword

COPY command [344](#)

RESTORE command [476](#)

NULLSTORCLAS keyword

COPY command [353](#)

RESTORE command [489](#)

O

object serialization [575](#)

object storage cloud [12](#)

OBJECTPREFIX

DUMP command [402](#)

RESTORE command [467](#)

ONLYINCAT keyword

COPY command [340](#)

DUMP command [415](#)

RELEASE command [446](#)

OPEN/EOV

nonspecific tape volume mount – eioption 17 [595](#)

tape volume security and verification exit – eioption 16 [594](#)

operating environment [19](#)

operating in serial or parallel mode [508](#)

OPERATIONS attribute [533](#), [534](#)

OPERCNSL parameter (Stand-Alone BUILDSEA command)

limited validation of [516](#)

purpose of [516](#)

specifying [271](#)

OPTIMIZE keyword [59](#)

OPTIMIZE keyword (DUMP command) [415](#)

option list [577](#)

optional parameters [270](#)

options for non-VSAM data sets, enqueue exit routine [564](#)

options installation exit routine [4](#)

OPTPTR parameter [577](#)

organizations, data set [22](#)

original record length [587](#)

original record pointer [587](#)

OUTDDNAME keyword

COPY command [345](#)

COPYDUMP command [370](#)

DUMP command [415](#)

PRINT command [436](#)

RESTORE command [477](#)

OUTDYNAM keyword

COPY command [345](#)

RESTORE command [477](#)

OUTDYNAM parameter (Stand-Alone BUILDSEA command) [272](#)

OUTDYNAM keyword [74](#), [100](#)

output from DFSMSdss (in DD statement) [249](#)

output tape I/O error – eioption 19 [595](#)

output volume

selecting [100](#)

specifying [74](#), [78](#), [93](#)

OUTTRACKS keyword

COPY command [346](#)

RESTORE command [478](#)

overview of DFSMSdss [3](#)

overview of the Stand-Alone Services process [519](#)

P

page ejection, suppressing [250](#)

page-number list [578](#)

PAGENO keyword [250](#)

PAGEPTR parameter [578](#)

PARALLEL command

for DFSMSdss [508](#)

PARALLEL keyword [59](#)

PARAM keyword [578](#)

PARM information in the EXEC statement [249](#)

partially qualified data set names [28](#), [256](#)

partitioned data set [20](#)

partitioned data set extended [20](#)

PASSDELAY option [150](#)

PASSDELAY subkeyword (DEFRA command) [380](#)

password DD, JCL statement [249](#)

PASSWORD keyword

COMPRESS command [284](#)

CONSOLIDATE command [293](#)

COPY command [346](#)

DEFRA command [382](#)

DUMP command [416](#)

PRINT command [436](#)

RELEASE command [449](#)

RESTORE command [478](#)

password protection [536](#)

PATCH parameter definition [508](#)

PATH keyword

DUMP command [416](#)

RESTORE command [479](#)

PDS (partitioned data set)

abnormal conditions [84](#)

compressing [144](#)

monitoring PDSs during compression [116](#)

moving damaged [116](#)

prevention of [116](#)

repairs by DFSMSdss during compression [116](#)

restoring, damaged [84](#)

storage requirements [20](#)

supported [22](#)

translation during compression [116](#)

PDS subkeyword (COPY command) [322](#)

PDSE (partitioned data set extended)

moving [116](#)

restoring [84](#)

supported [22](#)

PDSE subkeyword (COPY command) [322](#)

PERCENTUTILIZED keyword

COPY command [347](#)

RESTORE command [479](#)

with a logical data set restore operation [74](#)

with preallocated target [123](#)

performance

concurrent copy [60](#)

CONSOLIDATE command [146](#)

DEFRA command [146](#)

DUMP [59](#)

- performance (*continued*)
 - read DASD I/O pacing [63](#)
- performing a restore from dump tapes [524](#)
- PERMIT command [36](#)
- phantom catalog entries [89](#)
- physical copy [97](#), [125](#)
- physical data set copy
 - performing [99](#)
 - using a conditioned volume [99](#)
- physical full volume copy
 - FCINCREMENTAL keyword [64](#)
 - FCINCREMENTALLAST keyword [64](#)
 - FCINCRVERIFY keyword [64](#)
 - FCWAIT keyword [64](#)
- physical processing [25](#), [26](#)
- physical restore
 - INITIAL status [88](#)
 - output volume selection [78](#)
- physical restore of SMS-managed data sets [88](#)
- physical tape record
 - backspace – eioption 25 [602](#)
 - reading – eioption 03 [591](#)
 - writing – eioption 06 [592](#)
- physical volume dump [55](#)
- PHYSINDD [417](#)
- PHYSINDDNAME keyword
 - CONSOLIDATE command [293](#)
- PHYSINDYNAM [417](#)
- PHYSINDYNAM keyword
 - CONSOLIDATE command [294](#)
- planning an availability strategy [39](#)
- preallocated data set
 - COPY command [121](#)
 - moving to [121](#)
 - REPLACE keyword [74](#)
 - REPLACEUNCONDITIONAL keyword [74](#)
 - restoring to [74](#)
- preallocated targets
 - restoring [82](#)
 - restoring without [82](#)
- preallocation
 - non-VSAM [122](#)
 - VSAM [121](#)
- PREPARE keyword [17](#), [138](#)
- PREPARE keyword (CONVERTV command) [298](#)
- preparing for Stand-Alone Services [267](#), [529](#)
- presenting
 - ADRUFO record – eioption 13 [594](#)
 - WTOR response – eioption 15 [594](#)
- PRINT command
 - ADMINISTRATOR keyword [434](#)
 - ALLDATA keyword [434](#)
 - CPVOLUME keyword [434](#)
 - DATALENGTH keyword [435](#)
 - DATASET keyword [435](#)
 - DYNALLOC keyword [435](#)
 - ENQFAILURE subkeyword [437](#)
 - ERRORTRACKS keyword [435](#)
 - explanation [432](#)
 - for DFSMSdss [432](#)
 - INDDNAME keyword [435](#)
 - INDYNAM keyword [436](#)
 - IOERROR subkeyword [437](#)
 - KEYLENGTH keyword [436](#)

- PRINT command (*continued*)
 - OUTDDNAME keyword [436](#)
 - PASSWORD keyword [436](#)
 - sample operations [439](#)
 - SHARE keyword [437](#)
 - syntax diagram [433](#)
 - TOLERATE keyword [437](#)
 - TRACKS keyword [438](#)
 - VTOC keyword [439](#)
 - WAIT keyword [439](#)
- PRINT, access authorization for [555](#)
- printers supported [21](#)
- printing SYSPRINT record – eioption 02 [590](#)
- PROCESS keyword
 - CONSOLIDATE command [294](#)
 - COPY command [348](#)
 - DUMP command [417](#)
 - RELEASE command [449](#)
 - RESTORE command [480](#)
- processing
 - logical [25](#)
 - physical [25–27](#)
- protect-all [539](#)
- protected
 - catalogs [547](#)
 - group data set [537](#)
 - user data set [537](#)
- protecting
 - data sets [535](#)
 - passwords [536](#)
 - resources [535](#)
 - SMS-managed data sets [541](#)
 - usage of DFSMSdss [535](#)
- protection
 - DFSMSdss function [35](#)
 - DFSMSdss/ISMF modules [36](#)
 - functions with RACF [35](#)
 - ISMF access [35](#)
 - keywords with RACF [37](#)
- provider file [12](#)
- PURGE keyword
 - COPY command [348](#)
 - DUMP command [418](#)
 - RESTORE command [480](#)

Q

- qualified data set names [256](#)

R

- RACF (Resource Access Control Facility)
 - authorization checking [343](#), [476](#)
 - database [149](#)
 - DEFINE function [533](#)
 - keyword profiles [37](#)
 - logging [542](#)
 - protecting keyword modules [37](#)
 - protecting keywords [37](#)
- RACF-indicated, meaning of [538](#)
- RACF-protected, meaning of [533](#)
- RAMAC Virtual Array (RVA) [16](#), [46](#), [61](#), [110](#)
- RDEFINE command [36](#)

- read I/O pacing, performance considerations [63](#)
- READCHECK parameter (Stand-Alone RESTORE command) [526](#)
- readers, card, supported [21](#)
- reading
 - disk track – eioption 07 [592](#)
 - logical tape record – eioption 04 [591](#)
 - physical tape record – eioption 03 [591](#)
 - SYSIN record – eioption 01 [590](#)
 - utility SYSPRINT – eioption 09 [592](#)
- READIOPACING keyword
 - COPY command [349](#)
 - DUMP command [418](#)
- reblock installation exit routine [4](#)
- REBLOCK keyword
 - COPY command [349](#)
 - RESTORE command [480](#)
- REBLOCK keyword, with preallocated target [124](#)
- REBLOCK processing
 - determining block size [153](#)
 - track usage [153](#)
- RECATALOG keyword
 - COPY command [318](#)
 - during logical restore processing [76](#)
 - RESTORE command [465](#)
 - with preallocated target [123](#)
- recataloging an alternate index [319](#)
- reclaiming DASD space [143](#)
- record area length [587](#)
- record counting for copy, dump, and restore [4](#)
- record level sharing
 - backing up data sets [53](#)
 - copy operation [124](#)
 - moving data sets [124](#)
 - time out protection [53](#)
- records past last-used-block pointer, dumping [52](#)
- recovery
 - disaster [40](#)
 - SMS-managed environment [40](#)
 - system volumes [94](#)
 - user catalog [80](#)
- REDETERMINE keyword [139](#)
- REDETERMINE keyword (CONVERTV command) [298](#)
- REFDT keyword [29](#), [259](#)
- refreshing volumes
 - with incremental FlashCopy [64](#)
- relative generation filtering [257](#)
- RELBLOCKADDRESS keyword
 - COPY command [350](#)
 - RESTORE command [481](#)
- RELEASE command
 - ADMINISTRATOR keyword [444](#)
 - BY keyword [444](#)
 - cross-system serialization [445](#)
 - DDNAME keyword [445](#)
 - definition [17](#), [441](#)
 - DYNALOC keyword [445](#)
 - DYNAM keyword [445](#)
 - EXCLUDE keyword [445](#)
 - FILTERDD keyword [446](#)
 - for DFSMSdss [441](#)
 - INCAT keyword [446](#)
 - INCLUDE keyword [446](#)
 - LOGDDNAME keyword [447](#)
- RELEASE command (*continued*)
 - MINSECQTY keyword [448](#)
 - MINTRACKSUNUSED keyword [448](#)
 - module name [36](#)
 - ONLYINCAT keyword [446](#)
 - PASSWORD keyword [449](#)
 - PROCESS keyword [449](#)
 - sample operations [451](#)
 - SPHERE keyword [449](#)
 - STORGRP keyword [450](#)
 - syntax diagram [442](#)
 - SYS1 subkeyword [449](#)
 - unused space in data sets [143](#)
 - WAIT keyword [450](#)
- RELEASE, access authorization for [555](#)
- remote site [40](#)
- RENAME keyword (RESTORE command) [481](#)
- RENAME keyword, during logical restore processing [76](#)
- RENAMEUNCONDITIONAL keyword
 - COPY command [350](#)
 - RESTORE command [483](#)
 - with COPY command [101](#)
 - with preallocated target [124](#)
- renaming data sets [101](#)
- REPLACE keyword
 - COPY command [351](#)
 - moving to preallocated target data sets [121](#)
 - moving VSAM data sets [114](#)
 - preallocated data set [74](#)
 - RESTORE command [484](#)
 - SMS-managed data set [86](#)
 - unmovable data set [82](#)
- REPLACEUNCONDITIONAL keyword
 - COPY command [352](#)
 - moving to preallocated target data sets [121](#)
 - moving VSAM data sets [114](#)
 - preallocated data set [74](#)
 - RESTORE command [352](#), [486](#)
 - SMS-managed data sets [86](#)
 - unmovable data set [82](#)
- REPRO (IDCAMS command) [104](#)
- requirements
 - real storage for Stand-Alone Services program operation [515](#)
- RESERVE macro [562](#)
- RESET keyword
 - DUMP command [418](#)
 - RESTORE command [487](#)
- Resource Access Control Facility [37](#), [533](#)
- resource serialization [570](#)
- RESOWNER field [541](#)
- RESTORE command
 - absolute track data set [81](#)
 - access authorization for [555](#)
 - actions on non-VSAM data sets [495](#)
 - ACTIVE subkeyword of TGTGDS [490](#)
 - ADMINISTRATOR keyword [462](#)
 - AUTORELBLOCKADDRESS keyword [462](#)
 - BCSRECOVER keyword [463](#)
 - BLK subkeyword of TGTALLOCC [489](#)
 - BY keyword [463](#)
 - BYPASSACS keyword [464](#)
 - CANCELERROR keyword [464](#)
 - CATALOG keyword [465](#)

RESTORE command (*continued*)

- CDACOMPRESS keyword [401](#)
- CDACREDSTORE keyword [466](#)
- changing management class [87](#)
- changing storage class [86](#)
- CLOUD keyword [465](#)
- CLOUDCREDENTIALS keyword [466](#)
- CONTAINER keyword [467](#)
- COPYVOLID keyword [468](#)
- CPVOLUME keyword [468](#)
- CYL subkeyword of TGTALLOC [489](#)
- data set example [252](#)
- data sets with phantom catalog entries [89](#)
- DATASET keyword [468](#)
- DEBUG keyword [469](#)
- DEFERRED subkeyword of TGTGDS [490](#)
- DELETETOCATALOGENTRY keyword [469](#)
- direct data sets [81](#)
- DYNALLOC keyword [471](#)
- ENQFAILURE subkeyword [490](#)
- EXCLUDE keyword [471](#)
- FILTERDD keyword [472](#)
- filtering [73](#)
- for DFSMSdss [451](#)
- FORCE keyword [472](#)
- FORCECP keyword [472](#)
- FREESPACE keyword [472](#)
- FULL keyword [473](#)
- IMPORT keyword [473](#), [559](#)
- INCLUDE keyword [474](#)
- INDDNAME keyword [475](#)
- indexed sequential data sets [81](#)
- logical [73](#)
- LOGICALVOLUME keyword [475](#)
- MAKEMULTI keyword [475](#)
- MENTITY keyword [476](#)
- MGMTCLAS keyword [476](#)
- module name [36](#)
- MVOLSER keyword [476](#)
- non-VSAM data sets [81](#)
- NOPACKING keyword [476](#)
- NULLMGMTCLAS keyword [476](#)
- NULLSTORCLAS keyword [489](#)
- OBJECTPREFIX keyword [467](#)
- OUTDDNAME keyword [477](#)
- OUTDYNAM keyword [477](#)
- OUTTRACKS keyword [478](#)
- PASSWORD keyword [478](#)
- PATH keyword [479](#)
- PATH syntax [462](#)
- PERCENTUTILIZED keyword [479](#)
- physical [73](#), [77](#)
- preallocated target data sets [74](#)
- printed output
 - produced by integrated catalog facility user catalog [80](#)
- PROCESS keyword [480](#)
- PURGE keyword [480](#)
- REBLOCK keyword [480](#)
- RECATALOG keyword [465](#)
- RELBLOCKADDRESS keyword [481](#)
- RENAME keyword [481](#)
- RENAMEUNCONDITIONAL keyword [483](#)
- REPLACE keyword [484](#)

RESTORE command (*continued*)

- REPLACEUNCONDITIONAL keyword [352](#), [486](#)
- RESET keyword [487](#)
- ROLLEDOFF subkeyword of TGTGDS [490](#)
- SAM compressed data set [89](#)
- sample operations [497](#)
- SHARE keyword [487](#)
- SKIPPARNTDIRS keyword [488](#)
- SOURCE subkeyword of TGTALLOC [489](#)
- SOURCE subkeyword of TGTGDS [490](#)
- special considerations [452](#)
- SPHERE keyword [488](#)
- STORCLAS keyword [489](#)
- TGTALLOC keyword [489](#)
- TGTGDS keyword [490](#)
- TOLERATE keyword [490](#)
- TRACKS keyword [490](#)
- TRK subkeyword of TGTALLOC [489](#)
- TTRADDRESS keyword [491](#)
- UNDEFINEDSORG subkeyword [480](#)
- UNIX files [6](#)
- unmovable data sets [81](#)
- variables passed to ACS routines [166](#)
- VOLCOUNT keyword [492](#)
- WAIT keyword [493](#)
- WORKINGDIRECTORY keyword [494](#)
- WRITECHECK keyword [494](#)
- RESTORE FULL command
 - syntax diagram [454](#)
- restore operation
 - data set
 - key-sequenced with no secondary allocation [452](#)
 - partitioned with secondary allocation of zero [452](#)
 - RACF-protected VSAM, logical restore of [452](#)
 - RENAME, RENAMEUNCONDITIONAL, REPLACE keywords [452](#)
- RESTORE TRACKS command
 - syntax diagram [455](#)
- restoring
 - backups created with an earlier release [647](#)
 - damaged PDS [84](#)
 - data sets [73](#)
 - direct access data sets [82](#)
 - extended-format VSAM data set
 - stripe count of one [83](#)
 - GDG data sets [88](#)
 - in an SMS-managed environment [86](#)
 - multivolume data sets [114](#)
 - non-SMS-managed data set [89](#)
 - PDSE [84](#)
 - preallocated targets [74](#), [82](#)
 - preformatted empty VSAM data set [90](#)
 - SMS-managed data sets [86](#), [88](#)
 - SMS-managed GDG data sets [88](#)
 - undefined DSORG data set [83](#)
 - volumes [92](#)
 - VSAM cluster [84](#)
 - VSAM sphere [83](#)
 - without preallocated targets [82](#)
- restoring volumes
 - with incremental FlashCopy [64](#)
- restriction
 - copying an extended-format VSAM data set
 - stripe count of one [115](#)

- REWIND parameter (Stand-Alone TAPECNTL command) [529](#)
- RIOP keyword [301](#), [308](#), [311](#)
- RLS quiesce processing [53](#), [124](#)
- ROLLED-OFF state [88](#), [118](#)
- ROLLEDOFF subkeyword of TGTGDS
 - COPY command [355](#)
 - RESTORE command [490](#)
- RRDS data sets, supported [22](#)
- running Stand-Alone Services
 - in XA or ESA mode [515](#)
 - program [519](#)
 - under VM [515](#)
 - with a predefined console [516](#)
- RVA [320](#)
- RVA (RAMAC Virtual Array) [16](#), [46](#), [61](#), [110](#)

S

- SADMP service aid [523](#)
- SAF (system authorization facility) [533](#)
- SAM compressed data set, copying [301](#)
- Sample provider file [12](#)
- SCAN keyword [250](#), [251](#)
- SDM (system data mover) [61](#)
- SDUMP keyword [250](#)
- search order, standard catalog [263](#)
- security
 - for tape backups [66](#)
- SELECTMULTI keyword
 - CONVERTV command [298](#)
 - DUMP command [342](#), [412](#)
 - with backup function [50](#)
 - with conversion function [140](#), [141](#)
 - with COPY DATASET function [100](#)
 - with COPY function [25](#), [113](#)
- separator, definition [245](#)
- sequential data sets, supported [22](#)
- sequential data striping
 - extended sequential [4](#)
 - VSAM data sets [4](#)
- sequential extended-format data sets
 - processing [4](#)
 - support for [4](#)
- SERIAL command
 - for DFSMSdss [508](#)
- serialization
 - avoiding lockout [563](#)
 - concurrent copy [7](#)
 - data integrity [561](#)
 - data set [563](#)
 - DYNALLOC keyword [566](#)
 - ENQ keyword [563](#)
 - objects [575](#)
 - Read/Write scheme [566](#)
 - RESERVE macro [562](#)
 - resource [570](#)
 - volume [562](#)
 - WAIT option [563](#), [569](#)
- service [580](#)
- SET command
 - for DFSMSdss [509](#)
 - sample operations [510](#)
- setting
 - condition codes [508](#)
 - setting (*continued*)
 - patch bytes with SET PATCH command [509](#)
- SHARE keyword
 - COPY command [353](#)
 - DUMP command [420](#)
 - PRINT command [437](#)
 - RESTORE command [487](#)
- SHARE keyword, with backup function for HFS [49](#)
- shortcut keys [649](#)
- simulating conversion [137](#)
- SIZE keyword [250](#)
- SIZE variable [167](#)
- SKIPPARENTDIRS keyword
 - RESTORE command [488](#)
- SMS
 - authorization [547](#)
 - keyword (CONVERTV command) [299](#)
 - managed data sets [541](#)
- SMS Cloud Network Connections [14](#)
- SMS conversion
 - eligibility [136](#)
 - ineligibility [135](#), [136](#)
- SMS management
 - conversion by data movement [136](#)
 - conversion from, by data movement [137](#)
 - conversion to and from [17](#)
 - conversion without data movement [138](#)
 - converting from, without data movement [141](#)
 - converting to and from [135](#)
- SMS Report [139](#)
- SMS-managed data sets
 - backup [53](#)
 - moving [119](#)
 - physical RESTORE [88](#)
 - restoring [86](#)
 - restoring, GDG data sets [88](#)
- SMS-managed environment
 - backup [40](#)
 - moving data [16](#)
 - Non-SMS-managed data [40](#)
 - recovery [40](#)
 - restoring data sets [85](#)
 - SMS-managed data [40](#)
- SMS-managed volumes, moving data sets to [118](#)
- SMSGCNT keyword [250](#)
- SnapShot
 - authorization checking [131](#)
 - DEBUG(FRMSG keyword [111](#), [149](#))
 - DEBUG(FRMSG keyword, volume [132](#))
 - in conjunction with
 - physical full volume copy [57](#)
 - moving data sets with SnapShot [110](#)
 - moving volumes with native SnapShot [131](#)
 - native mode [16](#), [110](#), [111](#)
 - problem solving [111](#), [149](#)
 - problem solving, volume [132](#)
 - RAMAC Virtual Array (RVA) [46](#), [106](#)
 - reducing backup time [57](#), [86](#)
 - virtual concurrent copy [46](#), [86](#)
 - virtual concurrent copy mode [46](#), [106](#), [111](#)
- SOURCE subkeyword of TGTALLOC
 - COPY command [355](#)
 - RESTORE command [489](#), [490](#)

- SOURCE subkeyword of TGTGDS
 - COPY command [354](#)
- source, definition [248](#)
- space
 - considerations [59](#)
 - fragmentation on DASD [146](#)
 - management [17](#)
- space allocation [354](#), [489](#)
- space efficient FlashCopy [128](#), [231](#)
- space efficient volume [128](#)
- space fragmentation on DASD [371](#)
- space utilization, percent
 - COPY command [347](#)
 - RESTORE command [479](#)
- SPACEREL command
 - DDNAME keyword [504](#)
 - DEBUG keyword [504](#)
 - definition [18](#)
 - DYNAM keyword [505](#)
 - for DFSMSdss [502](#)
- SPECIAL attribute [533](#)
- special considerations for non-SMS-managed target [142](#)
- special requirements
 - conversion from SMS [141](#)
 - conversion to SMS [139](#)
 - data set backup [48](#)
 - moving data sets [111](#)
 - restoring data sets [79](#)
- specified operating environment [19](#)
- sphere eligibility [139](#)
- SPHERE keyword
 - COPY command [353](#)
 - DUMP command [420](#)
 - RELEASE command [449](#)
 - RESTORE command [488](#)
- sphere processing, VSAM eligibility [139](#)
- spheres
 - dumping VSAM [52](#)
 - restoring VSAM [83](#)
- stand-alone restore program
 - overview [8](#)
 - physical processing [27](#)
 - system volumes, backing up [27](#)
 - system volumes, recovering [94](#)
 - used to restore Linux volumes [177](#)
 - VM, running on [94](#)
 - volume dump, physical [56](#)
- Stand-Alone Services feature
 - BUILDSA command
 - ADMINISTRATOR parameter [530](#)
 - determining authorization level [529](#)
 - examples [272](#)
 - INDDNAME parameter [270](#)
 - IPL parameter [271](#)
 - OPERCNSL parameter [271](#)
 - OUTDDNAME parameter [272](#)
 - OUTDYNAM parameter [272](#)
 - required parameters [270](#)
 - syntax [270](#)
 - commands
 - BUILDSA [269](#)
 - RESTORE [524](#)
 - TAPECNTL [528](#)
 - core image
 - Stand-Alone Services feature (*continued*)
 - core image (*continued*)
 - definition of [267](#), [529](#)
 - specifying information for [270](#)
 - specifying the JCL output location for [272](#)
 - specifying the output DASD volume [272](#)
 - DASD
 - devices in XA or ESA mode [515](#)
 - target device address, specifying [525](#)
 - dump data set
 - device address, specifying [525](#)
 - device type, specifying [525](#)
 - examples
 - BUILDSA command [272](#)
 - copying the Stand-Alone Services core image [273](#)
 - IPL [521](#)
 - RESTORE command [527](#)
 - TAPECNTL command [529](#)
 - operator console
 - potential problems [516](#)
 - predefining [515](#), [516](#)
 - optional parameters
 - RESTORE command [526](#)
 - predefined console
 - potential problems with [523](#)
 - purpose of [516](#)
 - required parameters
 - BUILDSA command [270](#)
 - RESTORE command [525](#)
 - TAPECNTL command [528](#)
 - RESTORE command
 - description [524](#)
 - ENDTRK parameter [526](#)
 - example [527](#)
 - FILE parameter [527](#)
 - FROMADDR parameter [525](#)
 - FULL parameter [526](#)
 - NOREADCHECK parameter [526](#)
 - NOVERIFY parameter [526](#)
 - optional parameters [526](#)
 - READCHECK parameter [526](#)
 - required parameters [525](#)
 - STARTTRK parameter [526](#)
 - syntax [525](#)
 - TAPEVOLSER parameter [527](#)
 - TOADDR parameter [525](#)
 - VERIFY parameter [525](#)
 - specifying
 - information for the Stand-Alone Services core image [270](#)
 - the DASD target device address [525](#)
 - the dump data set device address [525](#)
 - the dump data set device type [525](#)
 - the input device [520](#)
 - the JCL output location for IPL-able core image [272](#)
 - the message output device [520](#)
 - the output DASD volume for IPL-able core image [272](#)
 - where Stand-Alone Services is IPLed from [269](#)
 - supported
 - platforms [515](#)
 - syntax
 - BUILDSA command [270](#)

- Stand-Alone Services feature (*continued*)
 - syntax (*continued*)
 - RESTORE command [525](#)
 - TAPECNTL command [528](#)
 - tape
 - library, setup Stand-Alone feature [517](#)
 - library, transient mount [518](#)
 - mount and demount procedures [517](#)
 - tape library
 - setup Stand-Alone device feature [517](#)
 - transient mount feature [517](#)
 - TAPECNTL command
 - description [528](#)
 - DEVTYPE parameter [528](#)
 - example [529](#)
 - required parameters [528](#)
 - REWIND parameter [529](#)
 - syntax [528](#)
 - UNITADDR parameter [529](#)
 - UNLOAD parameter [529](#)
 - using
 - tape library [516](#)
- standard catalog search order [263](#)
- standard naming conventions [539](#)
- STARTTRK parameter (Stand-Alone RESTORE command) [526](#)
- startup function – eioption 00 [589](#)
- statistical information [33](#)
- status assignment, GDG data sets [355](#), [490](#)
- STPCAT DD statement [78](#)
- STGADMIN.ADR.COPY.FCSETGT profile
 - FCSETGTOK keyword [535](#)
- storage
 - administrator [542](#)
- storage class
 - changing with COPY [120](#)
 - changing with RESTORE [86](#)
- storage group [3](#)
- Storage Management Subsystem, converting to and from [135](#)
- storage requirements
 - DFSMSdss [19](#)
 - PARALLEL command, effect of [19](#)
 - PDS (partitioned data set) [20](#)
 - per command
 - above 16MB [20](#)
 - below 16MB [19](#), [20](#)
 - VSAM data set [20](#)
- STORCLAS keyword
 - COPY command [353](#)
 - RESTORE command [489](#)
- STORCLAS profile [541](#)
- STORGRP keyword
 - COPY command [354](#)
 - DUMP command [421](#)
 - logical processing [25](#)
 - RELEASE command [450](#)
 - SPACEREL command [505](#)
- striped data sets, supported [22](#)
- structure, exit interface [584](#)
- subkeywords
 - maximum number [246](#)

- subkeywords (*continued*)
 - multiple [246](#)
 - specifying [246](#)
- summary of changes [xxv](#)
- supported
 - platforms for Stand-Alone Services [515](#)
- supported devices [22](#), [93](#)
- suppressing page ejection [250](#)
- switching modes [508](#)
- syntax [311](#)
- syntax diagram
 - CONSOLIDATE command [286](#)
 - CONVERTV command [295](#)
 - COPY command [301](#)
 - COPYDUMP command [369](#)
 - DEFRAG command [371](#)
 - DUMP command [387](#), [390](#)
 - PRINT command [433](#)
 - reading a [246](#)
 - RELEASE command [442](#)
 - RESTORE FULL command [454](#)
 - RESTORE TRACKS command [455](#)
- SYS1 subkeyword
 - COPY command [348](#)
 - DUMP command [417](#)
 - RELEASE command [449](#)
- SYS1.ANTMAIN.SNAPnnnn data sets [61](#)
- SYSALLDA [98](#)
- SYSDA [98](#)
- SYSIN
 - DD statement, JCL [248](#)
 - record, reading – eioption 01 [590](#)
- SYSPRINT
 - DD statement, JCL [248](#)
 - reading utility – eioption 09 [592](#)
 - record printing – eioption 02 [590](#)
 - writing record – eioption 10 [593](#)
- system
 - authorization facility [533](#)
 - programming information [584](#)
- system consoles supported [21](#)
- system data mover (SDM) [61](#)
- system data set
 - dumping [52](#)
 - moving [112](#)
- system environment [19](#)
- system requirements [19](#)
- system volumes
 - backing up [56](#)
 - recovering [94](#)

T

- tape
 - mount and demount procedures [517](#)
 - output I/O error – eioption 19 [595](#)
 - physical record backspace – eioption 25 [602](#)
 - reading logical record – eioption 04 [591](#)
 - reading physical record – eioption 03 [591](#)
 - volume mount OPEN/EVOC nonspecific – eioption 17 [595](#)
 - volume security OPEN/EOV – eioption 16 [594](#)
 - writing logical record – eioption 05 [591](#)

- tape (*continued*)
 - writing physical record – eioption 06 [592](#)
- tape devices (with Stand-Alone Services)
 - in XA or ESA mode [515](#)
- tape devices supported [21](#)
- TAPEVOLSER parameter (Stand-Alone RESTORE command) [527](#)
- target allocation [489](#)
- target class selection [496](#)
- target definition [249](#)
- target volume [16](#), [119](#)
- task processing, controlling [508](#)
- TASK-ID [585](#)
- temporary
 - data set [540](#)
 - data set names [539](#)
 - message data set [539](#)
- temporary copied catalogs [24](#)
- temporary data set [23](#), [141](#)
- temporary data set names [22](#)
- temporary work space [98](#)
- TEST keyword [138](#)
- TEST keyword (CONVERTV command) [299](#)
- TGTALLOC keyword
 - COPY command [354](#)
 - RESTORE command [489](#)
- TGTGDS keyword
 - COPY command [355](#)
 - RESTORE command [490](#)
- THEN command [512](#)
- TMPMSGDS keyword [250](#)
- TOADDR parameter (Stand-Alone RESTORE command) [525](#)
- TOL(ENQF) keyword, with backup function for HFS [49](#)
- TOLERATE keyword
 - COPY command [355](#)
 - DUMP command [421](#)
 - PRINT command [437](#)
 - RESTORE command [490](#)
- TRACE keyword [250](#)
- track utilization, maximize by reblocking [153](#)
- TRACKS keyword
 - COPY command [356](#)
 - DUMP command [422](#)
 - physical processing [26](#)
 - PRINT command [438](#)
 - RESTORE command [490](#)
- tracks read, on dump [415](#)
- trademarks [654](#)
- Transparent Cloud Tiering TCT [8](#)
- Transparent Cloud Tiering TCT requirement [9](#)
- TRK subkeyword of TGTALLOC
 - COPY command [354](#)
 - RESTORE command [489](#)
- TSO FCWITHDRAW
 - freeing subsystem resources [109](#)
 - freeing subsystem resources, volume [127](#)
 - withdrawing FlashCopy relationship [109](#)
 - withdrawing FlashCopy relationship, volume [127](#)
- TTRADDRESS keyword
 - COPY command [357](#)
 - RESTORE command [491](#)
- TYPRUN keyword [250](#)

U

- UAPTR parameter [578](#)
- UIMPTR parameter [578](#)
- UNCATALOG keyword
 - COPY command [357](#)
 - DUMP command [423](#)
- uncataloged data set [30](#)
- undefined DSORG data sets
 - moving [111](#)
 - restoring [83](#)
- UNDEFINEDSORG subkeyword
 - COPY command [348](#)
 - RESTORE command [480](#)
- UNITADDR parameter (Stand-Alone TAPECNTL command) [529](#)
- universal access authority (UACC) [35](#)
- unlike devices
 - moving volumes [132](#)
- UNLOAD parameter (Stand-Alone TAPECNTL command) [529](#)
- unloading a tape with the TAPECNTL command [528](#)
- unmovable data set
 - moving [116](#)
 - REPLACE keyword [82](#)
 - REPLACEUNCONDITIONAL keyword [82](#)
 - restoring to preallocated targets [82](#)
- unused space, releasing [143](#)
- unwanted data sets, deleting [144](#)
- USEEXCP keyword [251](#)
- User
 - area list [578](#)
 - Area Pointer [587](#)
 - Exit Allowance [585](#)
 - Return Code [585](#)
- user catalog, moving [112](#)
- User Interaction Module (UIM)
 - example [629](#)
 - list [578](#)
 - sample output [639](#)
- user interaction module exit functions [34](#)
- user interface
 - ISPF [649](#)
 - TSO/E [649](#)
- using an object storage cloud [14](#)
- using SIZE and MAXSIZE variables [167](#)
- utilities
 - in data set copy operation [104](#)
 - moving data sets [104](#)
- utilization, percent
 - COPY command [347](#)
 - RESTORE command [479](#)
- UTILMSG keyword [251](#)

V

- VALIDATE keyword
 - record counting [4](#)
- VALIDATE keyword (DUMP command) [424](#)
- variable length (VL) parameter [579](#)
- verification
 - bypass exit – eioption 22 [596](#)
 - data set – eioption 21 [596](#)
 - exit, OPEN/EOV – eioption 16 [594](#)
- VERIFY parameter (Stand-Alone RESTORE command) [525](#)

- VERSION1 keyword
 - DEFRAG command [383](#)
- versions of DFSMSdss, compatibility [647](#)
- virtual concurrent copy
 - defined [8](#)
 - SnapShot [8](#)
- virtual concurrent copy mode [16](#)
- virtual concurrent copy working space [61](#)
- virtual input/output (VIO) device supported [21](#)
- virtual storage access method [52](#)
- vital records [26](#), [39](#), [41](#)
- VL parameter [579](#)
- VM minivolumes [46](#)
- VM-format volumes
 - backing up [56](#)
 - moving [133](#)
 - recovering [94](#)
 - with DFSMSdss [21](#)
- VM, running Stand-Alone Services under [515](#)
- VOLCOUNT keyword
 - COPY command [357](#)
 - RESTORE command [492](#)
- volume
 - backup [5](#), [39](#), [55](#)
 - broken [27](#)
 - conversion without data movement [17](#)
 - copy and dump combining [57](#)
 - DUMP with concurrent copy [427](#)
 - formats supported by DFSMSdss [21](#)
 - input, specifying [98](#), [100](#)
 - line operator module names [36](#)
 - logical copy operation [125](#)
 - logical dump [55](#)
 - multiple target [123](#)
 - multiple target, restore [79](#)
 - notification – eioption [20](#) [595](#)
 - output, selecting [100](#)
 - output, selection [74](#), [78](#)
 - output, specifying [93](#)
 - physical copy [125](#)
 - physical dump [55](#)
 - preparing for conversion [138](#)
 - selecting target [119](#)
 - serialization [562](#)
 - system, recovering [94](#)
 - VM-formatted [21](#)
- volume consideration [231](#)
- volume copy
 - reducing backup with volume dump [57](#)
- volume dump
 - reducing backup with volume copy [57](#)
- volume list, ISMF [97](#)
- volume serial rules
 - for Linux volumes [170](#)
- volumes
 - using incremental FlashCopy [64](#)
- VSAM (virtual storage access method)
 - cluster, restoring [84](#)
 - data set
 - converting [85](#), [114](#)
 - copying [115](#)
 - dumping [52](#)
 - moving [114](#)
 - moving preformatted empty [124](#)

- VSAM (virtual storage access method) (*continued*)
 - data set (*continued*)
 - restoring preformatted empty [90](#)
 - storage requirements [20](#)
 - preallocation [121](#)
 - sphere
 - dumping [52](#)
 - eligibility [139](#)
 - restoring [83](#)
- VSAM (Virtual Storage Access Method)
 - data set
 - filtering [255](#)
 - restore actions on [495](#)
- VTOC
 - indexed [21](#)
 - keyword (PRINT command) [418](#), [439](#)
 - nonindexed [21](#)
- VTOC Index [22](#)

W

- WAIT keyword
 - COMPRESS command [285](#)
 - CONSOLIDATE command [294](#)
 - COPY command [359](#)
 - DEFRAG command [383](#)
 - DUMP command [424](#)
 - PRINT command [439](#)
 - RELEASE command [450](#)
 - RESTORE command [493](#)
- WAIT option [563](#), [569](#)
- wait-state codes for Stand-Alone program [522](#)
- when to use
 - logical processing [25](#)
 - physical processing [27](#)
- work space [98](#)
- working space data set [61](#)
- WORKINGDIRECTORY keyword
 - DUMP command [425](#)
 - RESTORE command [494](#)
- WORKUNIT keyword [251](#)
- WORKVOL keyword [251](#)
- WRITECHECK keyword
 - CONSOLIDATE command [295](#)
 - COPY command [360](#)
 - DEFRAG command [383](#)
 - RESTORE command [494](#)
- writing
 - disk track – eioption [08](#) [592](#)
 - logical tape record – eioption [05](#) [591](#)
 - physical tape record – eioption [06](#) [592](#)
 - SYSPRINT record – eioption [10](#) [593](#)
 - to the operator [507](#)
 - WTO message – eioption [11](#) [593](#)
 - WTOR message – eioption [12](#) [593](#)
- Writing to the Operator command
 - for DFSMSdss [507](#)
- WTO
 - command [507](#)
 - message writing – eioption [11](#) [593](#)
- WTOR
 - command [507](#)
 - message writing – eioption [12](#) [593](#), [594](#)
- WTOR command

WTOR command (*continued*)
for DFSMSdss [507](#)

X

XABUFF keyword [251](#)

Z

ZBUFF64R keyword [251](#)

ZCOMPRESS keyword

DUMP command [425](#)

zFS data sets

DUMP command considerations [420](#)

logical copy [49](#)

logical copy and serialization [565](#)

logical data set COPY [301](#)

logical dump [49](#)

logical dump and serialization [565](#)

physical dump and serialization [565](#)

read/write serialization scheme [566](#)

RELEASE command consideration [442](#)



Product Number: 5655-ZOS

SC23-6868-70

