This edition applies to Version 1 Release 13 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC35-0423-16.

© Copyright IBM Corporation 1984, 2012.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Figures . ........................................ xi

Tables ........................................... xiii

About this document . ........................ xv
Required product knowledge .............. xv
Referenced documents ....................... xv
Accessing z/OS DFSMS information on the Internet xvi
The z/OS Basic Skills Information Center .......................... xvi

How to send your comments to IBM ...... xvii
If you have a technical problem .......... xvii

Summary of changes ......................... xix
Changes made in z/OS Version 1 Release 13 xix
Changes made in z/OS Version 1 Release 12 xx
Changes made in z/OS Version 1 Release 11 xxi
Changes made in z/OS Version 1 Release 10 xx

Part 1. DFSMSdss Storage
Administration Guide ........................ 1

Chapter 1. Introduction to the
DFSMSdss component of DFSMS. ........ 7
Understanding the role of DFSMSdss ....... 7
Managing user data with SMS .......... 7
Sequential data striping .................... 7
Record counting ............................. 8
Installation exit routines ................. 8
Authorization checking ..................... 8
Managing availability with DFSMSdss . 9
  Backing up and restoring volumes and data sets 10
  Using DFSMShsm for backup ............ 10
  Using concurrent copy ................. 11
  Using the stand-alone restore program of
  DFSMSdss .................................... 12
Managing data movement with DFSMSdss . 12
  Moving data ................................ 12
  Moving data in an SMS-managed environment 12
  Moving data with concurrent copy .... 13
  Moving data with FlashCopy ........... 13
  Moving data with Snapshot ............ 13
Converting data to and from SMS management 13
  Converting data sets with data movement 13
  Converting volumes without data movement 14
Managing space with DFSMSdss .......... 14

Chapter 2. Requirements for running
DFSMSdss ................................... 15
Understanding the operating environment 15
Storage requirements ....................... 15

Hardware requirements ..................... 17
Volume formats .............................. 17
Indexed VTOC ................................ 18
Data set organizations .................... 18
Temporary data set names ................. 18

Chapter 3. Logical and physical
processing and data set filtering ......... 21
Defining logical and physical processing . 21
Logical processing .......................... 21
Physical processing .......................... 22
Data integrity considerations ............. 23
Choosing data sets for processing—filtering 24
  Filtering by data set names ............ 24
  Filtering by data set characteristics .... 25
  The FILTERDD keyword ................. 27
  Uses of filtering .......................... 27

Chapter 4. Invoking DFSMSdss ........... 29
Invoking DFSMSdss with ISMF .......... 29
How to invoke ISMF ....................... 29
Invoking DFSMSdss with JCL .......... 29
Invoking DFSMSdss with the application interface 29
  User interaction module exit functions 30

Chapter 5. Protecting DFSMSdss
functions ................................ 31
Protecting DFSMSdss and ISMF functions with
  RACF ........................................ 31
  ISMF functions you might want to protect 31
  Setting up the authorization structure .... 31
Protecting DFSMSdss functions with RACF
  FACILITY class profiles ............... 33
  Name-hiding ............................. 34

Chapter 6. Managing availability with
DFSMSdss ................................... 35
Planning an availability strategy .......... 35
  Backup and recovery .................... 35
  Disaster recovery ....................... 36
  Maintaining vital records ............. 38
  Archiving data sets ..................... 38
  Backing up data sets ................. 39
  Logical data set dump ................. 40
  Physical data set dump ............... 40
  Renaming data sets during dump processing 41
  Backup with concurrent copy .......... 42
  Using DFSMSdss as a backup utility for CICSVR 44
  A backup scenario ....................... 45
  Backing up data sets with special requirements 46
  Dumping HFS data sets ............... 46
  Dumping zFS data sets ............... 46
  Dumping multivolume data sets ....... 47
  Dumping integrated catalog facility user catalogs 48
Chapter 7. Managing data movement
with DFSMSdss

Preparing for data movement
Evaluating the use of logical and physical copy
Controlling what DFSMSdss copies
Moving data sets
Moving volumes
Logical data set copy
Physical data set copy
Specifying input volumes
Selecting output volumes
Renaming data sets
Expiration date handling
SMS to SMS
SMS to non-SMS
Non-SMS to SMS
Non-SMS to non-SMS
Defining RACF profiles
Moving data sets with utilities
Moving data sets with concurrent copy
Specifying concurrent copy for COPY requests
Moving data sets with FlashCopy
Designating FlashCopy usage
Moving data sets with SnapShot
Moving data sets with special requirements
Moving undefined DSORG and empty non-VSAM data sets
Moving system data sets
Moving catalogs
Moving non-VSAM data sets that have aliases
Moving multivolume data sets
Converting VSAM and non-VSAM data sets to multivolume.
Moving VSAM data sets
Moving a PDSE
Moving a damaged PDS
Moving unmovable data sets
Moving data sets to unlike devices
Moving indexed sequential data sets
Moving direct access data sets
Moving GDG data sets
Moving SMS-managed data sets
Moving non-SMS-managed data sets
Moving to preallocated data sets
Moving data sets being accessed with record level sharing
Moving preformatted empty VSAM data sets
VTOC considerations for moving volumes.
Logical volume copy operation
Physical volume copy operation
Moving volumes with FlashCopy
Designating FlashCopy usage
Determining why FlashCopy cannot be used
Freeing subsystem resources
Choosing space efficient FlashCopy with the FCSETGTOK keyword

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initializing the volume with the FCWITHDRAW keyword</td>
<td>134</td>
</tr>
<tr>
<td>Backing up volumes with FlashCopy consistency group</td>
<td>135</td>
</tr>
<tr>
<td>Moving volumes with SnapShot</td>
<td>137</td>
</tr>
<tr>
<td>Designating SnapShot usage</td>
<td>137</td>
</tr>
<tr>
<td>Determining why SnapShot cannot be used</td>
<td>137</td>
</tr>
<tr>
<td>Moving volumes to like devices of equal capacity</td>
<td>138</td>
</tr>
<tr>
<td>Moving volumes to like devices of greater capacity</td>
<td>138</td>
</tr>
<tr>
<td>Moving volumes to unlike devices</td>
<td>139</td>
</tr>
<tr>
<td>Moving VM-format volumes</td>
<td>139</td>
</tr>
<tr>
<td>Designating SnapShot usage</td>
<td>158</td>
</tr>
<tr>
<td>Determining why SnapShot cannot be used</td>
<td>158</td>
</tr>
<tr>
<td>Data sets excluded from DEFRAG or CONSOLIDATE processing</td>
<td>158</td>
</tr>
<tr>
<td>DEFRAG options</td>
<td>159</td>
</tr>
<tr>
<td>Serialization</td>
<td>161</td>
</tr>
<tr>
<td>Security considerations</td>
<td>164</td>
</tr>
<tr>
<td>Maximizing track utilization by reblocking data sets</td>
<td>165</td>
</tr>
</tbody>
</table>

**Chapter 8. Converting data to and from SMS management**

- Evaluating conversion to SMS management | 141 |
- Data sets ineligible for conversion to SMS | 141 |
- Volumes eligible for conversion to SMS | 142 |
- Conversion by data movement | 142 |
- Conversion without data movement | 143 |
- Preparing a volume for conversion | 144 |
- Conversion to SMS management without data movement | 145 |
- SMS report | 146 |
- Special data set requirements for conversion to SMS | 146 |
- VSAM sphere eligibility | 146 |
- Multivolume data sets | 146 |
- GDG data sets | 147 |
- Temporary data sets | 148 |
- VTOC and VVDS | 148 |
- Converting from SMS management without data movement | 148 |
- Special data set requirements for conversion from SMS | 148 |
- Multivolume data sets | 148 |
- GDG data sets | 149 |
- Temporary data sets | 149 |
- VTOC and VVDS | 149 |
- Special considerations for using non-SMS-managed targets | 149 |

**Chapter 9. Managing space with DFSMSdss**

- Reclaiming DASD space | 151 |
- Releasing unused space in data sets | 151 |
- Compressing a PDS | 152 |
- Deleting unwanted data sets | 152 |
- Combining data set extents | 153 |
- Consolidating free space and extents on volumes | 155 |
- When to run DEFRAG and CONSOLIDATE functions | 155 |
- Designating FlashCopy usage | 156 |
- Preserve Mirror FlashCopy | 156 |
- Determining why FlashCopy cannot be used | 157 |
- Designating SnapShot usage | 158 |
- Determining why SnapShot cannot be used | 158 |
- Data sets excluded from DEFRAG or CONSOLIDATE processing | 158 |
- DEFRAG options | 159 |
- Serialization | 161 |
- Security considerations | 164 |
- Maximizing track utilization by reblocking data sets | 165 |

**Chapter 10. Diagnosing problems in DFSMSdss operations**

- Determining the source of the failure: DFSMSdfp, DFSMSdss, or DFSMSshsm | 167 |
- Using keywords to identify the problem | 168 |
- Component identification keyword | 168 |
- Release-level keyword | 169 |
- Type-of-failure and function keywords | 169 |
- Module keyword | 174 |
- Maintenance-level keyword | 176 |
- Using the IBM Support Center | 176 |
- Using the software support facility | 176 |
- Using IBMLink/ServiceLink | 177 |
- Info/System | 177 |

**Chapter 11. ACS routine information**

- ACS variables available during Copy function | 179 |
- ACS variables available during RESTORE and CONVERTV processing | 181 |
- Using SIZE and MAXSIZE variables | 182 |

**Chapter 12. Dumping and restoring Linux for zSeries partitions and volumes**

- Preparing to work with Linux volumes | 185 |
- Understanding the hardware environment | 185 |
- Choosing VOLSERs for Linux volumes | 186 |
- Formatting and partitioning Linux volumes | 186 |
- Obtaining authorization for Linux volumes | 187 |
- Backing up a Linux volume with partitions | 187 |
- Using DFSMSdss dump and restore commands | 188 |
- Example 1. DUMP FULL | 189 |
- Example 2. DUMP FULL with CONCURRENT COPY | 190 |
- Example 3. DUMP DATASET | 190 |
- Example 4. COPY FULL | 191 |
- Example 5. COPY FULL COPYVOLID ALLEXCP | 192 |
- Example 6. RESTORE FULL | 192 |
- Example 7. RESTORE DATASET | 193 |
- Example 8. COPYDUMP | 195 |
- Submitting JCL batch jobs to a z/OS system using FTP | 195 |
- Using DFSMSdss stand-alone services | 195 |

**Chapter 13. Format of the DFSMSdss dump data set**

- Format of the DFSMSdss dump data set | 197 |
- ADRBMB data area | 198 |
### Chapter 14. DFSMSdss patch area

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample JCL</td>
<td>215</td>
</tr>
<tr>
<td>Forcing the use of preallocated VSAM data sets (PN04574)</td>
<td>215</td>
</tr>
<tr>
<td>Ignoring VSAM duplicate key errors (PN05529)</td>
<td>216</td>
</tr>
<tr>
<td>Modifying the timeout period for enqueue lockout detection (PL84514)</td>
<td>217</td>
</tr>
<tr>
<td>Controlling the wait/retry time for serialization of system resources (PN11523)</td>
<td>217</td>
</tr>
<tr>
<td>Using CONVERTV on data sets with a revoked user ID in the RESOWNER field (OY59957)</td>
<td>218</td>
</tr>
<tr>
<td>Restoring inconsistent PDSE data sets (OY60301)</td>
<td>218</td>
</tr>
<tr>
<td>Changing default protection status during RESTORE (PN37489)</td>
<td>219</td>
</tr>
<tr>
<td>Restoring or copying undefined, multivolume SMS-managed data sets (OY65818)</td>
<td>219</td>
</tr>
<tr>
<td>Bypassing backup-while-open processing (OY63531)</td>
<td>220</td>
</tr>
<tr>
<td>Bypassing storage and management class authorization checking during RESTORE (OY65348)</td>
<td>220</td>
</tr>
<tr>
<td>Issuing notification for tape and migrated data sets (OY66092)</td>
<td>221</td>
</tr>
<tr>
<td>Using RESET with concurrent copy (OY65555)</td>
<td>221</td>
</tr>
<tr>
<td>Forcing RESTORE after message ADR482E (OY67532)</td>
<td>222</td>
</tr>
<tr>
<td>Restoring VSAM KSDS or VRRDS after messages ADR789W, ADR364W, and ADR417W (OY67942)</td>
<td>222</td>
</tr>
<tr>
<td>Using CONVFF on data sets with a revoked user ID in the RESOWNER field (OY59957)</td>
<td>223</td>
</tr>
<tr>
<td>Restoring VSAM data sets with expiration date of 1999365 (OW00780)</td>
<td>223</td>
</tr>
<tr>
<td>Restoring VSAM data sets with expiration dates beyond 2000 (OW00780)</td>
<td>223</td>
</tr>
<tr>
<td>Changing default insertion of EOF track during COPY with ALLDATA specified (OW15003)</td>
<td>224</td>
</tr>
<tr>
<td>Using RESET or UNCATALOG in a logical data set dump (PN60114)</td>
<td>224</td>
</tr>
<tr>
<td>Changing secondary allocation quantity in format 1 DSCB for PDSE data sets (OW07755)</td>
<td>225</td>
</tr>
<tr>
<td>Changing reference date default settings during data set COPY and RESTORE processing (OW12011)</td>
<td>226</td>
</tr>
<tr>
<td>Changing default protection processing during COPY (OW10314)</td>
<td>227</td>
</tr>
<tr>
<td>Bypassing management and storage class access checks during COPY (PN72592)</td>
<td>227</td>
</tr>
<tr>
<td>Changing default handling of invalid tracks created during data set COPY and RESTORE processing (OW08174)</td>
<td>228</td>
</tr>
<tr>
<td>Forcing RESTORE to the same volumes as the source VSAM data set (OW07077)</td>
<td>228</td>
</tr>
<tr>
<td>Modifying number of volumes allocated for SMS data sets during logical RESTORE and COPY (OW15880)</td>
<td>229</td>
</tr>
<tr>
<td>Dumping a keyed VSAM data set that has data CAs without corresponding index CIs (OW17877)</td>
<td>230</td>
</tr>
<tr>
<td>Changing the default DEFRAG processing of checkpointed data sets (OW20285)</td>
<td>231</td>
</tr>
<tr>
<td>Setting the percentage to overallocate target data set space (OW27837)</td>
<td>231</td>
</tr>
<tr>
<td>Bypassing RLS processing (OW32817)</td>
<td>232</td>
</tr>
<tr>
<td>Changing creation date default settings during logical data set COPY and RESTORE (OW19618)</td>
<td>233</td>
</tr>
<tr>
<td>Copying and dumping a PDSE data set using the VALIDATE PDSE option (OW48074)</td>
<td>233</td>
</tr>
<tr>
<td>Changing the default maximum number of active parallel subtasks</td>
<td>234</td>
</tr>
<tr>
<td>Changing the default initialization processing during DUMP with FCWITHDRAW (OA18929)</td>
<td>234</td>
</tr>
<tr>
<td>Changing the default DEFRAG processing of LINLIST-indicated data sets (OW43874)</td>
<td>235</td>
</tr>
<tr>
<td>Changing the FASTREPLICATION default setting during Copy and Defrag (OA11637)</td>
<td>236</td>
</tr>
<tr>
<td>Tuning hardware assisted compression (OA13300)</td>
<td>236</td>
</tr>
<tr>
<td>Resetting the data-set-changed indicator during physical full or partial RESTORE operation (OA20907)</td>
<td>237</td>
</tr>
<tr>
<td>Requesting that DFSMSdss double-check data set high used RBA values for LDS data sets (OA23805)</td>
<td>238</td>
</tr>
<tr>
<td>Enabling or disabling use of the catalog search interface for data set name filtering</td>
<td>238</td>
</tr>
<tr>
<td>Requesting that DFSMSdss restore the VM formatted volume that was DUMPed by z/OS V1R10 before OA27531 was applied</td>
<td>239</td>
</tr>
<tr>
<td>Adding timestamps to messages</td>
<td>239</td>
</tr>
<tr>
<td>Enabling building appropriate channel programs</td>
<td>239</td>
</tr>
<tr>
<td>ADRPTCHB data area</td>
<td>240</td>
</tr>
<tr>
<td>ADRPTCHB cross-reference</td>
<td>242</td>
</tr>
</tbody>
</table>

### Part 2. DFSMSdss Storage Administration Reference

### Chapter 15. Specifying DFSMSdss commands

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command syntax</td>
<td>253</td>
</tr>
<tr>
<td>How many subkeywords are allowed?</td>
<td>254</td>
</tr>
<tr>
<td>Specifying subkeywords in a command data set</td>
<td>254</td>
</tr>
<tr>
<td>How to read syntax diagrams</td>
<td>255</td>
</tr>
<tr>
<td>JCL that you need</td>
<td>257</td>
</tr>
<tr>
<td>How to control DFSMSdss through PARM information in the EXEC statement</td>
<td>258</td>
</tr>
<tr>
<td>Examples of invoking DFSMSdss with JCL</td>
<td>261</td>
</tr>
</tbody>
</table>

### Chapter 16. DFSMSdss filtering—choosing the data sets you want processed

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How DFSMSdss filters data sets</td>
<td>263</td>
</tr>
<tr>
<td>Virtual storage access method (VSAM) data set considerations</td>
<td>263</td>
</tr>
<tr>
<td>Filtering by data set names</td>
<td>264</td>
</tr>
<tr>
<td>Using an asterisk in partially qualified data set</td>
<td>264</td>
</tr>
<tr>
<td>Using a percent sign in partially qualified data set</td>
<td>264</td>
</tr>
</tbody>
</table>

---

vi  
z/OS V1R13.0 DFSMSdss Storage Administration
Examples of fully and partially qualified data set names ............. 265
Relative generation filtering ..................................... 266
Filtering by data set characteristics ................................ 266
Some examples of the BY keywords .................................. 269
Standard catalog search order ........................................ 270
Broken data set considerations ....................................... 270

Chapter 17. Syntax—DFSMSdss

function commands ................................................. 271

What DFSMSdss commands do ........................................ 271
Building the stand-alone IPL-able core image ........................ 271
Using DUMP and RESTORE for backup and recovery ............... 271
Moving data with COPY ............................................ 271
Converting to and from Storage Management Subsystem (SMS) with CONVERTV ............................................ 272
Managing space with COMPRESS, CONSOLIDATE, DEFRAG, and RELEASE. ............................................ 272
Using COPY for partitioned data set (PDS) and partitioned data set extended (PDSE) conversions ............................................ 272
Copying DFSMSdss-produced dump data with COPYDUMP ............... 272
Printing for diagnostic purposes with PRINT .......................... 273
BUILDSA command for DFSMSdss .................................... 273
BUILDSA syntax .................................................... 274
Explanation of BUILDSA command keywords .......................... 274
BUILDSA command examples ......................................... 276
CGCREATED command for DFSMSdss .................................. 278
CGCREATED syntax .................................................. 279
Explanation of CGCREATED command keywords ....................... 279
COMPRESS command for DFSMSdss .................................... 280
COMPRESS syntax ................................................... 280
Explanation of COMPRESS command keywords ....................... 281
Example of compress operations ....................................... 282
CONSOLIDATE command for DFSMSdss ................................ 285
CONSOLIDATE command syntax ...................................... 285
Explanation of CONSOLIDATE command keywords ..................... 287
Example of a CONSOLIDATE operation .................................. 295
CONVERTV command for DFSMSdss .................................... 296
CONVERTV command syntax ......................................... 296
Explanation of CONVERTV command keywords ......................... 297
Examples of CONVERTV operations ................................... 300
COPY Command for DFSMSdss ......................................... 301
Special Considerations for COPY ....................................... 301
COPY DATASET Command Syntax for Logical Data Set ............... 302
COPY DATASET Command Syntax for Physical Data Set ................ 305
COPY FULL and COPY TRACKS Syntax .................................. 307
Explanation of COPY Command Keywords ............................... 311
Data Integrity Considerations for Full or Tracks Copy Operation ......... 363
Examples of Full and Tracks Copy Operations ......................... 363
Examples of Data Set Copy Operations .................................. 364
ALLDATA and ALLEXCP Interactions .................................. 369
COPYDUMP command for DFSMSdss .................................... 372
COPYDUMP syntax .................................................. 372
Explanation of COPYDUMP command keywords ....................... 372
Examples of COPYDUMP operations .................................... 373
DEFRAG command for DFSMSdss ...................................... 374
DEFRAG syntax .................................................... 375
Explanation of DEFRAG command keywords .......................... 376
Examples of DEFRAG operations ..................................... 388
Results of a successful DEFRAG operation ............................ 389
DUMP command for DFSMSdss ........................................ 391
Special considerations for dump ....................................... 391
DUMP FULL and DUMP TRACKS syntax .............................. 392
DUMP DATASET syntax for logical data set ............................ 394
DUMP DATASET syntax for physical data set .......................... 396
Explanation of DUMP command keywords ............................. 398
Data integrity considerations for full or tracks dump operation .......... 424
Format of the output data set ........................................ 424
Examples of full and tracks dump operations .......................... 424
Examples of physical data set dump operations ........................ 425
Examples of logical data set dump operations .......................... 428
PRINT command for DFSMSdss ....................................... 431
PRINT syntax .................................................... 431
Explanation of PRINT command keywords ............................. 432
Examples of print operations .......................................... 438
RELEASE command for DFSMSdss ..................................... 440
RELEASE syntax for physical processing ............................... 441
RELEASE syntax for logical processing ............................... 442
Explanation of RELEASE command keywords .......................... 443
Example of a release operation ....................................... 450
RESTORE command for DFSMSdss ..................................... 451
Special considerations for RESTORE .................................... 452
Data integrity considerations for full or tracks restore operations .. 452
RESTORE FULL and RESTORE TRACKS command syntax ............ 453
RESTORE DATASET command syntax for logical data set .......... 454
RESTORE DATASET command syntax for physical data set .......... 457
Explanation of RESTORE command keywords ......................... 458
DFSMSdss RESTORE process ........................................ 489
Examples of full and tracks restore operations .......................... 493
Examples of physical data set restore operations ....................... 494
Examples of logical data set restore operations ......................... 497

Chapter 18. Syntax—auxiliary commands .................................. 501

Writing to the operator for DFSMSdss ...................... 501
WTO command .................................................. 501
WTOR command for DFSMSdss ...................................... 501
Scheduling tasks ................................................... 502
SERIAL command for DFSMSdss .................................... 502
PARALLEL command for DFSMSdss ................................ 502
Controlling task processing .......................................... 502
SET command—setting condition codes and patch bytes ............... 503
IF-THEN-ELSE command sequence for DFSMSdss—using condition codes 504
EOJ command—ending your DFSMSdss step .......................... 508
Chapter 19. DFSMSdss stand-alone services
Preparing to run the stand-alone services program 509
Running stand-alone services in 370 mode 509
Running stand-alone services in XA or ESA mode 510
Running stand-alone services with a predefined console 511
Using a tape library 511
Using an automatic cartridge loader 514
Controlling command sequence processing 514
IPLing and running the Stand-Alone Services Program 514
IPLing Stand-Alone Services 516
RESTORE—restoring a formatted dump tape 520
TAPECNTL—rewinding and unloading a tape 525
Building the IPL-able core image 526
BUILDASA function 526
Understanding BUILDASA command authorization levels 526

Chapter 20. Data security and authorization checking 529
Effects of SPECIAL, OPERATIONS, and DASDVOL 529
SPECIAL 529
OPERATIONS 530
DASDVOL 530
General data security information 531
Protecting resources and data sets 531
Protecting the usage of DFSMSdss 531
Password protection 532
Protected user and group data sets 532
Generic and discrete profile considerations 533
Security-level, category, and label checking 535
Protect-all and always-call 535
Standard naming conventions 535
DFSMSdss temporary data set names 535
Discretely protected multivolume data set 537
Erase-on-scratch 537
SMS-managed data set protection 537
Logging 538
DFSMSdss storage administrator 538
ADMINISTRATOR keyword 539
FACILITY class profiles for the ADMINISTRATOR keyword 539
DFSMSdss volume, data set and catalog access authority 540
Volume access and DASDVOL 541
Data set access authorization levels 544
Protected catalogs 544
Non-SMS versus SMS authorization 545
System operator authorization, special data set types 545
Access authorization for DFSMSdss commands 545
CGCREATED 546
COMPRESS 546
CONSOLIDATE 546
CONVERTV 546
COPY 547
COPYDUMP 551
DEFRA G 551
DUMP 551
PRINT 552
RELEASE 552
RESTORE 553

Chapter 21. Data integrity—serialization 559
Volume serialization 559
Avoiding lockout 560
The WAIT option 561
Data set serialization 561
Enqueuing—ENQ 561
Dumping HFS data sets 562
Dumping zFS data sets 563
Dynamic allocation (DYNALLOC) 564
Enqueuing versus dynamic allocation of data sets 564
Read/Write serialization scheme 564
WAIT option 567
An example of RESERVE-ENQUEUE processing 568
Backup-while-open data sets (CICS and DFSMStvs) 568
Backup-while-open status definition 570
Backup-while-open processing 570
Backup-while-open and concurrent copy 572
TOLERATE (ENQFAILURE) and SHARE considerations 572
CICS recovery data 572
Backup-while-open data sets (IMS) 573

Chapter 22. Application programming interface 575
Calling block structure 575
User interactions 578
Service considerations 579
Cross-memory Application Interface overview 579
Using the cross memory application interface to control DFSMSdss 581
System programming information 582
Application interface blocks 583
Exit identification block 583
Application programming interface restrictions 587
Cross-memory application interface restrictions 587
User interaction module exit option descriptions 588
Function startup (Eioption 00) 588
Reading SYSIN record (Eioption 01) 589
Printing SYSPRINT record (Eioption 02) 589
Reading physical tape record (Eioption 03) 590
Reading logical tape record (Eioption 04) 590
Writing logical tape record (Eioption 05) 591
Writing physical tape record (Eioption 06) 591
Reading disk track (Eioption 07) 591
Writing disk track (Eioption 08) 591
Reading utility SYSPRINT (Eioption 09) 591
Writing SYSPRINT record (Eioption 10) 592
Writing WTO message (Eioption 11) 592
Writing WTO R message (Eioption 12) 592
Presenting ADRUFO record (Eioption 13) 593
Function ending (Eioption 14) 593
Presenting WTOR response (Eioption 15) 593
Figures

1. Output from a Dump of an Integrated Catalog Facility User Catalog .............. 49
2. Output from Restore of Integrated Catalog Facility User Catalog .............. 81
3. SMS Report ........................................ 146
4. Sample JCL for dumping the contents of a volume. .................. 189
5. Sample JCL for dumping two or more output tapes at the same time. .......... 190
6. Sample JCL for DUMP FULL with CONCURRENT COPY. ............... 190
7. Sample JCL for DUMP DATASET ........................................ 191
8. Sample JCL for dumping all of the Linux partitions .................. 191
9. Sample JCL for making a full volume copy of a volume .............. 192
10. Sample JCL for creating a backup copy of a Linux volume ............. 192
11. Sample JCL for restoring a full volume from a DFSMSdss dump .......... 193
12. Sample JCL for restoring individual partitions or data sets .......... 193
13. Sample JCL for renaming data sets to be restored .................. 194
14. Sample JCL for restoring only one partition of a volume ............... 195
15. Sample JCL for copying Linux volume dumps .......................... 195
16. Printed Output Resulting from a Successful DEFRAG Run on nonEAV: ... 390
17. A Section of the Printed Output Resulting from a Successful DEFRAG Run on EAV: ... 391
18. Output Resulting from a PRINT Command ........................................ 440
19. Restore Actions on Non-VSAM Data Sets ........................................ 491
20. DFSMSdss Target Class Selection ........................................ 492
21. Stand-Alone Services Restore Process Overview .................. 515
22. DFSMSdss Data Security Decisions ........................................ 542
23. Block Diagram for Backup-While-Open Serialization .................. 569
24. DFSMSdss Application Interface Structure ........................................ 578
25. DFSMSdss Exit Interface Structure ........................................ 583
26. The Application Program Process ........................................ 619
27. Application Interface Program Example ........................................ 620
28. User Interaction Module Example ........................................ 623
29. Output Resulting from Use of the UIM Exits ........................................ 634
Tables

1. Minimum Storage Requirements for DFSMSdss Operations with I/O Buffers below 16-Megabyte Virtual.............. 16
2. Minimum Storage Requirements for DFSMSdss Operations with I/O Buffers above 16-Megabyte Virtual.............. 16
3. Minimum Storage Requirements for a Restore to an Unlike Device.............. 17
4. Module Names for DFSMSdss/ISMF Line Operators ........................................... 32
5. Module Names for DFSMSdss/ISMF Data Set Application Commands .................. 32
6. RACF FACILITY Class Profile Names for DFSMSdss Keywords .......................... 33
7. RSA private tokens and required cryptographic hardware for decryption ........... 69
8. DFSMSdss processing of dump encryption requests ........................................... 71
9. DFSMSdss handling of the data-set-changed indicator during data set restore operations... 77
10. Data Mover Selection Matrix for Data Set Copy ............................................... 106
11. Default Situation for DFSMSdss to Allocate the SMS-Managed GDG Data Set .... 122
12. Data Set Erase Table for DEFRAG with z/OS Security Server (RACF element) .... 164
13. Summary of Type-of-Failure Keywords ......................................................... 169
14. Variables Passed to ACS Routines during DFSMSdss Copy Function ................ 179
15. Variables Passed to ACS Routines during DFSMSdss Restore and CONVERTV Processing ................................................................. 181
16. ADRBMB Mapping Macro ................................................................. 198
17. ADRBMB Mapping Macro ................................................................. 198
18. ADRTAPB Mapping Macro ................................................................. 199
19. ADRTAPB Mapping Macro ................................................................. 208
20. ADRPTCHB Mapping Macro ................................................................. 240
21. BY Keywords ...................................................................................... 267
22. ALLDATA and ALLEXCP Interactions When Copying to LIKE Device .......... 370
23. ALLDATA and ALLEXCP Interactions When Copying to UNLIKE Device ...... 371
24. Physical Data Set Restore Actions on SMS-Managed Data Sets ....................... 489
25. Stand-Alone Services Options when Using an IBM 3494 or 3495 Tape Library ... 512
26. DFSMSdss FACILITY Class Profiles ......................................................... 531
27. DASDVOL Access Authority ................................................................. 543
28. Access Authority for the DFSMSdss COPY Command .................................... 547
29. DFSMSdss COPY Command and RACF Profiles ........................................... 548
30. DFSMSdss Copy and Define Discrete Profile Summary ................................... 549
31. Access Authority for the DFSMSdss DUMP Command .................................... 552
32. Access Authority for the DFSMSdss RESTORE Command ......................... 553
33. DFSMSdss Copy and Define Profile Summary .............................................. 555
34. Data Set Enqueue Options for Non-VSAM Data Sets Specified on DFSMSdss Commands .... 562
35. Read/Write Access Serialization Scheme ................................................. 564
36. Resource Serialization ................................................................. 566
37. DFSMSdss Support for IMS Backup-While-Open Data Sets ......................... 573
38. ADRREID0 Mapping Macro ................................................................. 604
39. Data Set Attributes and How They Are Determined ..................................... 637
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 245.
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 17, "Syntax—DFSMSdss function commands," on page 271.
- 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 641.

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.

Referenced documents

The following publications are referenced in this book:

<table>
<thead>
<tr>
<th>Publication Title</th>
<th>Order Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>z/OS DFSMS Introduction</td>
<td>SC26-7397</td>
</tr>
<tr>
<td>z/OS DFSMSdfp Storage Administration</td>
<td>SC26-7402</td>
</tr>
<tr>
<td>z/OS DFSMS Installation Exits</td>
<td>SC26-7396</td>
</tr>
<tr>
<td>z/OS DFSMS Using the Interactive Storage Management Facility</td>
<td>SC26-7411</td>
</tr>
<tr>
<td>z/OS MVS System Messages, Vol 1 (ABA-AOM)</td>
<td>SA22-7631</td>
</tr>
</tbody>
</table>
Accessing z/OS DFSMS information on the Internet

In addition to making softcopy information available on physical media, IBM provides access to z/OS softcopy information on the Internet. To view, search, and print z/OS information, go to the z/OS Internet Library:

http://www.ibm.com/systems/z/os/zos/bkserv/

The z/OS Basic Skills Information Center

The z/OS Basic Skills Information Center is a Web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS system programmer.

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:

• Provide basic education and information about z/OS without charge
• Shorten the time it takes for people to become productive on the mainframe
• Make it easier for new people to learn z/OS.

To access the z/OS Basic Skills Information Center, open your Web browser to the following Web site, which is available to all users (no login required):

http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp
How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:

1. Send an email to mhvrdfs@us.ibm.com
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   U.S.A.
4. Fax the comments to us as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:
• Your name and address
• Your email address
• Your telephone or fax number
• The publication title and order number:
  z/OS V1R13.0 DFSMSdss Storage Administration
  SC35-0423-17
• The topic and page number related to your comment
• The text of your comment.

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

IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you submit.

If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:
• Contact your IBM service representative
• Call IBM technical support
• Visit the IBM zSeries support web page at http://www.ibm.com/systems/z/support/
Summary of changes

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

Changes made in z/OS Version 1 Release 13

This document contains information previously presented in z/OS Version 1 Release 13 DFSMSdss Storage Administration, SC35-0423-16.

The following sections summarize the changes to that information.

New information:
- Appendix A, “Coexistence Considerations,” on page 639 has been added.

Changes made in z/OS Version 1 Release 13

This document contains information previously presented in z/OS Version 1 Release 13 DFSMSdss Storage Administration, SC35-0423-15.

The following sections summarize the changes to that information.

Changed information:
- Fields have been added to “Constants for ADREID0” on page 612.

Changes made in z/OS Version 1 Release 13

This document contains information previously presented in z/OS Version 1 Release 12 DFSMSdss Storage Administration, SC35-0423-14.

The following sections summarize the changes to that information.

New information:
- Support for requesting that DFSMSdss add timestamps to certain message classes. For more information, refer to “Adding timestamps to messages” on page 239 and the description of the MSGTIME keyword in “How to control DFSMSdss through PARM information in the EXEC statement” on page 258.
- DFSMSdss FlashCopy Batch Protection, which allows DFSMSdss to direct a fast replication request to the storage group ACS routine. For more information, refer to “Moving data sets with FlashCopy” on page 109 and Table 14 on page 179.
- A new patch byte to modify how DFSMSdss builds channel programs to read tape blocks. For more information, refer to “Enabling building appropriate channel programs” on page 239.

Changed information:
- Use of the catalog search interface (CSI) for data set name filtering now must be explicitly enabled if you want it to be used. For information, refer to “Enabling or disabling use of the catalog search interface for data set name filtering” on page 238.
• Description of specifying X'FFFFFFF' as the high cylinder value with the TRACKS parameter of the COPY, DUMP, PRINT and RESTORE commands. When this value is used, DFSMSdss chooses the high cylinder value to be the end of the volume. For more information, refer to "TRACKS" on page 358 (COPY command), "TRACKS" on page 421 (DUMP command), "TRACKS" on page 436 (PRINT command) and "TRACKS" on page 485 (RESTORE command).

• Description of automatically refreshing the UCB of a volume that is the target of a COPY or RESTORE operation. For more information, refer to "VTOC considerations for moving volumes" on page 129.

• "ADRTAPB data area" on page 199 and "ADREID0 data area" on page 604.

Changes made in z/OS Version 1 Release 12

This document contains information previously presented in z/OS Version 1 Release 11 DFSMSdss Storage Administration, SC35-0423-13.

The following sections summarize the changes to that information.

New information:

• Keywords on the COPY command, to support fast reverse restore: "FCFASTREVERSERESTORE" on page 329 and "FCFULLVOLUMERELATION" on page 330.

• RACF® FACILITY class profile to protect the usage of the FCFASTREVERSERESTORE keyword of the COPY command. For more information, see Table 6 on page 33.

Changed information:

• Logical restore processing now converts data sets with the IMBED or REPLICATE attributes to data sets without those attributes. For information, see "Logical data set restore" on page 73.

• For backups to and from tape, and for backups on DASD when the backup data set is in the extended format, DFSMS now uses BSAM instead of EXCP to read from and write to DFSMSdss dump data sets during DUMP, COPYDUMP and RESTORE operations. This allows DFSMSdss to support 256K blocks when writing to and reading from a tape. It also allows the use of Extended Format Sequential (extended addressable, Compressible or striped) data sets on DASD. Changes for this are in:
  - "Managing availability with DFSMSdss" on page 9
  - "Space considerations" on page 56
  - "Format of the DFSMSdss dump data set" on page 197
  - "ADRTAPB data area" on page 199
  - "How to control DFSMSdss through PARM information in the EXEC statement" on page 258
  - "COPYDUMP command for DFSMSdss" on page 372
  - "DUMP command for DFSMSdss" on page 391
  - "COMPRESS" on page 401
  - "OUTDDNAME" on page 414

• The "Readers’ Comments - We’d Like to Hear from You" section at the back of this publication has been replaced with a new section "How to send your comments to IBM" on page xvii. The hardcopy mail-in form has been replaced with a page that provides information appropriate for submitting readers comments to IBM.
Changes made in z/OS Version 1 Release 11

This document contains information previously presented in z/OS Version 1 Release 11 DFSMSdss Storage Administration, SC35-0423-12.

The following sections summarize the changes to that information.

New information:

• “Enabling or disabling use of the catalog search interface for data set name filtering” on page 238 in Chapter 14, “DFSMSdss patch area,” on page 215.

Changes made in z/OS Version 1 Release 11

This document contains information previously presented in z/OS Version 1 Release 10 DFSMSdss Storage Administration Guide, SC35-0423-11 and the DFSMSdss section of the z/OS Version 1 Release 9 DFSMS Storage Administration Reference, SC26-7402-11

Moved information:

To improve on the usability of information, the following DFSMS publications were restructured in V1R11:

1. DFSMS Storage Administration Reference
2. DFSMSdss Storage Administration Guide
3. DFSMShsm Storage Administration Guide

Each of the new publications now contains information pertaining to only one feature of z/OS:

1. z/OS DFSMSdfp Storage Administration contains the DFSMSdfp section of the former DFSMS Storage Administration Reference.
2. z/OS DFSMSdss Storage Administration contains:
   a. Part 1, “DFSMSdss Storage Administration Guide,” on page 1
   b. Part 2, “DFSMSdss Storage Administration Reference,” on page 245
3. z/OS DFSMShsm Storage Administration contains:

New information:

• “Backing up data sets with extended attributes” on page 52 in Chapter 6, “Managing availability with DFSMSdss,” on page 35.

• Chapter 7, “Managing data movement with DFSMSdss,” on page 97 includes the following updated sections:
  - “Selecting output volumes” on page 101
  - “VTOC considerations for moving volumes” on page 129
  - “Moving volumes to like devices of greater capacity” on page 138

• “Changing creation date default settings during logical data set COPY and RESTORE (OW19618)” on page 233 in Chapter 14, “DFSMSdss patch area,” on page 215

• Chapter 22, “Application programming interface,” on page 575 includes the following updated sections:
  - “Application programming interface restrictions” on page 587
  - “Target data set allocation notification exit (Eioption 28)” on page 603
- "ADREID0 data area" on page 604
Moving data sets .......................... 98
Moving volumes ................................ 99
Logical data set copy .......................... 99
Physical data set copy ........................ 100
Specifying input volumes ................... 100
Selecting output volumes .................. 101
Renaming data sets ......................... 102
Expiration date handling ................... 104
   SMS to SMS .................................. 104
   SMS to non-SMS .............................. 105
   Non-SMS to SMS .............................. 105
   Non-SMS to non-SMS ....................... 105
Defining RACF profiles ..................... 105
Moving data sets with utilities .......... 105
Moving data sets with concurrent copy .. 108
   Specifying concurrent copy for COPY requests
Moving data sets with FlashCopy .......... 109
   Designating FlashCopy usage .............. 110
   Preserve Mirror FlashCopy ............... 110
   Determining why FlashCopy cannot be used
   Freeing subsystem resources ............. 112
   Moving data sets with SnapShot ........ 113
   Designating SnapShot usage ............. 113
   Determining why SnapShot cannot be used
Moving data sets with special requirements 114
Moving undefined DSORG and empty non-VSAM data sets 114
Moving system data sets .................. 115
Moving catalogs ............................ 115
Moving non-VSAM data sets that have aliases 116
Moving multivolume data sets .......... 116
Converting VSAM and non-VSAM data sets to multivolume . 118
Moving VSAM data sets ................... 118
   Restrictions for the COPY command .... 118
Moving a PDSE ............................... 119
Moving a damaged PDS .................... 119
Moving unmovable data sets .......... 120
Moving data sets to unlike devices ...... 120
Moving indexed sequential data sets ..... 120
Moving direct access data sets .......... 121
Moving GDG data sets .................... 121
   Moving generation data sets to
      SMS-managed volumes .................. 122
   Moving generation data sets to
      non-SMS-managed volumes ............. 122
   Moving SMS-managed data sets .......... 123
      Selecting target volumes .............. 123
      Changing storage class with Copy .... 124
      Changing management class with Copy 124
   Moving non-SMS-managed data sets ...... 125
   Moving to preallocated data sets ...... 125
      Rules for moving to preallocated target data sets 125
      Specifying multiple target volumes 127
      How keywords work with preallocated targets ......................................... 128
   Moving data sets being accessed with record level sharing 128
   Moving preformatted empty VSAM data sets 129
   VTOC considerations for moving volumes 129
Logical volume copy operation ............ 129
   Physical volume copy operation .......... 130
   Moving volumes with FlashCopy .......... 131
      Designating FlashCopy usage .......... 131
      Determining why FlashCopy cannot be used 132
      Freeing subsystem resources .......... 132
      Choosing space efficient FlashCopy with the
         FCSETGTOK keyword ................ 133
      Initializing the volume with the FCWITHDRAW
         keyword .............................. 134
      Backing up volumes with FlashCopy
         consistency group .................... 135
         Creating consistent copies with FlashCopy
         consistency group .................... 135
         Freezing the source volumes in copy
         operations ........................... 135
         Thawing the frozen volumes in CGCREATED
         operation ............................ 135
         Verifying the consistency group ....... 135
      Moving volumes with SnapShot .......... 137
         Designating SnapShot usage .......... 137
         Determining why SnapShot cannot be used .... 137
      Moving volumes to like devices of equal capacity 138
      Moving volumes to like devices of greater capacity 138
      Moving volumes to unlike devices ...... 139
      Moving VM-format volumes .......... 139
Chapter 8. Converting data to and from SMS management ........ 141
   Evaluating conversion to SMS management 141
      Data sets ineligible for conversion to SMS .................................. 141
      Data sets ineligible for conversion from SMS .................................. 142
   Conversion by data movement ............ 142
      Converting to SMS management by data movement .................................. 142
      Conversion from SMS management by data movement .................................. 142
      Conversion without data movement .......................................................... 143
      Simulating conversion ................................................................. 143
      Preparing a volume for conversion ...................................................... 144
      Converting to SMS management without data movement .................................. 145
      SMS report ................................ 146
      Special data set requirements for conversion to SMS ................................. 146
         VSAM sphere eligibility ............... 146
         Multivolume data sets ............... 146
         GDG data sets .......................... 147
         Temporary data sets ............... 148
         VTOC and VVDS ........................ 148
      Converting from SMS management without data movement .................................. 148
      Special data set requirements for conversion from SMS .......................... 148
         Multivolume data sets ............... 148
         GDG data sets .......................... 149
         Temporary data sets ............... 149
         VTOC and VVDS ........................ 149
      Special considerations for using non-SMS-managed targets .......................... 149

Part 1. DFSMSdss Storage Administration Guide 3
Chapter 9. Managing space with DFSMSdss
Reclaiming DASD space. .......................... 151
Releasing unused space in data sets .......... 151
Compressing a PDS ............................. 152
Deleting unwanted data sets ..................... 152
Combining data set extents ....................... 153
Consolidating free space and extents on volumes 155
When to run DEFRAG and CONSOLIDATE functions ............ 155
Designating FlashCopy usage .................. 156
Preserve Mirror FlashCopy ...................... 156
Determining why FlashCopy cannot be used ........ 157
Designating SnapShot usage .................. 158
Determining why SnapShot cannot be used ....... 158
Data sets excluded from DEFRAG or CONSOLIDATE processing ..... 158
DEFRAG options ................................ 159
General hints ................................ 160
Serialization ................................ 161
Security considerations ......................... 164
Maximizing track utilization by reblocking data sets .......... 165

Chapter 10. Diagnosing problems in DFSMSdss operations
Determining the source of the failure: DFSMSdfp, DFSMSdss, or DFSMShsm ............ 167
Using keywords to identify the problem ........ 168
Component identification keyword .............. 168
Release-level keyword ........................ 169
Type-of-failure and function keywords ......... 169
ABENDxxx ................................ 169
MSGADReerr ................................ 170
WAIT ...................................... 171
LOOP ........................................ 171
INCORROUT ................................ 172
DOC ......................................... 173
PERFM ....................................... 173
Module keyword ................................ 174
Maintenance-level keyword ..................... 176
Using the IBM Support Center ................. 176
Using the software support facility ............ 176
Using IBMLink/ServiceLink ..................... 177
Info/System ................................ 177

Chapter 11. ACS routine information
ACS variables available during Copy function ............. 179
ACS variables available during RESTORE and CONVERT processing ............ 181
Using SIZE and MAXSIZE variables ............. 182

Chapter 12. Dumping and restoring Linux for zSeries partitions and volumes
Preparing to work with Linux volumes .............. 185
Understanding the hardware environment .......... 185
Choosing VOLSERs for Linux volumes ............ 186
Formatting and partitioning Linux volumes .......... 186
Using dasdfmt to format a Linux volume .......... 186
Using fdasd to partition a Linux volume .......... 187
Obtaining authorization for Linux volumes .......... 187
Backing up a Linux volume with partitions .......... 187
Using DFSMSdss dump and restore commands .......... 188
Example 1. DUMP FULL .......................... 189
Example 2. DUMP FULL with CONCURRENT COPY ................ 190
Example 3. DUMP DATASET ...................... 190
Example 4. COPY FULL .......................... 191
Example 5. COPY FULL COPYVOLID ALLEXCP .............. 192
Example 6. RESTORE FULL ...................... 192
Example 7. RESTORE DATASET .................. 193
Example 8. COPYDUMP .......................... 195
Submitting JCL batch jobs to a z/OS system using FTP ............ 195
Using DFSMSdss stand-alone services ............. 195

Chapter 13. Format of the DFSMSdss dump data set
Format of the DFSMSdss dump data set ............. 197
ADRBMB data area ................................ 198
ADRBMB constants ................................ 198
ADRBMB cross-reference ........................ 199
ADRTAPB data area ................................ 199
ADRTAPB constants ................................ 208
ADRTAPB cross-reference ........................ 208

Chapter 14. DFSMSdss patch area
Sample JCL ................................... 215
Forcing the use of preallocated VSAM data sets (PN04574) .......... 215
Ignoring VSAM duplicate key errors (PN05539) .......... 216
Modifying the timeout period for enqueue lockout detection (PL84514) .......... 217
Controlling the wait/retry time for serialization of system resources (PN11523) .......... 217
Using CONVERT on data sets with a revoked user ID in the RESOWNER field (OY59957) .......... 218
Restoring inconsistent PDSE data sets (OY60301) .......... 218
Changing default protection status during RESTORE (PN37489) .......... 219
Restoring or copying undefined, multivolume SMS-managed data sets (OY63818) .......... 219
Bypassing backup-while-open processing (OY63531) .......... 220
Bypassing storage and management class authorization checking during RESTORE (OY65348) .......... 220
Issuing notification for tape and migrated data sets (OY66092) .......... 221
Using RESET with concurrent copy (OY65555) .......... 221
Forcing RESTORE after message ADR482E (OY67532) .......... 222
Restoring VSAM KSDS or VRRDS after messages ADR789W, ADR364W, and ADR417W (OY67942) .......... 222
Restoring VSAM data sets with expiration date of 1999365 (OW00780) .......... 223
Restoring VSAM data sets with expiration dates beyond 2000 (OW00780) .......... 223
Changing default insertion of EOF track during COPY with ALLDATA specified (OW15003) .......... 224
Using RESET or UNCATALOG in a logical data set dump (PN60114) .......... 224
|
|

|
|

Changing secondary allocation quantity in format 1
DSCB for PDSE data sets (OW07755) . . . . .
Changing reference date default settings during
data set COPY and RESTORE processing
(OW12011) . . . . . . . . . . . . . .
Changing default protection processing during
COPY (OW10314) . . . . . . . . . . . .
Bypassing management and storage class access
checks during COPY (PN72592) . . . . . . .
Changing default handling of invalid tracks
created during data set COPY and RESTORE
processing (OW08174) . . . . . . . . . .
Forcing RESTORE to the same volumes as the
source VSAM data set (OW07077) . . . . . .
Modifying number of volumes allocated for SMS
data sets during logical RESTORE and COPY
(OW15880) . . . . . . . . . . . . . .
Dumping a keyed VSAM data set that has data
CAs without corresponding index CIs (OW17877) .
Changing the default DEFRAG processing of
checkpointed data sets (OW20285) . . . . . .
Setting the percentage to overallocate target data
set space (OW27837) . . . . . . . . . . .
Bypassing RLS processing (OW32817) . . . . .
Changing creation date default settings during
logical data set COPY and RESTORE (OW19618) .
Copying and dumping a PDSE data set using the
VALIDATE PDSE option (OW48074). . . . . .
Changing the default maximum number of active
parallel subtasks . . . . . . . . . . . .
Changing the default initialization processing
during DUMP with FCWITHDRAW (OA18929) . .
Changing the default DEFRAG processing of
LINKLIST-indicated data sets (OW43874) . . . .
Changing the FASTREPLICATION default setting
during Copy and Defrag (OA11637) . . . . . .
Tuning hardware assisted compression (OA13300)
Resetting the data-set-changed indicator during
physical full or partial RESTORE operation
(OA20907) . . . . . . . . . . . . . .
Requesting that DFSMSdss double-check data set
high used RBA values for LDS data sets (OA23805)
Enabling or disabling use of the catalog search
interface for data set name filtering . . . . . .
Requesting that DFSMSdss restore the VM
formatted volume that was DUMPed by z/OS
V1R10 before OA27531 was applied. . . . . .
Adding timestamps to messages . . . . . . .
Enabling building appropriate channel programs
ADRPTCHB data area . . . . . . . . . .
ADRPTCHB cross-reference . . . . . . .

225

226
227
227

228
228

229
230
231
231
232
233
233
234
234
235
236
236

237
238
238

239
239
239
240
242

Part 1. DFSMSdss Storage Administration Guide

5


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 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 DFSMSdfp 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.

Notes:
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.

**Related reading:** For more information about these messages, see [z/OS MVS System Messages, Vol 1 (ABA-AOM)](https://www.ibm.com).  

**Installation exit routines**

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

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

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

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

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

**Related reading:** For more information about these exit routines, see [z/OS DFSMS Installation Exits](https://www.ibm.com).

**Authorization checking**

**Related reading:** For information about authorization checking, see [Chapter 20, “Data security and authorization checking,” on page 529](https://www.ibm.com).

**Managing availability with DFSMSdss**

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

There are two general forms of backup:

- **Data set backup**  
  Protects against the loss of individual data sets

- **Volume backup**  
  Protects against the loss of a volume.

For data set backup, you can perform incremental backups to help reduce processing time while still meeting your backup requirements. Incremental backup means that data sets are backed up only if they have changed since their last backup.
Volume backups are used to protect against media failure. You can use volume backups in conjunction with incremental data set backups to recover a volume. As a result, you need not do volume backups as frequently. Incremental data set backups should be done daily and volume backups weekly. If a volume is lost for some reason, you can restore from the most recent volume backup and apply incremental data set backups to the volume to bring it back to its most current status.

Notes:
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.

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

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

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

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

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

Notes:
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.

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 "flashed" 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 "flashed" 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 59.
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 Chapter 19, “DFSMSdss stand-alone services,” on page 509.

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 DFSMShsm 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 317.

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 141.

### 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 141.

### 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.

**Related reading:** For more information about managing space, see Chapter 9, “Managing space with DFSMSdss,” on page 151.
Chapter 2. Requirements for running DFSMSdss

This topic describes the requirements for running DFSMSdss.

Understanding the operating environment

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

You can run the DFSMSdss stand-alone restore program on an IBM System 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

In most cases, you can let DFSMSdss determine the amount of storage it uses for an operation. Sometimes, however, you might want closer control of the amount of storage DFSMSdss uses. Use the storage estimates in this topic as a starting point for determining minimum region sizes in which DFSMSdss can run. Table 1 and Table 2 on page 16 show the minimum storage requirements, in bytes, to run each DFSMSdss operation. Table 3 on page 17 shows the minimum storage requirements, in bytes, to restore partitioned and VSAM data sets to unlike devices. The legend for Table 1 on page 16, Table 2 on page 16, and Table 3 on page 17 is displayed beneath Table 3 on page 17. The values include the storage required to load the DFSMSdss program into the region.

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

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

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

If you use the PARALLEL command to run two or more DFSMSdss tasks concurrently, the total storage required is the sum of the storage required for all functions to be run in parallel. However, because DFSMSdss is reentrant, the DFSMSdss code is not duplicated in storage. Therefore, do not include the DFSMSdss load module size more than once.
Table 1. Minimum Storage Requirements for DFSMSdss Operations with I/O Buffers below 16-Megabyte Virtual

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

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

<table>
<thead>
<tr>
<th>DFSMSdss Command</th>
<th>Storage Below 16MB Virtual</th>
<th>Storage Above 16MB Virtual</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPRESS</td>
<td>dsssize + (number of additional volumes, up to five * copysize)</td>
<td>2 * trksize of largest device</td>
</tr>
<tr>
<td>CONVERTV</td>
<td>dsssize</td>
<td>buffersize * 5</td>
</tr>
<tr>
<td>COPY (FULL)</td>
<td>dsssize</td>
<td>trksize * 5</td>
</tr>
<tr>
<td>COPYDUMP</td>
<td>dsssize</td>
<td>buffersize * 5</td>
</tr>
<tr>
<td>DEFRAG</td>
<td>dssize + (16KB if VSAM present) + (1KB * number of non-VSAM entries in VVDS)</td>
<td>(trksize * 5) + (152 * number of data set extents) + (144 * number of VSAM components) + (48 * number of non-VSAM data sets)</td>
</tr>
<tr>
<td>DUMP (FULL) OPT(1)</td>
<td>dsssize</td>
<td>buffersize * 7</td>
</tr>
<tr>
<td>DUMP (FULL) OPT(2)</td>
<td>dssize</td>
<td>buffersize * 6</td>
</tr>
<tr>
<td>DUMP (FULL) OPT(3)</td>
<td>dssize</td>
<td>buffersize * 15</td>
</tr>
<tr>
<td>DUMP (FULL) OPT(4)</td>
<td>dssize</td>
<td>buffersize * (3 * trk/cyl)</td>
</tr>
<tr>
<td>PRINT</td>
<td>dsssize</td>
<td>3 * trksize</td>
</tr>
<tr>
<td>RELEASE</td>
<td>dsssize</td>
<td>trksize</td>
</tr>
<tr>
<td>RESTORE (FULL)</td>
<td>dssize</td>
<td>buffersize * 6</td>
</tr>
</tbody>
</table>
Table 3. Minimum Storage Requirements for a Restore to an Unlike Device

<table>
<thead>
<tr>
<th>Type of Data Set</th>
<th>Storage Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioned Data Sets</td>
<td>dsssize + (((trksize + 64) * 5) + 8KB)</td>
</tr>
<tr>
<td>VSAM Data Sets</td>
<td>dsssize + (((trksize + 64) * 5) + (2 * maximum record size) + (3 * buffspace))</td>
</tr>
</tbody>
</table>

Legend: This legend applies to Table 1 on page 16, Table 2 on page 16, and Table 3.

<table>
<thead>
<tr>
<th>KB</th>
<th>1024 bytes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsssize</td>
<td>DFSMSdss load module size, 2065KB.</td>
</tr>
<tr>
<td>copysize</td>
<td>IEBCOPY load module size + IEBCOPY storage requirements below the 16MB line (that is, 1MB minimum, or 2MB if the data set being compressed has more than 1000 members).</td>
</tr>
<tr>
<td>buffersize</td>
<td>64KB for output to tape unless DFSMSdss is forced to a smaller blocksize. The maximum of the largest trksize used and a blocksize of 32KB.</td>
</tr>
<tr>
<td>buffspace</td>
<td>Buffer space specified in the DEFINE command when the data set was allocated.</td>
</tr>
<tr>
<td>trksize</td>
<td>The size, in bytes, of a track on your DASD volume.</td>
</tr>
<tr>
<td>trks/cyl</td>
<td>The number of tracks per cylinder on your DASD volume.</td>
</tr>
</tbody>
</table>

Hardware requirements

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

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

Volume formats

You can use DFSMSdss with the following DASD volume formats:
- Volumes with indexed VTOCs, see "Indexed VTOC" on page 18 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.

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)
  - HFS (hierarchical file system) data set
- Sequential, including extended-format data sets and Large Format data sets
- VSAM data sets that are cataloged in an ICF catalog, including:
  - ESDS (entry-sequenced data set)
  - KSDS (key-sequenced data set)
  - KSDS with key ranges
  - LDS (linear data set)
  - RRDS (relative record data set)
  - VRRDS (variable relative record data set)
  - Extended-format ESDS, KSDS, LDS, RRDS, and VRRDS, including striped ESDS, KSDS, LDS, RRDS, and VRRDS
  - Extended-addressable VSAM ESDS, KSDS, LDS, RRDS, and VRRDS, including striped ESDS, KSDS, LDS, RRDS, and VRRDS
  - zFS (zSeries® file system) data set
- Unmovable data set types (PSU, POU, DAU, ABSTR, ISU, and direct with OPTCD=A).

**Notes:**

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

**Temporary data set names**

DFSMSdss must allocate the following temporary data sets to perform certain functions such as copy and restore. The high-level qualifiers of those data set names can be protected, and your installation must ensure that these temporary data sets can be allocated.
Message data set—
Allocated by DFSMSdss to store messages. This data set lets DFSMSdss print out messages by task rather than intermixing them. This data set is deleted when DFSMSdss completes the operation. System-generated temporary names are used.

Special DEFrag 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.xxxxxxx.volser.DUMMY

where xxxxxxx 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:

<table>
<thead>
<tr>
<th>Number of qualifiers ((n))</th>
<th>Temporary name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dsnh1q.Atidasid.chmmsstt</td>
</tr>
<tr>
<td>2</td>
<td>First 2 qualifiers.Atidasid.chmmsstt</td>
</tr>
<tr>
<td>&gt;2</td>
<td>First 3 qualifiers.Atidasid.chmmsstt</td>
</tr>
</tbody>
</table>

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

The last qualifier is chmmsstt where \(c\) is:

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

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

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

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

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

```
CATHLQ.EXPORT.hhmmssstit
```

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.hhmmssstit
```

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.

Defining logical and physical processing

DFSMSdss can perform two kinds of processing when executing COPY, DUMP, and RESTORE commands:

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

Each type of processing offers different capabilities and advantages.

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

Logical processing

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

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

DFSMSdss performs logical processing if:

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

DFSMSdss uses catalogs or VTOCs to select data sets for logical processing. If you do not specify input volumes, DFSMSdss uses the catalogs to select data sets for copy and dump operations. If you specify input volumes using the LOGINDDNAME, LOGINDYNAM, or STORGRP keywords on the COPY or DUMP command, DFSMSdss uses VTOCs to select data sets for processing.
**Note:** To copy or dump entire multivolume data sets, you do not need to specify all the volumes in the LOGINDDNAME or LOGINDYNAM volume list. However, you must specify the SELECTMULTI keyword with either the FIRST or ANY subkeywords.

**When to use logical processing**

Use logical processing for the following situations:

- Data is copied to an unlike device type.
  Logical processing is the only way to move data between unlike device types.
- Data that may need to be restored to an unlike device is dumped.
  Data must be restored the same way it is dumped. This is particularly important to bear in mind when making backups that you plan to retain for a long period of time (such as vital records backups). If a backup is retained for a long period of time, it is possible that the device type it originally resided on will no longer be in use at your site when you want to restore it. This means you will have to restore it to an unlike device, which can be done only if the backup was made logically.
- Aliases of VSAM user catalogs are to be preserved during copy and restore functions.
  Aliases are not preserved for physical processing.
- Unmovable data sets or data sets with absolute track allocation are moved to different locations.
- Multivolume data sets are processed.
- VSAM and multivolume data sets are cataloged as part of DFSMSdss processing.
- Data sets are deleted from the source volume after a successful dump or copy operation.
- Non-VSAM and VSAM data sets are renamed after a successful copy or restore operation.
- You want to 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.

**Notes:**

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

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

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

- You specify the DATASET keyword on the COPY or DUMP command and input volumes with the PHYSINDDNAME 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:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOC</td>
<td>Allocation type (cylinder, track, block, absolute track, or movable)</td>
</tr>
<tr>
<td>CATLG</td>
<td>Whether a data set is cataloged or not (using the standard catalog search order)</td>
</tr>
<tr>
<td>CREDT</td>
<td>Creation date (absolute or relative)</td>
</tr>
<tr>
<td>DATACLAS</td>
<td>Data class for SMS</td>
</tr>
<tr>
<td>DSCHA</td>
<td>Whether the data-set-changed indicator is on</td>
</tr>
</tbody>
</table>
DSORG  Data set organization (SAM, PAM, PDS, PDSE, BDAM, EXCP, HFS, ISAM, VSAM, or zFS)
EXPDT  Expiration date (absolute or relative)
EXTNT  Number of extents
FSIZE  Data set size (number of allocated or used tracks)
MGMTCLAS  Management class for SMS
MULTI  Whether the VTOC shows that the data set is single-volume or multivolume (allocated single-volume data sets that have never been opened and are not cataloged may be selected as multivolume).
REFDT  Last-referenced date (absolute or relative)
STORCLAS  Storage class for SMS

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ or =</td>
<td>Equal to</td>
</tr>
<tr>
<td>LT or &lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>LE or &lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>GT or &gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>GE or &gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>NE or ≠</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

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:

```plaintext
//VRPAY JOB Accounting Information,MORGAN
//STEP1 EXEC PGM=ADDRDSSU,REGION=4000K
//SYSPRT DD SYSOUT=* /*
//DROUT DD DSN=PAYROLL.DAY1,DISP=(NEW,CATLG),UNIT=3480,LABEL=(I,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).

Related reading: For more information about setting up data set naming conventions, see [MVS/ESA SML: Managing Data](#).
Chapter 4. Invoking DFSMSdss

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

Invoking DFSMSdss with ISMF

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

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

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

How to invoke ISMF

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

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 253.

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 22, “Application programming interface,” on page 575.

### 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 23, “Examples of the application program with the user interaction module (UIM),” on page 619.
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

The names of the load modules for DFSMSdss/ISMF are stored in command tables in both the panel library, DGTPLIB, and the load library, DGTLLIB. The load module names are listed in Table 4 and Table 5. The module names are found in the DGTSMM01 member of the panel library.

Table 4 lists the names for the corresponding line operators. The module names for line operators are found in the DGTTLP03 member of the load library. Table 5 lists the names for commands. These names are in the DGTTCTD2 member of the load library.

Table 4. Module Names for DFSMSdss/ISMF Line Operators

<table>
<thead>
<tr>
<th>Line Operator</th>
<th>Data Set Application Module Name</th>
<th>Volume Application Module Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGCREATE</td>
<td></td>
<td>DGTFCG01</td>
</tr>
<tr>
<td>COMPRESS</td>
<td>DGTFCM01</td>
<td>DGTFCS01</td>
</tr>
<tr>
<td>CONSOLID</td>
<td></td>
<td>DGTFCI01</td>
</tr>
<tr>
<td>CONVERTV</td>
<td></td>
<td>DGTFCN01</td>
</tr>
<tr>
<td>COPY</td>
<td>DGTFCY01</td>
<td>DGTFCV01</td>
</tr>
<tr>
<td>DEFRAO</td>
<td></td>
<td>DGTFRD01</td>
</tr>
<tr>
<td>DUMP</td>
<td>DGTFDI01</td>
<td>DGTFDM01</td>
</tr>
<tr>
<td>RELEASE</td>
<td>DGTFRDL01</td>
<td>DGTFRV01</td>
</tr>
<tr>
<td>RESTORE</td>
<td>DGTFRTO1</td>
<td>DGTFRR01</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Module Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPRESS</td>
<td>DGTFCP01</td>
</tr>
<tr>
<td>COPY</td>
<td>DGTFCO01</td>
</tr>
<tr>
<td>DUMP</td>
<td>DGTFDU01</td>
</tr>
<tr>
<td>RELEASE</td>
<td>DGTFRD01</td>
</tr>
<tr>
<td>RESTORE</td>
<td>DGTFRR00</td>
</tr>
</tbody>
</table>

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

End of Programming Interface information

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

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

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

2. To allow users to execute an application, line operator, or command, use the PERMIT command.

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 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 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.

Table 6. RACF FACILITY Class Profile Names for DFSMSdss Keywords

<table>
<thead>
<tr>
<th>Keyword or Function</th>
<th>Profile Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYPASSACS with COPY</td>
<td>STGADMIN.ADR.COPY.BYPASSACS</td>
</tr>
<tr>
<td>BYPASSACS with RESTORE</td>
<td>STGADMIN.ADR.RESTORE.BYPASSACS</td>
</tr>
<tr>
<td>CGCREATED</td>
<td>STGADMIN.ADR.CGCREATE</td>
</tr>
<tr>
<td>CONCURRENT with COPY</td>
<td>STGADMIN.ADR.COPY.CNCURRNT</td>
</tr>
<tr>
<td>CONCURRENT with DUMP</td>
<td>STGADMIN.ADR.DUMP.CNCURRNT</td>
</tr>
<tr>
<td>CONSOLIDATE</td>
<td>STGADMIN.ADR.CONSOLID</td>
</tr>
<tr>
<td>CONVERTV</td>
<td>STGADMIN.ADR.CONVERTV</td>
</tr>
<tr>
<td>DEFRAG</td>
<td>STGADMIN.ADR.DEFRAG</td>
</tr>
</tbody>
</table>
### Table 6. RACF FACILITY Class Profile Names for DFSMSdss Keywords (continued)

<table>
<thead>
<tr>
<th>Keyword or Function</th>
<th>Profile Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETECATALOGENTRY with RESTORE</td>
<td>STGADMIN.ADR.RESTORE.DELCATE</td>
</tr>
<tr>
<td>FCCEFREEZE with COPY</td>
<td>STGADMIN.ADR.COPY.FCCEFREEZE</td>
</tr>
<tr>
<td>FCFASTIVERSERESTORE with COPY</td>
<td>STGADMIN.ADR.COPY.FCFRR</td>
</tr>
<tr>
<td>FCSETGQTOK with COPY</td>
<td>STGADMIN.ADR.COPY.FCSETGT</td>
</tr>
<tr>
<td>FCTOPPRCP with COPY</td>
<td>STGADMIN.ADR.COPY.FCTOPPRCP</td>
</tr>
<tr>
<td>FCTOPPRCPRIM with DEFRAST</td>
<td>STGADMIN.ADR.DEFRAG.FCTOPPRCP</td>
</tr>
<tr>
<td>FlashCopy with CONSOLIDATE</td>
<td>STGADMIN.ADR.CONSOLID.FLASHCPY</td>
</tr>
<tr>
<td>FlashCopy with COPY</td>
<td>STGADMIN.ADR.COPY.FLASHCPY</td>
</tr>
<tr>
<td>FlashCopy with DEFRAST</td>
<td>STGADMIN.ADR.DEFRAG.FLASHCPY</td>
</tr>
<tr>
<td>IMPORT with RESTORE</td>
<td>STGADMIN.ADR.RESTORE.IMPORT</td>
</tr>
<tr>
<td>INCAT(catname) with COPY</td>
<td>STGADMIN.ADR.COPY.INCAT</td>
</tr>
<tr>
<td>INCAT(catname) with DUMP</td>
<td>STGADMIN.ADR.DUMP.INCAT</td>
</tr>
<tr>
<td>INCAT(catname) with RELEASE</td>
<td>STGADMIN.ADR.RELEASE.INCAT</td>
</tr>
<tr>
<td>NEWNAMEUNCONDITIONAL with DUMP</td>
<td>STGADMIN.ADR.DUMP.NEWNAME</td>
</tr>
<tr>
<td>PROCESS(SYS1) with COPY</td>
<td>STGADMIN.ADR.COPY.PROCESS.SYS</td>
</tr>
<tr>
<td>PROCESS(SYS1) with DUMP</td>
<td>STGADMIN.ADR.DUMP.PROCESS.SYS</td>
</tr>
<tr>
<td>PROCESS(SYS1) with RELEASE</td>
<td>STGADMIN.ADR.RELEASE.PROCESS.SYS</td>
</tr>
<tr>
<td>TOLERATE(ENQF) with COPY</td>
<td>STGADMIN.ADR.COPY.TOLERATE.ENVQF</td>
</tr>
<tr>
<td>TOLERATE(ENQF) with DUMP</td>
<td>STGADMIN.ADR.DUMP.TOLERATE.ENVQF</td>
</tr>
<tr>
<td>TOLERATE(ENQF) with RESTORE</td>
<td>STGADMIN.ADR.RESTORE.TOLERATE.ENVQF</td>
</tr>
</tbody>
</table>

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](https://www.ibm.com/support/docview.wss?uid=swg27010131).


### 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 33.
Chapter 6. Managing availability with DFSMSdss

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

Planning an availability strategy

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

- **Backup of volumes and data sets**—The general type of backup to guard against users accidentally losing or incorrectly changing their data sets and against losing volumes because of hardware failures.
- **Disaster recovery backup**—Backup to protect against the loss of all your data in a major disaster at your site. These backups are stored off site and, in the event of a major disaster, are recovered at another site.
- **Vital records backup**—Backup copies of data sets kept to meet externally imposed retention requirements, such as tax records.
- **Archival**—Backup of data that is unused for a long period of time. You remove the data from DASD and retain it on tape in case it is needed again.

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

Backup and recovery

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

Data set backup

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

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

For a more detailed discussion of considerations for determining frequency of backup and number of versions, refer to [MVS/ESA SML: Managing Data](#).

Volume backup

Volume backup is necessary to guard against losing a volume, but it need not be done often if you are doing incremental backup on a regular basis. If you lose a volume, you can recover from the latest volume backup, and then recover data sets.
from incremental backups to return the volume to its status before the failure. This
form of recovery is sometimes referred to as forward recovery. 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.

For more details on planning for backup of SMS-managed data, refer to[MVS/ESA
SML: Managing Data](#).

**Non-SMS-managed data:** Typically, non-SMS-managed data is data that SMS does
not support or data that is in transition from non-SMS to SMS management. If it is
data that SMS does not support, you can probably still use DFSMSdss to back it up
and recover it, because DFSMSdss supports many kinds of data that SMS does not.
If it is data in transition to SMS management, you can use DFSMShsm or
DFSMShsm 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.
Note: You can also automatically transmit backups to another site.

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

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

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

If for some reason you must do volume dumps for disaster recovery, you should do logical volume dumps instead of physical volume dumps. That way, you can restore the backups to unlike devices. You can perform logical volume dumps by using DATASET (INCLUDE(**)) and either the LOGINDDNAME or LOGINDYNAME 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 91.

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

For more information about disaster recovery, refer to MVS/ESA SML: Managing Data.
Maintaining vital records

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

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

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

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

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

For more information about vital records, refer to MVS/ESA SML: Managing Data.

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.

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((REFTDT LT 99100)(MGMTCLAS EQ MCNAME1))) -
OUTDDNAME(TAPE1) -
DELETE -
COMPRESS -
PURGE
```

For more information about archiving, refer to MVS/ESA SML: Managing Data.
Backing up data sets

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

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

You can select data sets for dump processing based on data set names and numerous data attributes, as discussed in Chapter 16, “DFSMSdss filtering—choosing the data sets you want processed,” on page 263.

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 42.

Notes:
1. If you are using DFSMSdss on data sets that DFSMShsm is also backing up, you should not use the RESET keyword because it might cause confusion as to which backup is the most current.
2. DFSMSdss does not permanently record the names of candidate volumes during dump processing.

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 

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:

```plaintext
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:

```plaintext
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 example, the following DUMP command results in a physical data set dump:
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 102). 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.

Notes:
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 199.

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 413.

Backup with concurrent copy

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 of 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 57.

**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 "flashed" 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 "flashed" 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 11.
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.

Notes:
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:CNCURRNT.

For information about specifying CONCURRENT and other DUMP command keywords, refer to "DUMP command for DFSMSdss" on page 391.

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 317.

Using DFSMSdss as a backup utility for CICSVR
CICSVR users can choose DFSMSdss as their backup utility by specifying the CICSVRBACKUP keyword. DFSMSdss notifies the CICSVR server address space every time that a CICSVR backup is made for a VSAM base cluster. CICSVR stores the backup information in its recovery control data set (RCDS). This enables CICSVR to manage backups that are made by DFSMSdss. Through the CICSVR dialog panels, CICSVR 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. 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 401.

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 35, 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 INDYNAME(111111) OUTDD(TAPE1) RESET OPTIMIZE(1)
  DUMP INDYNAME(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)) OUTD(TAPE4) RESET -
  OPTIMIZE(4) DATASET(INCLUDE(CRIT=.*)) -
  BY(DSCHA,SEQ,YES)
```

Other naming conventions can also be used to identify groups of data sets. For instance, you can use department numbers, charge numbers, user initials, or project codes to identify data sets you want to dump together.

For more information about naming conventions, refer to [MVS/ESA SML: Managing Data](#).

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

### Backing up data sets with special requirements

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

#### Dumping HFS data sets

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

**Logical dump**

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

For more information about the serialization of HFS data sets, refer to [Chapter 21, “Data integrity—serialization,” on page 559](#).

**Physical dump**

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

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

#### Dumping zFS data sets

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

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

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

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

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

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

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:
  - Specify SPHERE and you must list all parts of the base cluster in the volume list.
  - Do not specify SPHERE and you must list all parts of the base cluster and the associated indexes in the volume list.

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

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:
  - Specify SPHERE and you must list any part of the base cluster in the volume list.
  - Do not specify SPHERE and you must list any part of the base cluster and the associated alternate indexes in the volume list.

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

  For VSAM data sets, the volume list is affected by the use of the SPHERE keyword as follows:
  - Specify SPHERE and you must list the volume that contains the first extent of the data component for the base cluster in the volume list.
- Do not specify SPHERE and you must specify the following information in the volume list:
  - The volume that contains the first extent of the data component for the base cluster.
  - The volume that contains the first extent of the data component for the associated alternate indexes.

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

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

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

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

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

DFSMSdss cannot process the following non-VSAM data sets because they are missing one or more parts:
- Multivolume data sets whose catalog volume order differs from the VTOC volume order
- Single volume data sets with the same name that are cataloged as one multivolume data set
- Multivolume data sets whose last volume indicator in the VTOC entry is not set.

A multivolume data set standard user label is not supported.

**Dumping integrated catalog facility user catalogs**

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

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

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

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

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

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

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

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

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

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 39. 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(nn.nn) command.
For more information about the RLS timeout value used during DFSMSdss operations, refer to [z/OS DFSMSdfp 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 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 zSeries partitions and volumes, refer to Chapter 12, “Dumping and restoring Linux for zSeries partitions and volumes,” on page 185.

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

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

**Back 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 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.

**Back 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.

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

**Combining volume copy and volume dump to reduce your
backup window**

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

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

For more information about full volume copy, FlashCopy, and SnapShot, refer to
“Moving volumes” on page 99.
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 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
- 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, using the FCNOCOPY keyword on the full volume copy and the FCWITHDRAW keyword on the full volume dump leaves the target volume (of the copy) in an indeterminate state. Some tracks on the volume might contain data from the source volume; other tracks might contain residual data from the target volume that existed before the copy. This indeterminate state can cause problems when accessing the target volume following the dump, if the VTOC locations of the source volumes and the target volumes are different before the copy. To avoid this problem, do one of the following:
  - Ensure that the VTOC locations for the source volumes and the target volumes are the same before you initiate the copy.
- Add an ICKDSF INIT step for the target volume of the COPY in step 2. Add this step after step 4 (Backup the copies to tape using full volume dump). The target volume of the copy initializes and returns to a consistent state.

- The FCWITHDRAW keyword is not supported for volumes attached at device address X'0000'.

For more information about the DUMPCONDITIONING keyword, refer to "DUMPCONDITIONING" on page 326.

For more information about the FCNOCOPY keyword, refer to "FCNOCOPY" on page 333.

For more information about the FCWITHDRAW keyword, refer to "FCWITHDRAW" on page 405.


**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 ALLEXCP or ALLDATA) instead of all allocated space, in sequential and partitioned data sets or in data sets with a null DSORG field. For VSAM key sequenced data sets, the VALIDATE keyword (the default) dumps only the used data instead of all of the allocated space.

- Use the COMPRESS keyword.

**Notes:**

1. DFSMSdss ignores the COMPRESS keyword if you specify it during a logical data set dump for physical sequential extended-format data sets.

2. If your tape drive has the improved data recording capability (IDRC) and you want to use hardware data compaction, you do not need to use the COMPRESS keyword with the DUMP command. If you want software
... compression, specify the COMPRESS keyword, but you do not need to specify DCB=TRTCH=COMP in the JCL. In most cases, hardware data compaction without software data compression gives the best performance. However, you can use software compression and hardware compaction at the same time.

- Perform incremental data set backup instead of volume backup. This reduces the amount of dumped data and decreases processing time.
- 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 "flashed" 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 "flashed" 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.Sysname.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.Sysname.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.Sysname.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](https://www.ibm.com/docs/en/zos.Interopment?topic=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.

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. 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.

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.

Notes:
1. Incremental FlashCopy relationship is limited to one full volume incremental relationship per volume. However, an Incremental FlashCopy relationship can coexist with other non-incremental FlashCopy relationships. Other limits remain, such as each source track can have up to 12 targets.
2. Incremental FlashCopy is only possible with a persistent relationship. With persistent relationships, the relation between the source and target is maintained after the background copy has completed.
3. DFSMSdss allows you to copy full volumes to target devices of greater capacity. However, if the original FlashCopy target volume is bigger than the source volume, you 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.

  ```shell
  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 -
  FCINCRVERIFY(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:

```plaintext
COPY FULL INDDYN(VOL00A) OUTDDYN(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 330, “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.

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 password are chosen; thus, iteration counts of 1000 or higher are often normal.

Note:

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

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

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

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

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 INDD(TAPE1) OUTDYNAM(VOL001) -
ENCRYPT(CLRTDES) 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 summarizes the RSA private tokens and required cryptographic hardware for decryption.

**Table 7. RSA private tokens and required cryptographic hardware for decryption**

<table>
<thead>
<tr>
<th>RSA private key token (internal)</th>
<th>Required cryptographic hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA private key token 1024 — Modulus-Exponent Internal form</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• Cryptographic Coprocessor feature</td>
</tr>
<tr>
<td></td>
<td>• PCI X Cryptographic Coprocessor</td>
</tr>
<tr>
<td></td>
<td>• Crypto Express2 Coprocessor.</td>
</tr>
<tr>
<td>RSA private key token 1024 — Chinese Remainder Theorem Internal form</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• PCI Cryptographic Coprocessor</td>
</tr>
<tr>
<td></td>
<td>• PCI X Cryptographic Coprocessor</td>
</tr>
<tr>
<td></td>
<td>• Crypto Express2 Coprocessor.</td>
</tr>
<tr>
<td>RSA private key token 2048 — Chinese Remainder Theorem Internal form</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• PCI Cryptographic Coprocessor with LIC January 2005 or later, and z/OS ICSF HCR770B or later</td>
</tr>
<tr>
<td></td>
<td>• PCI X Cryptographic Coprocessor</td>
</tr>
<tr>
<td></td>
<td>• Crypto Express2 Coprocessor.</td>
</tr>
</tbody>
</table>

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) zSeries 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.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Method of Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear TDES</td>
<td>ICSF Service</td>
</tr>
<tr>
<td>Clear 128-bit AES</td>
<td>ICSF Service</td>
</tr>
<tr>
<td>Secure TDES</td>
<td>ICSF Service</td>
</tr>
</tbody>
</table>

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
- CSFSYE 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 the 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 not both methods. Because DFSMSdss avoids performing double encryption of tape data, you must determine which type of encryption, if any, is to be used for your tape volumes. In general, DFSMSdss prevents you from combining both types of encryption to perform double encryption of tape volumes.

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

**Table 8. DFSMSdss processing of dump encryption requests**

<table>
<thead>
<tr>
<th>Dump encryption request</th>
<th>DFSMSdss action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your DUMP command specifies host-based encryption (through the RSA or KEYPASSWORD keywords), and all of the available tape drives are encryption-capable tape drives. Your request might also specify host-based compression (through the HWCOMPRESS keyword).</td>
<td>• DFSMSdss ignores the compression request, if any.</td>
</tr>
<tr>
<td>Your DUMP command specifies host-based encryption and one or more of the available tape drives are not encryption enabled. Your request might also specify host-based compression.</td>
<td>• DFSMSdss issues informational message ADR518I to indicate that tape device encryption was used instead of host-based encryption.</td>
</tr>
<tr>
<td></td>
<td>• DFSMSdss ignores the compression request, if any.</td>
</tr>
<tr>
<td>Your DUMP command does not specify host-based encryption and all of the available tape drives are encryption-capable tape drives. Your request might also specify host-based compression.</td>
<td>• Encryption-capable tape drives perform encryption.</td>
</tr>
<tr>
<td></td>
<td>• DFSMSdss performs host-based encryption, if requested.</td>
</tr>
<tr>
<td>Your DUMP command does not specify host-based encryption and one or more of the available tape drives are not encryption enabled. Your request might also specify host-based compression.</td>
<td>• DUMP requests for encryption-capable tape drives are encrypted by the tape drives. It also requests encryption from the encryption-capable tape drives.</td>
</tr>
<tr>
<td></td>
<td>• DUMP requests for non-encrypting tape drives are processed without encryption of any type.</td>
</tr>
<tr>
<td></td>
<td>• DFSMSdss performs host-based encryption, if requested.</td>
</tr>
</tbody>
</table>

For tapes that require host-based encryption, ensure that your dump-requesting jobs use only 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 from the encryption-capable tape drives.
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, which are identified with the INDDNAME keyword.

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

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

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

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

As with a data set DUMP command, you can use filtering to select data sets for restore processing. DFSMSdss reads the entire dump data set once during a restore regardless of how much data is actually being restored. This will result in multiple tape mounts if the dump data set is on multiple tapes.

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 263.

For information about using automatic class selection (ACS) routines during DFSMSdss restore operations, refer to Chapter 11, “ACS routine information,” on page 179.

Logical data set restore

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

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

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

Output volume selection

In most cases, specifying output volumes is optional for a logical data set RESTORE command. Output volume specification is required only if the data set:
- Exists and is to be restored to a volume that is different from the current location
- Does not exist and is to be restored to a volume that is different from the source volume.

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

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

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

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

```plaintext
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 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 standard format sequential data sets to large format data sets. To do this, simply preallocate a large format data set. In restoring a standard 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 standard format sequential data set is found, the standard format sequential data set is used and is upgraded to a large format data set. If the preallocated standard format sequential data set is not large enough to hold the data being restored, it is scratched and reallocated as a large format data set.

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

You may also use data set RESTORE to upgrade your data sets that are not enabled for CA reclaim to data sets that are enabled for CA reclaim. To do this, simply preallocate a data set that is enabled for CA reclaim to use as the target of the restore. In restoring a data set that is not enabled for CA reclaim, if a preallocated data set that is enabled for CA reclaim is found, the preallocated data set that is enabled for CA reclaim is used for the target of the restore. If the preallocated data set that is enabled for CA reclaim is not large enough to hold the data being restored, it is scratched and reallocated as a data set that is enabled for CA reclaim. In restoring a data set that is enabled for CA reclaim, if a preallocated data set that is not enabled for CA reclaim is found, the data set that is not enabled for CA reclaim is used for the target of the restore. If the preallocated data set that is not enabled for CA reclaim is not large enough to hold the data being restored, it is scratched and reallocated as a data set that is not enabled for CA reclaim. For more information, refer to the topic on CA reclaim in z/OS DFSMS Using Data Sets.

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.

**Cataloging data sets during logical restore processing**

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

When a data set is restored as an SMS-managed data set, it is cataloged using the standard order of search. The RECATALOG keyword is ignored.
Examples of the CATALOG and RECATALOG keywords in a logical data set RESTORE command follow:

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

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

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

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

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

**Note:** The RENAME keyword works only if the data set exists on the output DASD with the old name. If you really want to unconditionally rename a data set, use 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 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.

<table>
<thead>
<tr>
<th>How is DFSMSdss invoked?</th>
<th>Is the data set renamed?</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoked directly to restore the data set.</td>
<td>No</td>
<td>DFSMSdss sets the data-set-changed indicator in the VTOC to the value the data set had when it was backed up.</td>
</tr>
<tr>
<td>Invoked directly to restore the data set.</td>
<td>Yes</td>
<td>DFSMSdss turns off the data-set-changed indicator for the data set because it is newly allocated.</td>
</tr>
<tr>
<td>Invoked by an application through the DFSMSdss API.</td>
<td>No</td>
<td>DFSMSdss leaves the data-set-changed indicator unmodified, so that other backup applications can continue to track the data set.</td>
</tr>
<tr>
<td>Invoked by an application through the DFSMSdss API.</td>
<td>Yes</td>
<td>DFSMSdss leaves the data-set-changed indicator unmodified, so that other backup applications can continue to track the data set.</td>
</tr>
</tbody>
</table>
Physical data set restore

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

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

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

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

The following data sets cannot be processed by physical data set DUMP or RESTORE operations:
- VSAM data sets not cataloged in an integrated catalog facility catalog.
- Page, swap, and SYS1.STGINDEX data sets.

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

Output volume selection

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

Cataloging data sets during physical restore processing

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

The catalog that DFSMSdss uses to catalog a data set is determined as follows:
- If the first qualifier of the data set name is an alias for a user catalog, the catalog pointed to is used for that data set.
- Otherwise, the master catalog is used.
DFSMSdss does not catalog VSAM data sets during physical restore processing. If the CATALOG keyword is specified, it is ignored when processing VSAM data sets. You should use the IDCAMS DEFINE RECATALOG command to catalog VSAM data sets that were allocated by DFSMSdss (not preallocated). To recatalog and later access the VSAM data set, the volume serial numbers for the target and source volumes must match and the data set must be cataloged in the same catalog from which it was dumped. The volume serial number and the catalog name are printed in message ADR4181 during restore.

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

### Coexistence considerations

For information about restoring dumps created with previous releases, refer to “Restoring backups using DFSMSdss” on page 639 in Appendix A, “Coexistence Considerations,” on page 639.

### Restoring data sets with special requirements

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

#### Restoring multivolume data sets and restoring data sets using multiple target volumes (spill volumes)

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

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

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

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

During logical restores of VSAM data sets whose data and index components are on different source volumes, DFSMSdss preserves the volume spread if enough target volumes of like device types are specified.
Note: DFSMSdss preserves the volume spread by placing the data and index components on separate devices only if all of the following are true:

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

For information about when DFSMSdss scratches and reallocates target data sets, refer to “Restoring to preallocated targets” on page 82.

Restoring integrated catalog facility catalogs

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

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

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

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

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

The following example shows the JCL used to restore an integrated catalog facility user catalog. If the master catalog is RACF-protected, RACF access to it is required, unless you have DASDVOL update access or the installation authorization exit routine bypasses authorization checking.
Figure 2 shows the printed output produced by the RESTORE command.

For more information about the LOCK attribute and the access authority, refer to z/OS DFSMS Managing Catalogs.

For more information about the installation authorization exit routine, refer to z/OS DFSMS Installation Exits.

Restoring non-VSAM data sets that have aliases

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

Restoring indexed sequential, unmovable, direct, and absolute track data sets

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

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:

```plaintext
RESTORE -
   INDDONE(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.

**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.
The following sections discuss variables available to automatic class selection (ACS) routines during DFSMSdss processing. This information is provided for guidance purposes only. It is not associated with any interface provided by DFSMSdss.

For information about writing ACS routines, refer to Chapter 17, “Syntax—DFSMSdss function commands,” on page 271.

Converting 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: 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.

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 181.

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 141.

For more information about ACS routines, refer to Chapter 11, “ACS routine information,” on page 179.

**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:

```plaintext
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:

```plaintext
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:

```plaintext
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:
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(=*))
   INDDYNAM((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 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
```

Notes:

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

For information about using DFSMSdss to restore Linux for zSeries partitions and volumes, see Chapter 12, “Dumping and restoring Linux for zSeries partitions and volumes,” on page 185.

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,cwu,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 ADHY0513I 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 ADHY509D to prompt the operator to continue or end the function.

For more information about how to perform a restore using DFSMSdss stand-alone services, see [Chapter 19, “DFSMSdss stand-alone services,” on page 509](#).

**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 639 in Appendix A, “Coexistence Considerations,” on page 639.
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.”](#)*

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 ALLEXCP. Use the ALLEXCP keyword 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 179 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 DUMP CONDITIONING 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:
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 123.)

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
Reduction of 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 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 with the RENAMEUNCONDITIONAL (RENAMEU) keyword for the COPY command. For VSAM data sets, you can only rename clusters. DFSMSdss derives the new names for the components of VSAM clusters as follows:

- If the data set is a linear data set and the new cluster name matches the following convention:
  
  HLQ1.DSNDBC.HLQ3.HLQ4.%nnnn.%nnn
  
  and the old component matches the following convention:
  
  HLQ1.DSNDBD.HLQ3.HLQ4.%nnnn.%nnn
  
  where $s$ 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 A0 for a data component, or to A1 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
  - DFSMSdss was invoked using the application interface
  - The UIM set the EI22DB2 bit ON
  - The new cluster matches the following convention:
    \[ \text{HLQ1.DSNDBC.HLQ3.HLQ4.%nnnn.%nnn} \]

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:

<table>
<thead>
<tr>
<th></th>
<th>Cluster Name</th>
<th>Data Component Name</th>
<th>Index Component Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>IBM.DFSMS.DSS</td>
<td>IBM.DFSMS.DSS.DAT1</td>
<td>IBM.DFSMS.DSS.INDX1</td>
</tr>
<tr>
<td>New</td>
<td>NEW.DFSMS.DSS</td>
<td>NEW.DFSMS.DSS.DAT1</td>
<td>NEW.DFSMS.DSS.INDX1</td>
</tr>
</tbody>
</table>

- 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:

<table>
<thead>
<tr>
<th></th>
<th>Cluster Name</th>
<th>Data Component Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>SYS1.10DF00.CLUSTER</td>
<td>SYS1.10DF00</td>
</tr>
<tr>
<td>New</td>
<td>SYS2.10DF00.CLUSTER</td>
<td>SYS2.10DF00</td>
</tr>
</tbody>
</table>

- 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:

<table>
<thead>
<tr>
<th></th>
<th>Cluster Name</th>
<th>Data Component Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>SYS1.10DF00.CLUSTER</td>
<td>SYS1.DFSMS</td>
</tr>
<tr>
<td>New</td>
<td>SYS2.10DF00.CLUSTER</td>
<td>SYS2.10DF00.DATA</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Cluster Name</th>
<th>Data Component Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>SYS1.IODF00.DATASET</td>
</tr>
<tr>
<td>New</td>
<td>SYS2.IODF00.DATASET.D</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Cluster Name</th>
<th>Data Component Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old</td>
<td>IBM.DFSMS.DSS.LARGE.VSAM.DSNAM.TEST</td>
</tr>
<tr>
<td></td>
<td>IBM.DFSMS.DSS.LARGE.VSAM.DSNAM.TEST.DATA</td>
</tr>
<tr>
<td></td>
<td>IBM.DFSMS.DSS.LARGE.VSAM.DSNAM.TEST.INDEX</td>
</tr>
<tr>
<td>New</td>
<td>NEWNAM.DFSMS.DSS.LARGE.VSAM.DSNAM.TEST</td>
</tr>
<tr>
<td></td>
<td>NEWNAM.DFSMS.DSS.LARGE.VSAM.DSNAM.TEST.DATA</td>
</tr>
<tr>
<td></td>
<td>NEWNAM.DFSMS.DSS.LARGE.VSAM.DSNAM.TEST.I</td>
</tr>
</tbody>
</table>

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 20, “Data security and authorization checking,” on page 529.

Moving data sets with utilities

In some cases, DFSMSdss invokes a utility to move a data set. Table 10 on page 106 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 10. Data Mover Selection Matrix for Data Set Copy

<table>
<thead>
<tr>
<th>Data Set Type</th>
<th>Like Devices</th>
<th>Unlike Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>DFSMSdss</td>
<td>DFSMSdss</td>
</tr>
<tr>
<td>Partitioned (not PDSE)</td>
<td>DFSMSdss (1, 2)</td>
<td>DFSMSdss (1, 2)</td>
</tr>
<tr>
<td>Partitioned (not PDSE) load modules</td>
<td>DFSMSdss (3)</td>
<td>IEBCOPY</td>
</tr>
<tr>
<td>Partitioned data set extended (PDSE)</td>
<td>DFSMSdss (4)</td>
<td>DFSMSdss (4)</td>
</tr>
<tr>
<td>Direct nonrelative block address mode</td>
<td>DFSMSdss</td>
<td>DFSMSdss (5)</td>
</tr>
<tr>
<td>Direct relative block address mode (6)</td>
<td>DFSMSdss</td>
<td>DFSMSdss</td>
</tr>
<tr>
<td>ESDS</td>
<td>DFSMSdss (7)</td>
<td>DFSMSdss (7, 8)</td>
</tr>
<tr>
<td>RRDS</td>
<td>DFSMSdss (7)</td>
<td>IDCAMS (REPRO)</td>
</tr>
<tr>
<td>LDS</td>
<td>DFSMSdss (7)</td>
<td>IDCAMS (REPRO)</td>
</tr>
<tr>
<td>KSDS or VRRDS</td>
<td>DFSMSdss (9)</td>
<td>IDCAMS (REPRO)</td>
</tr>
<tr>
<td>Key range data set</td>
<td>DFSMSdss (10)</td>
<td>IDCAMS (REPRO)</td>
</tr>
<tr>
<td>Extended-format VSAM</td>
<td>DFSMSdss (7)</td>
<td>IDCAMS (REPRO)</td>
</tr>
<tr>
<td>Integrated catalog facility user catalogs</td>
<td>IDCAMS (EXPORT/IMPORT)</td>
<td>IDCAMS (EXPORT/IMPORT)</td>
</tr>
<tr>
<td>Undefined DSORG</td>
<td>DFSMSdss</td>
<td>DFSMSdss</td>
</tr>
</tbody>
</table>
Table 10. Data Mover Selection Matrix for Data Set Copy (continued)

<table>
<thead>
<tr>
<th>Data Set Type</th>
<th>Like Devices</th>
<th>Unlike Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notes: 1. All partitioned data sets that are not load modules are compressed during a copy to a like or unlike device. 2. DFSMSdss calls the IGWFAMS utility when you are converting a PDS to a PDSE. 3. If copying partitioned load modules with REBLOCK, DFSMSdss calls IEB_COPY 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, see &quot;VOLCOUNT&quot; on page 360. • The target data set is non-SMS, the source component or components span multiple volumes, and there is not enough space on one target volume to contain the entire data set. 10. DFSMSdss calls IDCAMS if the source and target CASIZE, physical record size, or physical block size are different; if the components span multiple volumes; for a KSDS with IMBED and either the source HURBA=HARBA or it has extended indexes.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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, Eoption 24 (for more information, see "CONCURRENT" on page 317). The application can resume normal operation at this time.

End of 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 "flashed" 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 "flashed" 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 11.

**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.

**Notes:**

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 301.

### 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.
  - **PDS or PDSE conversion** — Conversion occurs when you specify the CONVERT keyword with these data sets.
  - **Block-by-block processing of direct access data sets** — Block-by-block processing occurs when you specify the REBLOCKADDRESS or the AUTOREBLOCKADDRESS 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 SET &STORGRP = 'fcstrgrp'
```

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 179.

**Designating FlashCopy usage**

The `FASTREPLICATION(REQUIRED | PREFERRED | NONE)` keyword tells DFSMSdss how you want FlashCopy to be used. The default is `FASTREPLICATION(PREFERRED)`.

`FASTREPLICATION(REQUIRED)` specifies that DFSMSdss must use FlashCopy to move data. If FlashCopy cannot be used, DFSMSdss issues error message ADR938E which indicates the processing of the current data set or that the entire COPY task failed. If the processing of the current data set failed, DFSMSdss does not try any other methods of data movement for the current data set and attempts to use FlashCopy for the subsequent data sets. If the entire copy task failed, DFSMSdss terminates the copy operation.

**Restriction:** You cannot use the `FASTREPLICATION(REQUIRED)` and `CONCURRENT` keywords together.

`FASTREPLICATION(PREFERRED)` specifies that you want DFSMSdss to use FlashCopy before any other method to move data (even when you specify the `CONCURRENT` keyword). If FlashCopy cannot be used and you have specified the `CONCURRENT` keyword, DFSMSdss attempts to use concurrent copy. If you have not specified the `CONCURRENT` keyword or if concurrent copy has failed, DFSMSdss uses traditional data movement methods to copy the data.

`FASTREPLICATION(NONE)` specifies that DFSMSdss not attempt to use FlashCopy to copy data.

For more information about the `FASTREPLICATION` keyword, refer to “FASTREPLICATION” on page 328.

**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 FCTOPPRCPrimary 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 PPRC 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 operation.

**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.

**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.

**Note:** If the target 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, refer to **z/OS DFSMSdfp Storage Administration**.

**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.
Notes:
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 DFSMSdfp 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 333, “FCNOCOPYTOCOPY” on page 334, and “FCWITHDRAW” on page 405 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 REBLOCKADDRESS OR the AUTORELOCKADDRESS keyword.
  - Utilities — SnapShot cannot be used if your data must be moved with a utility.

If the source data is in an RVA, DFSMSdss attempts to allocate the target data set on the same device type in the same RVA, thus increasing the probability that SnapShot can copy the data. If the source data set is multivolume and not contained entirely in one partition of one RVA subsystem, it is not possible to allocate the target so that SnapShot can be used. These data sets are allocated to whatever volumes are available, irrespective of their SnapShot capability.

**Designating SnapShot usage**

The FASTREPLICATION(REQUIRED | PREFERRED | NONE) keyword tells DFSMSdss how you want fast replication such as SnapShot to be used. The default is FASTREPLICATION(PREFERRED).

FASTREPLICATION(REQUIRED) specifies that DFSMSdss must use fast replication such as SnapShot to move data. If SnapShot cannot be used, DFSMSdss issues error message ADR938E which indicates that the processing of the current data set or the entire COPY task failed. If the processing of the current data set failed,
DFSMSdss does not try any other methods of data movement for the current data set. However, DFSMSdss attempts to use fast replication such as SnapShot for the subsequent data sets. If the entire copy task failed, DFSMSdss terminates the copy operation.

**Restriction:** You cannot use the FASTREPLICATION(REQUIRED) and CONCURRENT keywords together.

FASTREPLICATION(PREFERRED) specifies that DFSMSdss attempt to use SnapShot before any other method to move data (even when you specify the CONCURRENT keyword). If SnapShot cannot be used and you have specified the CONCURRENT keyword, DFSMSdss attempts to use virtual concurrent copy. If you do not specify the CONCURRENT keyword or if virtual concurrent copy fails, DFSMSdss uses traditional data movement methods to copy the data.

FASTREPLICATION(NONE) specifies that you do not want DFSMSdss to use SnapShot to copy data. Instead, DFSMSdss attempts to use virtual concurrent copy if the CONCURRENT keyword is specified. If virtual concurrent copy cannot be used, DFSMSdss uses traditional data movement methods to move the data.

For more information about the FASTREPLICATION keyword, see “FASTREPLICATION” on page 328.

**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 321.

### 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(UNDEFINEDDSORG) 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

**Note:** When you are copying or restoring multivolume data sets, be aware of the following considerations:

- DFSMSdss does not preserve candidate volumes. However, for SMS-managed data sets, if you copy and do not specify any output volumes, DFSMSdss preserves the source volume count. If you copy and do specify the output volumes, DFSMSdss sets the volume count to the number of output volumes specified.
- DFSMSdss does not ensure that the copied or restored data set is on the same number of volumes as the original data set, nor does DFSMSdss ensure that the copied or restored data set extents are the same as the original data set. Instead, DFSMSdss tries to allocate the new data set on as few volumes as possible. This may result in the copied or restored data set becoming a single-volume data set.
- In addition, DFSMSdss tries to allocate each volume so that all data is contained in a single primary allocation of contiguous space with few, if any, of the secondary allocations being used.
Converting VSAM and non-VSAM data sets to multivolume

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

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

Moving VSAM data sets

When you move a VSAM data set and the REPLACE or REPLACEUNCONDITIONAL keywords are not specified, you must specify DELETE, 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:

- DFSMSdss must be able to invoke IDCAMS to copy an extended-format VSAM data set.
- When performing a logical copy operation of an extended-format data set, the target data set allocation must be consistent with the source data set allocation as follows:
  - If the source is extended-format VSAM, then the target must be extended-format VSAM.
  - If the source is extended-addressable VSAM, then the target must be extended-addressable VSAM.
– If the source is a compressed-format VSAM KSDS, then the target must be a compressed-format VSAM KSDS.
– If the source is an alternate index for an extended-format KSDS, then the target must be an alternate index for an extended-format KSDS.
– If the source is an alternate index for a compressed-format KSDS, then the target must be an alternate index for a compressed-format KSDS.
– The target control interval size must be equal to the source.

• 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.

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 458):

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 459).

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 11 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>
<thead>
<tr>
<th>Target Environment</th>
<th>Source Environment</th>
<th>Source Status</th>
<th>DFSMSdss Function</th>
<th>TGTGDS Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMS</td>
<td>Non-SMS</td>
<td>Cataloged</td>
<td>COPY</td>
<td>DEFERRED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MOVE</td>
<td>ACTIVE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not Cataloged</td>
<td>COPY</td>
<td>DEFERRED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MOVE</td>
<td>DEFERRED</td>
</tr>
<tr>
<td>SMS</td>
<td>ACTIVE</td>
<td></td>
<td>COPY</td>
<td>DEFERRED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MOVE</td>
<td>ACTIVE</td>
</tr>
<tr>
<td></td>
<td>DEFERRED</td>
<td></td>
<td>COPY</td>
<td>DEFERRED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MOVE</td>
<td>DEFERRED</td>
</tr>
<tr>
<td></td>
<td>ROLLED-OFF</td>
<td></td>
<td>COPY</td>
<td>DEFERRED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MOVE</td>
<td>ROLLED-OFF</td>
</tr>
</tbody>
</table>

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

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 179 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 87 and “Changing management class with restore processing” on page 88).

Selecting target volumes

In an SMS-managed environment, you generally allow the system to place data sets for you. If for some reason you want to control the placement of the data sets (for example, because of performance problems or because you want to put data sets on some new, empty volumes you have just added to a storage group), you must take special steps.

If you use OUTDDNAME or OUTDYNAM to specify a volume list, the volume serial numbers are passed as input to the ACS routines. Depending on how your ACS routines are written, this input might or might not be used in determining where to place the data set.

One way to guarantee that data sets go to particular volumes is to write your storage group ACS routine such that data sets are moved to the volumes you select.
Alternatively, if a data set’s storage class has the guaranteed-space attribute, the data set is placed on the user-specified volumes if the volumes reside in the same storage group and ACS selects that storage group for the data set. By using BYPASSACS and STORCLAS keywords, you can ensure that the storage group selected contains the volumes you specify with OUTDDNAME or OUTDYNAM. However, for this procedure to work, your storage group ACS routine must use storage class to determine the storage group for a data set. This allows you to determine which storage class to specify with the STORCLAS keyword to ensure that the storage group containing the volumes specified with OUTDDNAME or OUTDYNAM is selected.

Changing storage class with Copy

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.

Changing management class with Copy

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.

---

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.

Rules for moving to preallocated target data sets

To use a preallocated data set, you must specify the REPLACE or REPLACEUNCONDITIONAL keyword. If the REPLACE keyword is specified, the preallocated data set name must be identical to the source data set name. If the RENAMEUNCONDITIONAL(newname) and REPLACEUNCONDITIONAL keywords are specified, the preallocated data set name must match the new name filter criteria. You cannot, however, copy a data set to a preallocated target data set with the same name within an SMS environment because SMS does not support duplicate data set names.

The rules for moving VSAM and non-VSAM data sets to preallocated data sets follow.

VSAM preallocation: An existing data set qualifies as a preallocated target for a data set copy operation if the cluster name matches and the complete cluster is available on target volumes.

The preallocated data set is usable if all of the following conditions that apply to the data set being processed are met:

- The user is authorized to update the target data set.
- The cluster types match.
• The number of components match.
• The key length and offset match.
• The KEYRANGES match.
• None of the components are multivolume.
• Sufficient space is available for each component.
• Key sequential data sets (KSDS) are reusable or empty.
• Key range data sets are empty.
• The data set is cataloged in the standard order of search, if required for the copy operation.
• The data set has no alternate indexes or paths defined over it (except for a single path defined directly over the base cluster).

If a target data set is preallocated, it is scratched and reallocated when it is being renamed and:
• Any of the following source and target data set attributes do not match:
  – CI size
  – Record length
  – IMBED (only KSDS and key range data sets)
  – Key length (only KSDS and key range data sets)
  – REPLICATE (only KSDS and key range data sets)
  – SPANNED
• The data set was not defined as reusable and the high-used relative byte address (RBA) of a target VSAM KSDS is not 0.
• The target data set is not large enough to contain the source data set.

**Non-VSAM preallocation:** An existing data set qualifies as a preallocated target for a data set copy operation if the data set names match, the complete data set is available on target volumes, and:
• For single-volume target qualification, the data set organization is partitioned or the data set’s volume sequence number in the VTOC is 1 and the last volume flag is on.
• For multivolume target or single-volume target with the last volume flag off, the data set is cataloged in the standard order of search. All volume serial numbers returned by a locate operation on the data set are in the output volume list. (Candidate volumes are acceptable.)

**Note:** If a target data set is preallocated, but is not large enough to contain the source data set, it will be scratched and reallocated if it is being renamed.

You may use data set COPY to upgrade your standard format sequential data sets to large format data sets. When copying a data set and a usable preallocated target is found, it will be used as the target of the copy operation. When copying a standard format sequential data set and a preallocated large format data set is found, it will be used. If the preallocated large format data set does not have enough space for the source data, it will be scratched and reallocated as a large format data set. When copying a large format data set and a standard format sequential data set is found, it will be used and upgraded to a large format data set. If the preallocated standard format sequential data set does not have a large enough allocation to hold the source data, it will be scratched and reallocated as a large format data set.
If a user wishes to downgrade a large format data set to a standard format sequential data set, allocate a standard format sequential data set and use a utility such as IEBCOPY to copy the data from the large format data set to the standard format sequential data set.

You may also use data set COPY to upgrade your data sets that are not enabled for CA reclaim to data sets that are enabled for CA reclaim. To do this, simply preallocate a data set that is enabled for CA reclaim to use as a target of the copy. In copying a data set that is not enabled for CA reclaim, if a preallocated data set that is enabled for CA reclaim is found, the preallocated data set that is enabled for CA reclaim is used for the target of the copy. If the preallocated data set that is enabled for CA reclaim is not large enough to hold the data being copied, it is scratched and reallocated as a data set that is enabled for CA reclaim. In copying a data set that is not enabled for CA reclaim, if a preallocated data set that is not enabled for CA reclaim is found, the data set that is not enabled for CA reclaim is used for the target of the copy. If the preallocated data set that is not enabled for CA reclaim is not large enough to hold the data being copied, it is scratched and reallocated as a data set that is not enabled for CA reclaim.

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 114). 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 179 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 INDDYNAM 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...
The 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 179.

### 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 311.

### 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 328.

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 321.

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 333, “FCNOCOPYTOCOPY” on page 334, and “FCWITHDRAW” on page 405.

Choosing space efficient FlashCopy with the FCSETGTOK keyword

A space efficient volume does not have all of its physical space allocated when it is created. Instead, its physical space is allocated on a track basis. When data is written to a space efficient volume, a track of physical space is taken from the segments assigned to a repository volume, and is used to hold the data for the space efficient volume. A repository volume can provide the physical space for multiple space efficient volumes.

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

You can use 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 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 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 33.

Attention: Space efficient FlashCopy 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 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. 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
- 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.

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 234.

For information about the FCWITHDRAW keyword, refer to “FCWITHDRAW” on page 405.
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.

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.

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)
/*

Notes:
1. The freeze and thaw operations require the specified devices support the FlashCopy Consistency Group function.
2. There is one Consistency Group timer per logical subsystem (LSS) for FlashCopy.
3. The Consistency Group timer has a default of 120 seconds. The timer value can be set via the ESS Web User Interface.
4. The 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 279 and "FCCGVERIFY" on page 280.

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 311.

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 328.

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 321.

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”** discusses the advantages and disadvantages of the two types of conversion.
- **“Conversion by data movement” on page 142** describes how to use the COPY and DUMP/RESTORE commands to convert data sets to SMS management.
- **“Conversion without data movement” on page 143** describes how to use the CONVERTV command to convert volumes to SMS management.
- **“Special data set requirements for conversion to SMS” on page 146** describes some of the data sets that have special requirements for conversion to SMS management.
- **“Converting from SMS management without data movement” on page 148** describes how to use the CONVERTV command to convert volumes from SMS management.
- **“Special data set requirements for conversion from SMS” on page 148** describes some of the data sets that have special requirements for conversion from SMS management.

### Evaluating conversion to SMS management

When you convert data to SMS management, the first thing to consider is whether to convert data sets with or without data movement. If you have SMS-managed volumes with sufficient free space, you can convert data sets by simply moving them from non-SMS-managed volumes to SMS-managed volumes. The same is also true if you are converting data from SMS-management. Converting data sets to SMS management by data movement is often preferable because it allows the system to place the data sets for you. This ensures that the data sets are placed on volumes in storage groups that can meet the availability and performance requirements of the data set.

If, however, you do not have sufficient free space on your SMS-managed volumes to convert by data movement, you might have to convert data sets without data movement. The drawback to this method of conversion is that it does not allow the system to place data sets for you. You must ensure that the storage group in which you place the volume can meet the availability and performance requirements of the data sets.

Regardless of how you convert to SMS management, you must determine the eligibility for conversion of your data sets and volumes prior to conversion.

### Data sets ineligible for conversion to SMS

The following data sets cannot be converted to SMS management:
- Absolute track allocation data sets
- Direct with OPTCD=A
- GDS with candidate volumes
- Indexed sequential data sets
- Model DSCBs
SYS1 storage index data sets (SYS1.STGINDEX)
Indirectly cataloged data sets
Uncataloged, multivolume data sets
VSAM data sets not cataloged in an integrated catalog facility catalog
VVDS/VTOCIX
Unmovable data sets

Notes:
1. Using the CONVERTV command with the SMS and TEST keywords identifies ineligible data sets without actually converting any data.
2. VVDS/VTOCIX data sets can be SMS-managed, but DFSMSdss cannot be used to convert them, except when using the CONVERTV command to convert the volume that they are on.

Data sets ineligible for conversion from SMS
The following data sets cannot be converted from SMS management:
- Extended-format sequential data sets
- Extended-format VSAM data sets
- Indirectly cataloged data sets
- VSAM data sets that have record level sharing (RLS) information associated with them
- VSAM base cluster or alternate index with a component with greater than 255 extents.

Notes:
1. Using the CONVERTV command with the NONSMS and TEST keywords identifies ineligible data sets without actually converting them.
2. VVDS/VTOCIX data sets can be non-SMS-managed, but DFSMSdss cannot be used to convert them, except when using the CONVERTV command to convert the volume that they are on.

Volumes eligible for conversion to SMS
A volume is eligible for conversion if it:
- Is a DASD volume
- Is permanently mounted and accessible online
- Has an indexed VTOC
- Is defined in an SMS storage group in an active configuration

Conversion by data movement
By using the logical data set COPY or DUMP/RESTORE command, you can move data sets between non-SMS-managed and SMS-managed volumes. When moving data sets to SMS-managed volumes, COPY and RESTORE commands invoke ACS to assign classes to the data sets. This type of conversion to SMS allows the data sets to be placed on the most appropriate SMS-managed volume.

Converting to SMS management by data movement
When moving data sets to SMS-managed volumes, you can use the COPY or RESTORE command. You can specify storage and management class names with the STORCLAS and MGMTCLAS keywords. You can also specify output volumes with OUTDDNAME and OUTDYNAM. DFSMSdss passes the class names and volume serial numbers to ACS, which might use them in determining the classes and placement of the data set.
This method of converting data sets to SMS management is similar to moving data sets in an SMS-managed environment as described in "Moving SMS-managed data sets" on page 123.

If you use the COPY or RESTORE command on a data set that is ineligible for SMS management and if a non-SMS-managed volume has been specified in the output volume list, DFSMSdss puts it on a non-SMS-managed volume. However, if you specify STORCLAS and BYPASSACS with the COPY or RESTORE command for a data set that is ineligible for SMS management, the copy or restore operation fails.

For data sets cataloged outside the standard order of search, use the INCAT keyword on the COPY or DUMP command to identify what catalog to search. Use the SELECTMULTI keyword on the COPY or DUMP command to convert multivolume data sets. This allows you to specify only the volume with the primary component on the LOGINDD or LOGINDY parameter. You can use the SPHERE keyword on the COPY DUMP/RESTORE command to convert entire VSAM spheres (if you use SPHERE on the RESTORE command, you must specify it on the corresponding dump as well).

Conversion from SMS management by data movement

To take a data set out of SMS management with the COPY or DUMP/RESTORE command, you should specify the BYPASSACS and NULLSTORCLAS keywords. This forces DFSMSdss to make the data set non-SMS-managed.

Conversion without data movement

Conversion without data movement is divided into two phases: conversion of data sets and conversion of volumes. Convert data sets and the volumes they reside on without moving data by using the DFSMSdss CONVERTV command. You should set up RACF FACILITY class authorization to limit the people who can use the CONVERTV command. When you use the CONVERTV command to perform conversion, it attempts to convert all the data sets on the volume. After all the data sets are processed, the volume is placed in one of the following three states:

- CONVERTED—the volume and its data sets are converted to SMS management. A volume can be placed in this state with the CONVERTV command and the SMS keyword.
- INITIAL—new allocations cannot be made to the volume and, although users can access their data sets, the data sets cannot be extended to other volumes. A volume may be placed in this state because you have used the CONVERTV command with the PREPARE keyword to reduce activity to the volume prior to conversion. A volume may also be placed in this state if you are attempting to convert it but it contains data sets that are not eligible for conversion.
- NONSMS—the volume and its data sets were taken out of the CONVERTED or the INITIAL state and are non-SMS-managed. A volume can be placed in this state with the CONVERTV command and the NONSMS keyword.

Simulating conversion

Before you convert a volume to SMS management, you should simulate the conversion to ensure that all the data sets on the volume are eligible for conversion to SMS. In addition, simulating conversion shows you the classes ACS would assign to the data sets eligible for conversion.
You can simulate conversion by using the CONVERTV command with the SMS and TEST keywords. If the volume is ineligible for conversion, the data sets on the volume are still examined to determine their eligibility for conversion (provided the volume is permanently mounted and online).

When you use CONVERTV SMS TEST, you are in the ACS CONVERT environment. Only the storage class and management class ACS routines are executed. For a list of variables available to ACS routines during CONVERTV processing, see “ACS variables available during RESTORE and CONVERTV processing” on page 181.

Simulated conversion creates a report that identifies data sets ineligible for conversion. For a sample of this report, see “SMS report” on page 146. 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 181.

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.
Figure 3 shows a sample report generated by DFSMSdss during CONVERTV SMS processing.

**Figure 3. SMS Report**

### Special data set requirements for conversion to SMS

Some data sets have special requirements for conversion to SMS management. The sections below describe the special considerations for converting these data sets to SMS management.

#### VSAM sphere eligibility

A VSAM sphere is considered to be a single data set by the CONVERTV command. As a result, either all the data sets of the sphere are converted or none of them are.

If any of the following parts are ineligible for conversion, then all the clusters that compose the sphere are ineligible for conversion:

- Components of a base cluster
- Alternate indexes related to the base cluster
- Alternate index components
- Paths relating alternate indexes to the base cluster

You must direct all parts of a VSAM sphere (the base cluster, base cluster components, alternate indexes, alternate index components, and paths) to the same catalog by using an alias. If they are not directed to the same catalog, the sphere cannot be converted to SMS management. To correct this problem you can either rename the data sets in the sphere, or add or delete catalog aliases, and rerun the CONVERTV command.

#### Multivolume data sets

If you do not specify SELECTMULTI, all volumes must be included in DDNAME or DYNAM volume lists.
If you specify input volumes (with either the DDNAME or DYNAM volume list), a data set is selected based on the following criteria:

- When you either specify SELECTMULTI(ALL) or specify input volumes without specifying the SELECTMULTI keyword, all of the volumes that contain a part of the non-VSAM data set or VSAM base cluster must be in the volume list.
- When you specify SELECTMULTI(ANY), any part of the non-VSAM data set or VSAM base cluster can be on a volume in the volume list.
- When you specify SELECTMULTI(FIRST), the volume list must include the volume that contains either the first part of the non-VSAM data set or the primary data component of the base cluster for a VSAM sphere.

Multivolume data sets are not eligible for conversion if any part of the data set resides on volumes that:

- Do not have indexed VTOCs
- Are not defined in an SMS storage group
- Are defined to a different storage group
- Are not permanently mounted and online

If the previous requirements are satisfied, DFSMSdss verifies that all the volumes on which the data set resides:

- Are permanently mounted and online
- Have indexed VTOCs
- Are defined to the same storage group

If all these criteria are met, the data set is converted to SMS management.

**Notes:**

1. If SELECTMULTI(FIRST) or SELECTMULTI(ANY) is specified, volumes not specified in the DDNAME or DYNAM volume lists are put in the INITIAL state following a successful conversion of the data set to SMS (unless the volume is already in the INITIAL or SMS state.)
2. If SELECTMULTI is not specified or if SELECTMULTI(ALL) is specified, volumes not specified in the DDNAME or DYNAM volume lists are not put in the INITIAL state.

DFSMSdss cannot determine whether or not a volume being converted is a candidate volume for one or more data sets in the system. If such a volume is converted, DFSMSdss cannot ensure consistent conversion for all of the volumes of the data set (or sets) for which the volume is a candidate. This can result in a data set having both SMS-managed and non-SMS-managed volumes in its volume list, which can cause the data set to become unusable.

To avoid this situation when performing CONVERTV operations, if you specify any volume of a multivolume data set in the list of volumes to be converted, ensure that you also include at least one of the primary volumes of the data set. This allows DFSMSdss to ensure that all of the volumes of the data set are converted consistently.

**GDG data sets**

Generation data groups (GDGs) require special consideration while being cataloged or uncataloged during SMS conversion. Uncataloged GDGs are converted to SMS management, but are left uncataloged. Messages ADR877I and ADR879I indicate NOT CATALOGED for the catalog name in the data set name lists for SMS processing.
Temporary data sets

Data set VTOC entries of temporary data sets are updated to indicate uncataloged SMS status.

VTOC and VVDS

Data set VTOC entries for the VTOC, VTOC index, and VVDS are updated to SMS management.

Converting from SMS management without data movement

If you want to take volumes out of SMS management, you can use the CONVERTV command with the NONSMS keyword. All volumes and most data sets are eligible for NONSMS processing. After you execute this command, the volume indicators that designate the volume as an SMS-managed volume are turned off. The active SMS configuration should be updated to remove the volume from its storage group, otherwise data set allocations to the volume will fail. Thereafter, only non-SMS-managed data sets can be allocated to the volume.

As with the SMS keyword, you can specify the TEST keyword with NONSMS. No conversion is actually done, but a report is generated that identifies the data sets that are and are not eligible for conversion from SMS management. The report also indicates whether the volume as a whole is eligible for conversion from SMS management.

To convert a data set from SMS management, the data set’s classes are deleted from its catalog entry. Nonspecific volumes also are deleted from the catalog entry. For a VSAM data set, the SMS-related items are deleted from the catalog entry. For a non-VSAM data set, the catalog entries are updated to remove the SMS information. After the catalog and VVDS updates and deletions are made, the VTOC entry is updated to be non-SMS-managed.

Note: You cannot specify the CATALOG and REDETERMINE keywords with NONSMS.

Special data set requirements for conversion from SMS

When being converted from SMS management, some data sets require special consideration. The following sections discuss some of the special requirements for converting data sets from SMS management.

Multivolume data sets

All pieces of a multivolume data set must be converted from SMS management at the same time. You can do this by using the SELECTMULTI keyword.

If you do not specify SELECTMULTI, then you must specify all the volumes in the DDNAME or DYNAM volume list on which the data set resides.

If you specify input volumes (with either the DDNAME or DYNAM volume list) for
NONSMS processing, a data set is selected based on the following criteria:

- When you either specify SELECTMULTI(ALL) or specify input volumes without specifying the SELECTMULTI keyword, all of the volumes that contain a part of the non-VSAM data set or VSAM base cluster must be in the volume list.
- When you specify SELECTMULTI(ANY), any part of the non-VSAM data set or VSAM base cluster can be on a volume in the volume list.
- When you specify SELECTMULTI(FIRST), the volume list must include the volume that contains the first part of either the non-VSAM data set or the primary data component of the base cluster for a VSAM sphere.

Those volumes not included in the volume list will be placed in the INITIAL state. Being in the INITIAL state locks all allocations to the volume until all data sets residing on it are converted.

DFSMSdss cannot determine whether or not a volume being converted is a candidate volume for one or more data sets in the system. If such a volume is converted, DFSMSdss cannot ensure that all of the volumes of the data set (or sets) for which the volume is a candidate, are converted consistently. This can result in a data set having both SMS-managed and non-SMS-managed volumes in its volume list, which can cause the data set to become unusable.

To avoid this situation when performing CONVERTV operations, if you specify any volume of a multivolume data set in the list of volumes to be converted, ensure that you also include at least one of the primary volumes of the data set. This allows DFSMSdss to ensure that all of the volumes of the data set are converted consistently.

**GDG data sets**

When you convert from SMS management, generation data group (GDG) data sets require special consideration with regard to cataloging. Data sets marked as “deferred roll in and rolled out” are uncataloged.

**Temporary data sets**

Data set VTOC entries for temporary data sets are updated to non-SMS status.

**VTOC and VVDS**

Data set VTOC entries for the VTOC, VTOC index, and VVDS are updated to non-SMS status.

**Special considerations for using non-SMS-managed targets**

When moving to non-SMS-managed targets, there are some special considerations for certain data sets:

- Extended-format data sets cannot be moved to a non-SMS-managed target.
- COPY with DELETE and without 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, “DFSMShsm filtering—choosing the data sets you want processed,” on page 263. DFSMSdss selects only data sets that have space that can be released. You can also use ISMF to build a list of data sets based on the amount of unused space and to invoke DFSMSdss to release the unused space in them.

Exclude data sets whose last block pointer in the data set VTOC entry is not maintained in the VTOC by using the EXCLUDE keyword. This can occur if you use an access method other than BSAM, QSAM, BPAM, or VSAM. DFSMSdss does not release space for data sets whose last block pointer in the data set entry is 0.

The following options can help you use the release function more effectively:

**MINSECQTY(n)**

Allows you to specify that space not be released unless the user’s secondary allocation is greater than or equal to \( n \). In this way, you ensure that the user can still add to the data set after the release.

The default value for \( n \) is 1.

**MINTRACKSUNUSED(n)**

Allows you to specify that space not be released unless the number of unused tracks is greater than or equal to \( n \). Without MINTRACKSUNUSED, space is released if the data set has one or more unused tracks.

Note: When space in a data set is released, all unused space is released, not just the amount beyond the minimum unused (as specified by MINTRACKSUNUSED).

To protect the user, DFSMSdss does not release any space in a data set if:

- The data set has the maximum number of used extents. A data set with the maximum number of allocated extents but fewer than the maximum number of used extents will have the unused space released.
- The cylinder-allocated data set has unused tracks but not an entire unused cylinder.
• The data set’s name begins with SYS1, unless the PROCESS(SYS1) keyword is specified. To limit the use of PROCESS, you need to set up a RACF FACILITY class profile.

**Compressing a PDS**

The COMPRESS command compresses a PDS on a specified volume. Compression removes unused space between members in a partitioned data set. This recovered space is then available for reuse at the end of the data set. Depending on the filtering criteria you specify, you can compress all the partitioned data sets or only some of the data sets. This command is useful for compressing system partitioned data sets before applying maintenance (thus avoiding certain space-related abends). You must not compress the data sets that contain DFSMSdss or IEBCOPY executable code.

The actual PDS compression is done in place. To prevent loss of data if the system or the compression operation abnormally ends during processing, back up your volume or data sets that meet the filtering criteria before using this command.

COMPRESS does not support processing partitioned data sets that:
- Are unmovable
- Have no directory

**Deleting unwanted data sets**

You can use the DELETE and PURGE keywords and data set filtering with a physical data set dump to delete unwanted data sets from DASD.

**Note:** This does not apply to VSAM data sets, multivolume non-VSAM data sets, or migrated data sets.

On a logical data set dump when using the DELETE keyword, VSAM, non-VSAM, and multivolume data sets are deleted. DFSMSdss cannot be used to delete migrated data sets.

The following steps show how to delete (scratch and uncatalog) all data sets that have expired and all data sets that have not been referred to in the last year. The data sets are not actually moved to a dump volume.

1. **JCL requirement:**
   ```
   //NOTAPE DD DUMMY
   ```

   The above JCL prevents moving any data sets.

2. Issue the following control statements to delete (scratch and uncatalog) all data sets not referred to in the last year:
   ```
   DUMP INDD(VOL111) OUTDD(NOTAPE) -
   DATASET(BY(REFDT,LE,*,-366)) -
   DELETE PURGE
   ```

3. Issue the following control statements to delete all expired data sets:
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 INDD(VOL111) OUTDD(NOTAPE) -
DATASET(INCLUDE(/**) -
   BY(EXPDT,LT,*) -
   DELETE
```

**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)) -
   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. 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
\textit{z/OS DFSMSdfp Storage Administration}.

Notes:
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:

   \begin{verbatim}
   DUMP INDD(DASD1) OUTDD(TAPE1) OPTIMIZE(3) -
   DATASET(BY((DSORG,NE,VSAM),(ALLOC,EQ,MOV),(MULTI,EQ,NO)) -
   DELETE PURGE
   \end{verbatim}

2. Enter this control statement to defragment the volume:
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 FCTOPPRCPrimary 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 FCTOPPRCPrimary 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](https://www.ibm.com)  

For more information about the FCTOPPRCPrimary keyword, refer to [z/OS DFSMSdfp Storage Administration](https://www.ibm.com)

**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.
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 meet all the criteria for SnapShot use. Use the DEBUG(FRMSG(MINIMAL | SUMMARIZED | DETAILED)) keyword to help you resolve this situation. Include this keyword to indicate the applicable fast replication message level (MIN, SUM, or DTL) in your DEFRAG command. The message level controls the type and amount of information that DFSMSdss provides.

DEBUG(FRMSG(MIN | SUM | DTL)) directs DFSMSdss to issue an informational message that indicates why SnapShot was not used. When you specify FASTREPLICATION(REQUIRED), the informational message is issued in addition to the ADR938E message whether you have specified the DEBUG(FRMSG(MIN | SUM | DTL)) keyword or not.

For more information about the DEBUG(FRMSG(MIN | SUM | DTL)) keyword, refer to z/OS DFSMSdfp 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 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 has been defined as a retained DLF object for use with Hiperbatch.

Note: Exclude system data sets that are opened and are being accessed without an enqueue.

**DEFRAG options**

You can use the following keywords to make more efficient use of the DEFRAG command:

- **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.
**MMOVPC(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.

**PASDelay**  
Specifies the time delay between the passes (\( p \)) specified in MAXMOVE\( (n,p) \) or MMOVPC\( (n,p) \) 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 MMOVPC 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 free space extent, perform only the first pass. Do so by coding the MAXMOVE\( (n) \) parameter using a very high value for \( n \), or code the MMOVPC\( (n) \) 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) MMOVPC(75)
```

- Experimenting with the DEFRAG FRAGI and MAXMOVE or MMOVPC 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 provided by message ADR213I. Therefore, the information in message ADR213I may not reflect the state of the volume at the completion of DFSMSdss processing because another job may allocate or delete data sets on the processed volume between the time the serialization is released and the ending statistics are obtained. The serialization scheme is described in [z/OS DFSMSdfp Storage Administration](#).

The DEFrag command does a RESERVE on the VTOC to serialize access to the VTOC. The DEFrag command also serializes access to each data set before relocating the extent of a data set. The enqueue scheme used by the DEFrag function ensures integrity on a single processor but does not ensure integrity for data sets on DASD shared between processors. This is due to the use of an ENQ scope of SYSTEM for the SYSDSN resource name. To ensure the integrity of data
sets on a shared DASD, you must do one of the following:

- Vary the volume offline from all processors except the one on which DEFRAG runs. After the DEFRAG function finishes, you can vary the volume back online for the other processors.

- In either a JES2 or a JES3 environment, you can use multisystem GRS (or equivalent function) to convert the scope of all enqueues with a resource name of SYSDSN from SYSTEM to SYSTEMS by placing SYSDSN in the GRS SYSTEM INCLUSION resource name list (RNL). This allows all systems in the GRS ring to be made aware of all SYSDSN enqueues. The default GRS System Inclusion RNL includes SYSDSN. However, ensure that this has not been changed on your system before using the DEFRAG command on a volume shared between two or more processors.

**Note:** GRS must not be used to convert the scope of any of DEFRAG function’s SYSTEMS enqueues (including SYSVSAM) to SYSTEM by placing the resource names in the GRS RNL. GRS may, however, be used to convert DEFRAG function’s RESERVE on SYSVTOC to a simple enqueue with a scope of SYSTEMS by including it in the GRS “RESERVE CONVERSION RNL”. If you choose not to do this, you can avoid doing two global serializations on the volume’s VTOC by placing SYSVTOC in the GRS Systems Exclusion RNL, thus changing RESERVE’s global enqueue to a local enqueue. For the restrictions that apply to enqueues and dequeues, see [z/OS MVS Planning: Global Resource Serialization](https://www.ibm.com). The DYNALLOC serialization mechanism of DFSMSdss does not solve all cross-system serialization problems. GRS (Global Resource Serialization) is recommended with shared DASD.

- If you are running on a system using JES2 and are not using multisystem GRS (or equivalent function), you can use DEFRAG function’s BY filtering to specifically include or exclude data sets from processing. Both creation date and last-referenced date criteria are needed to ensure that only those data sets that are not in use are selected for DEFRAG function processing. For example, if you choose to defragment a volume with typical TSO or batch data sets, you could select only those data sets that were created and referenced more than two days previously. Two days should be a minimum selection age because of the level of precision of the creation date. In the following example, data set A.B.C is created two minutes before the DEFRAG function begins.

<table>
<thead>
<tr>
<th>TIME OF DAY</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2359</td>
<td>Create data set A.B.C</td>
</tr>
<tr>
<td></td>
<td>Begin DEFRAG BY(CREDT,LT,-1). Data set A.B.C is selected because it has now been one day since creation</td>
</tr>
</tbody>
</table>

However, because there is a date change between the two actions, A.B.C was selected for the DEFRAG operation. A two day delay enforces a convention that more than 24 hours must pass before a data set is eligible for DEFRAG. In an environment with TSO and batch data sets, the probability that one of these data sets will be open more than 24 hours is low. In the following example, choosing data sets that have not been manipulated within the preceding 24 hours causes DEFRAG processing only for those data sets on volume SHARE3 that were created and referenced more than two days previously and that are not temporary data sets.
The two-day criterion is probably sufficient for TSO and batch type data sets. However, if there are applications that use the volumes being defragmented, consider setting the delay time to the maximum time that the application would have a data set open.

- If you are running on a system using JES3 with MDS enabled and are not using multisystem GRS (or equivalent function), you can use the DEFRAG command DYNALLOC keyword to provide serialization for data sets on shared DASD.

Note: Not all data sets allocated within a JES3 environment are known to the global. The use of the DYNALLOC keyword does not provide cross-system serialization for these data sets.

- Allocation of existing (old) data sets whose names appear in the RESDSN and DYNALDSN lists are not protected by the DYNALLOC serialization mechanism of DFSMSdss. DEFRAG processing for these data sets can be prevented by placing their names (or filters for the names) in the EXCLUDE LIST for the DEFRAG command.

- New data sets created with nonspecific allocation (no volume serial supplied) are not protected by the DYNALLOC serialization mechanism of DFSMSdss. However, you can use BY filtering with the DEFRAG command to specifically include or exclude data sets from processing. In the following example, the DEFRAG function processes only those data sets on volume SHARE3 that were created more than two days before.

```
DEFRAG BY(LIST((CREDT LT *, -2), (REFDT LT *, -2))) -
   EXCLUDE(LIST(SYS8*, T*, **)) DYNALLOC DYNAM(SHARE3)
```

You can also use the EXCLUDE parameter to avoid processing data sets that were created by long-running programs or subsystems more than two days previously but that are still allocated. In the following example, if the newly created data sets are temporary, the DEFRAG operation processes only those data sets on volume SHARE3 that were created more than two days before and are not temporary data sets.

```
DEFRAG BY(LIST(CREDIT, LT, *, -2)) 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.xxxxxxxxx.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, DADS functions might fail with message IEC602. To correct this problem.
or to delete the “SYS1.DFDSS.DEFRAG.xxxxxxx.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 VSAM data set has the ERASE attribute
  - The data set is password protected. (In this case, if the data set is also defined to RACF, the RACF ERASE option is taken. Table 12 provides more detail.)

**Table 12. Data Set Erase Table for DEFRAG with z/OS Security Server (RACF element).**

<table>
<thead>
<tr>
<th>Password Protected</th>
<th>Defined ERASE</th>
<th>RACF Protected</th>
<th>Erased on Scratch</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes (=ERASE)</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>Yes (=NOERASE)</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes (=ERASE)</td>
<td>Yes</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes (=NOERASE)</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes (=ERASE)</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes (=NOERASE)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**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](https://publib.boulder.ibm.com/infocenter/zos/v1r13/topic/zd16dfsms.pdf).
Maximizing track utilization by reblocking data sets

DFSMSdss provides a REBLOCK keyword that allows users to maximize the track usage by data sets during copy and restore processing. When REBLOCK is specified on a fully or partially qualified name of a sequential or partitioned data set during copy or restore processing, DFSMSdss will choose an optimal block size for the data set and the device. However, the installation reblock exit can specify that a different block size be used (except for partitioned load modules during copy operations).

REBLOCK is ignored for:
- Unmovable data sets
- Data sets with record format of U (except for partitioned load modules during copy operations), V, VB, VBS, or F
- Partitioned data sets with note lists (except for partitioned load modules during copy operations)
- Partitioned data sets that are also specified in the NOPACKING keyword

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

The reblockable indicator in the data set’s VTOC entry also determines whether a data set is to be reblocked or not. When the indicator is on, the data set is always reblocked to a system determined, optimal block size, except when the data set:
- Is a partitioned data set that is also specified in the NOPACKING keyword
- Is an unmovable data set
- Has a record format of V, VS, VBS, or F

The installation reblock exit is not called if the reblockable indicator is ON.

A PDSE being converted to a PDS will always be reblocked except when the data set has a record format of V, VS, VBS, or F; or when the data set’s record length is ‘0’.

For more information about the installation reblock exit routine, refer to z/OS DFSMSdfp Storage Administration.
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”
- “Using keywords to identify the problem” on page 168
- “Using the IBM Support Center” on page 176

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
If the failure occurred while you were using a function that does not appear in the list, it is possible the failure occurred in either DFSMSdfp or in the ISMF interface to DFSMShsm.

ISMF also uses the functions provided by the DFSMSdfp common services component, which consists of the following three routines:
- Common filter services
- DASD calculation services
- Device information services.

To begin the diagnostic procedure for a DFSMSdss failure, refer to "Using keywords to identify the problem."

For information about the DFSMSdss dump data set, refer to Chapter 13, “Format of the DFSMSdss dump data set,” on page 197.

For information about the DFSMSdss patch area, refer to Chapter 21, “Data integrity—serialization,” on page 559.

For information about diagnosing DFSMSdfp function failures, refer to z/OS DFSMSdfp 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 the z/OS DFSMSdfp Diagnosis.

Example: 5695DF175

Procedure: The component identification number for DFSMSdss is 5695DF175; see "Release-level keyword."

Release-level keyword
Using this keyword to identify the release level of DFSMSdss is optional in the SSF or EWS search argument. However, it is required in an APAR.

Example: RA10

Procedure: The keyword for z/OS V1R10 DFSMSdss is RA10; see "Type-of-failure and function keywords."

Type-of-failure and function keywords
Select one keyword from Table 13 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 13. Summary of Type-of-Failure Keywords

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Type of Failure</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABENDxxx</td>
<td>DFSMSdss ends abnormally because of a system-detected error.</td>
<td>169</td>
</tr>
<tr>
<td></td>
<td>Note: The ABENDxxx keyword does not apply to the DFSMSdss stand-alone restore program.</td>
<td></td>
</tr>
<tr>
<td>MSGADRnn</td>
<td>An error is related to a DFSMSdss message.</td>
<td>170</td>
</tr>
<tr>
<td>WAIT</td>
<td>The program does not seem to be doing anything.</td>
<td>171</td>
</tr>
<tr>
<td>LOOP</td>
<td>The program is doing something repetitively.</td>
<td>171</td>
</tr>
<tr>
<td>INCORROUT</td>
<td>Output from the program is incorrect or missing.</td>
<td>172</td>
</tr>
<tr>
<td>DOC</td>
<td>Documentation of the program is in error.</td>
<td>173</td>
</tr>
<tr>
<td>PERFM</td>
<td>Program performance is degraded.</td>
<td>173</td>
</tr>
</tbody>
</table>

ABENDxxx
Use this procedure when DFSMSdss ends abnormally. However, if the abend is user abend “0001,” skip this topic, and go directly to “MSGADRnn” on page 170.

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 174.

a. Record the relative task identifier. The number in parentheses (ttt) that immediately follows the message identifier is the relative task identifier. For example:

ADR013I (ttt)-mmmmm(yy), date_and_time TASK ABENDED . . .

b. Check prior messages for an ADR101I message with a task identifier that matches the one that you found in step 2a. For example:

ADR101I (ttt)-mmmmm(yy), TASKID xxx HAS BEEN ASSIGNED TO COMMAND 'command'

The command that is identified in this message is the function that is executing at the time of the abend. This command name is the function keyword.

You have finished when you have identified the function keyword for the ABENDxxx type-of-failure keyword. Refer to “Module keyword” on page 174 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 nnnn ADR nnnnnn OC yy

**Procedure:** To determine the keywords for type-of-failure, module, and occurrence code, follow these instructions:

1. Replace the nnnn in MSGADRnnnt with the message number and severity code. For example, if the message is ADR503A, the MSGADRnnnt type-of-failure keyword is MSGADR503A.

2. Replace the nnnnnn in ADRnnnnnn with the module identifier from the message (nnnnnnn in the following example).

ADR013I (ttt)-mmmmm(yy), date_and_time TASK ABENDED . . .

The number in the parentheses (ttt) immediately following the message identifier is the relative task identifier. The resulting module keyword is ADRnnnnnnn.

**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 176.

3. Replace the yy in OCyy 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 MSGADRnnnt type-of-failure keyword, the module keyword, and the occurrence code for the MSGADRnnnt type-of-failure. To identify the maintenance level of the module, refer to "Maintenance-level keyword" on page 176.

**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, SYSMDUMP, or SYSMDUMP data definition (DD) statement.
3. Follow these steps to find the function keyword:
   a. Record the relative task identifier. The relative task identifier is the number in parentheses (ttt) that immediately follows the message identifier in message ADR006I. Message ADR006I is printed when a DFSMSdss task begins. For example:
      
      ADR006I (ttt)-mmmmm(yy), date_and_time EXECUTION BEGINS
   b. Locate the tasks that are active (message ADR006I without a matching ADR013I). Message ADR013I is printed when a function ends. For example:
      
      ADR013I (ttt)-mmmmm(yy), date_and_time TASK COMPLETED
   c. Use CNTRL as the function keyword if no tasks are active, or if more than one function is active. See "Module keyword" on page 174.
   d. Check prior messages for an ADR101I message with a task identifier that matches the one that you found in step 3a. For example:
      
      ADR101I (ttt)-mmmmm(yy), TASKID xxx HAS BEEN ASSIGNED TO COMMAND 'command'

      The command identified in this message is the function that is executing at the time that the wait started. This function name is the function keyword.

You have finished when you have identified the function keyword for the WAIT type-of-failure. To identify the maintenance level of the module that failed, refer to "Module keyword" on page 174.

**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 176.

**LOOP**

Use this procedure when some part of the program repeats endlessly. If a message repeats endlessly, use the MSGADRnnnt 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)-mmmmm(yy), date_and_time EXECUTION BEGINS
   b. Locate the tasks that are active (message ADR006I without a matching ADR013I). Message ADR013I is printed when a function ends. For example: ADR013I (ttt)-mmmmm(yy), date_and_time TASK COMPLETED
   c. Use CNTRL as the function keyword if no tasks are active; see "Module keyword" on page 174.
   d. Analyze the program further to determine which task has failed if more than one function is active. You can best accomplish this by examining each function task in the dump.
   e. Check prior messages for an ADR101I message with a task identifier that matches the one that you found in step 3a. For example: ADR101I (ttt)-mmmmm(yy), TASKID xxx HAS BEEN ASSIGNED TO COMMAND 'command'

The command that is identified in this message is the function that is executing at the time the loop began. This function name is the function keyword.

You have finished when you have identified the function keyword for the LOOP type-of-failure. To identify the maintenance level of the module that failed, refer to "Module keyword" on page 174.

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 176.

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 nnnn ADR mmmmm 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 through 3 in "MSGADRnnnt" on page 170, 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 176.

**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 xxnnnnnnnnn

**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 176.

**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 176.

Module keyword

This topic applies only to the ABEND:xxx, WAIT, and LOOP type-of-failure keywords.

DFSMsdss uses subtasking to isolate functions; thus, a dump might contain a task and subtasks as follows:
- A DFSMsdss scheduler task
- A subtask for the reader/interpreter
- Subtasks for the DFSMsdss functions (for example, COPY or DUMP)
- A subtask for managing SVC services
- A subtask for managing IGWFAMS
- A subtask for attached utilities.

Each task has its own task control block (TCB) and request block (RB) chain in the dump. An explanation of the subtasks follows:
- The task for the scheduler is ADRDSSU, as indicated in the program request block/contents directory entry (PRB/CDE). Its normal state is waiting for the return from the EVENTS SVC (X'7D') in ADRDSSU.
- The subtask for the reader/interpreter is ADRRI01, as indicated in the PRB/CDE. Its normal state is waiting for the return from the WAIT SVC (X'01') in ADRRI01.
- The subtasks for the DFSMsdss functions are ADRBLDSA, ADRCGCR, ADRCPYD, ADRPRNT, ADREFRAG, ADRDTPF, ADRDTDS, ADRDTSRC, ADRDDDS, ADRCMPR0, ADRRLSE0, ADRTDPF, ADRTDDS, or ADRKVOL, as indicated in the PRB/CDE.
- The subtask that manages the SVC services is ADRSVCD. Its normal state is waiting for return from the WAIT SVC (X'01') in ADRSVCD.
The subtask that manages IGWFAMS is ADRATFMS. Its normal state is waiting for the return from the WAIT SVC (X'01') in ADRATFMS.

The subtask that manages other attached utilities is ADRMUTIL. Its normal state is waiting for the return from the WAIT SVC (X'01') in ADRMUTIL.

Procedure: 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/yyHDZIA10 aparnum
   
   .
   
   .
   
   ptfnum,yr.day,hh:mm:ss

   where:
   
   ADRxxxxx Module name
   mm/dd/yy Maintenance date
   HDZIA10 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 176. Otherwise, see Maintenance-level keyword
**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 in “Module keyword” on page 174).

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.”

**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.

The following services are available to you under one of these contracts:

**SRCHSERVICE**
Online database of APAR and PTF information, with extensive search capability.

**PSP**
Preventive Service Planning information database. This database contains the latest information concerning the installation of IBM products, including the latest service recommendations.

**SRD**
Service Request and Delivery Facility. This facility provides a means for election ordering and delivery of corrective services, including PTFs and APARS.

**ASAP**
Automatic Software Alert Process. This facility alerts the user when critical service information becomes available on a list of products that are selected by the user.

**ETR**
Electronic Technical Response. The user may electronically report problems and ask appropriate technical questions about IBM products through this facility. The ETR answers questions and problem reports by utilizing electronic responses. Optionally, you can request to talk to an IBM representative about your problem report. Submit nondefect-related, nontechnical questions to the question and answer (Q&A) queue in Canada on a severity 3, priority 3 basis.

**AST**
Automatic Status Tracking. This facility allows the user to request notification when the status of a user-selected APAR or PTF changes, or both.

**VPL**
View Program Listings. This is an online database of module listings for non-object code only (OCO) modules distributed through PTFs.

Info/System

Info/System is an interactive online database information retrieval program product. It is available primarily for use by customers with the companion 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 22, “Application programming interface,” on page 575.

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 14, are passed to the ACS routines.

Table 14. 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.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;ACCT_JOB</td>
<td>Accounting information from the JOB statement.</td>
</tr>
<tr>
<td>&amp;ACCT_STEP</td>
<td>Accounting information on the EXEC statement.</td>
</tr>
<tr>
<td>&amp;ACSENVIR</td>
<td>Environment in which ACS was invoked. Set to ’ALLOC.’</td>
</tr>
<tr>
<td>&amp;ALLVOL</td>
<td>Output volume serial numbers. References the same volume list as &amp;ANYVOL, but when used in a comparison, returns true only if all volume serial numbers satisfy the condition. &amp;ALLVOL is not available to the storage group ACS routine when VOLCOUNT(ANY) is specified.</td>
</tr>
<tr>
<td>&amp;ANYVOL</td>
<td>Output volume serial numbers. References the same volume list as &amp;ALLVOL, but when used in a comparison, returns true if any volume serial numbers satisfy the condition. &amp;ANYVOL is not available to the storage group ACS routine when VOLCOUNT(ANY) is specified.</td>
</tr>
<tr>
<td>&amp;APPLIC</td>
<td>Application identifier associated with the data set (available only if the RACF component of z/OS Security Server is installed).</td>
</tr>
<tr>
<td>&amp;DD</td>
<td>DDNAME</td>
</tr>
<tr>
<td>&amp;DEF_DATACLAS</td>
<td>Default data class name (available only if the RACF component of z/OS Security Server is installed).</td>
</tr>
<tr>
<td>&amp;DEF_MGMTCLAS</td>
<td>Default management class name (available only if the RACF component of z/OS Security Server is installed).</td>
</tr>
<tr>
<td>&amp;DEF_STORCLAS</td>
<td>Default storage class name (available only if the RACF component of z/OS Security Server is installed).</td>
</tr>
</tbody>
</table>
Table 14. Variables Passed to ACS Routines during DFSMSdss Copy Function (continued). The following variables are not available to the storage group ACS routine: &ACCT_JOB, &ACCT_STEP, &DD, &JOB, &PGM, and &XMODE.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DSN</td>
<td>Data set name.</td>
</tr>
<tr>
<td>&amp;DSNTYPE</td>
<td>Data set name type (for example: EXT, HFS, LIBRARY, PDS, or null).</td>
</tr>
<tr>
<td>&amp;DSORG</td>
<td>Data set organization.</td>
</tr>
<tr>
<td>&amp;DSOWNER</td>
<td>Owner or group considered to be the data set owner (available only if the RACF component of z/OS Security Server is installed).</td>
</tr>
<tr>
<td>&amp;DSTYPE</td>
<td>Data set type (for example, GDS, PERM, or TEMP).</td>
</tr>
<tr>
<td>&amp;EXPDT</td>
<td>Expiration date.</td>
</tr>
<tr>
<td>&amp;FLASHCPY</td>
<td>Data set allocated to FlashCopy storage group.</td>
</tr>
<tr>
<td>&amp;GROUP</td>
<td>Group identifier from the JOB statement.</td>
</tr>
<tr>
<td>&amp;HLQ</td>
<td>High-level qualifier of the data set name.</td>
</tr>
<tr>
<td>&amp;JOB</td>
<td>Job name, started task name, or TSO user ID from the JOB statement.</td>
</tr>
<tr>
<td>&amp;LLQ</td>
<td>Low-level qualifier of the data set name.</td>
</tr>
<tr>
<td>&amp;MAXSIZE</td>
<td>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 182 for more information about this value.</td>
</tr>
<tr>
<td>&amp;NQUAL</td>
<td>Number of qualifiers in the data set name.</td>
</tr>
<tr>
<td>&amp;NVOL</td>
<td>Number of output volumes specified by the user.</td>
</tr>
<tr>
<td>&amp;PGM</td>
<td>Program name from the EXEC card.</td>
</tr>
<tr>
<td>&amp;RECORG</td>
<td>Data set record organization.</td>
</tr>
<tr>
<td>&amp;RETPD</td>
<td>Retention period.</td>
</tr>
<tr>
<td>&amp;SIZE</td>
<td>Size of the data set in kilobytes. See “Using SIZE and MAXSIZE variables” on page 182 for more information about this value.</td>
</tr>
<tr>
<td>&amp;UNIT</td>
<td>Actual unit name (not esoteric names).</td>
</tr>
<tr>
<td>&amp;USER</td>
<td>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.</td>
</tr>
<tr>
<td>&amp;XMODE</td>
<td>Execution mode (for example, TSO, BATCH, or TASK).</td>
</tr>
</tbody>
</table>
ACS variables available during RESTORE and CONVERTV processing

When ACS is invoked during DFSMSdss RESTORE or CONVERTV processing, the variables shown in Table 15 are passed to the ACS routines.

Table 15. Variables Passed to ACS Routines during DFSMSdss Restore and CONVERTV Processing

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;ACSENVIR</td>
<td>Environment in which ACS was invoked. Set to 'RECOVER' for RESTORE. Set to 'CONVERT' for CONVERTV.</td>
</tr>
<tr>
<td>&amp;ALLVOL</td>
<td>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 &amp;ANYVOL, but when used in a comparison, returns true only if all volume serial numbers satisfy the condition. &amp;ALLVOL is not available to the storage group ACS routine when VOLCOUNT(ANY) is specified.</td>
</tr>
<tr>
<td>&amp;ANYVOL</td>
<td>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 &amp;ALLVOL, but when used in a comparison, returns true if any volume serial numbers satisfy the condition. &amp;ANYVOL is not available to the storage group ACS routine when VOLCOUNT(ANY) is specified.</td>
</tr>
<tr>
<td>&amp;APPLIC</td>
<td>Application identifier associated with the data set (available only if the RACF component of z/OS Security Server is installed).</td>
</tr>
<tr>
<td>&amp;DEF_DATACLAS</td>
<td>Default data class name (available only if the RACF component of z/OS Security Server is installed).</td>
</tr>
<tr>
<td>&amp;DEF_MGMTCLAS</td>
<td>Default management class name (available only if the RACF component of z/OS Security Server is installed).</td>
</tr>
<tr>
<td>&amp;DEF_STORCLAS</td>
<td>Default storage class name (available only if the RACF component of z/OS Security Server is installed).</td>
</tr>
<tr>
<td>&amp;DSN</td>
<td>Data set name.</td>
</tr>
<tr>
<td>&amp;DSNTYPE</td>
<td>Data set name type (for example: EXT, HFS, LIBRARY, PDS, or null).</td>
</tr>
<tr>
<td>&amp;DSORG</td>
<td>Data set organization.</td>
</tr>
<tr>
<td>&amp;DSOWNER</td>
<td>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).</td>
</tr>
<tr>
<td>&amp;DSTYPE</td>
<td>Data set type (for example, GDS, PERM, or TEMP).</td>
</tr>
<tr>
<td>&amp;EXPDT</td>
<td>Expiration date.</td>
</tr>
<tr>
<td>&amp;GROUP</td>
<td>Group identifier from the JOB statement. (This variable is passed to ACS routines during processing of RESTORE only, not CONVERTV.)</td>
</tr>
<tr>
<td>&amp;HLQ</td>
<td>High-level qualifier of the data set name.</td>
</tr>
<tr>
<td>&amp;LLQ</td>
<td>Low-level qualifier of the data set name.</td>
</tr>
<tr>
<td>&amp;MAXSIZE</td>
<td>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 182 for more information about this value.</td>
</tr>
<tr>
<td>&amp;NQUAL</td>
<td>Number of qualifiers in the data set name.</td>
</tr>
</tbody>
</table>
Table 15. Variables Passed to ACS Routines during DFSMSdss Restore and CONVERTV Processing (continued)

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;NVOL</td>
<td>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.</td>
</tr>
<tr>
<td>&amp;RECORG</td>
<td>Data set record organization.</td>
</tr>
<tr>
<td>&amp;RETPD</td>
<td>Retention period.</td>
</tr>
<tr>
<td>&amp;SIZE</td>
<td>Size of the data set in kilobytes. See &quot;Using SIZE and MAXSIZE variables&quot; for more information about this value.</td>
</tr>
<tr>
<td>&amp;UNIT</td>
<td>Actual unit name (not esoteric names).</td>
</tr>
<tr>
<td>&amp;USER</td>
<td>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.)</td>
</tr>
</tbody>
</table>

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 zSeries 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 zSeries partitions and volumes, and also how to use a z/OS system to back-up Linux partitions that are attached to a Linux for zSeries image:
- “Preparing to work with Linux volumes”
- “Backing up a Linux volume with partitions” on page 187
- “Using DFSMSdss dump and restore commands” on page 188.

This information is intended for storage administrators who are familiar with Linux for zSeries. 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 zSeries 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 z/OS, 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"
- "Using fdasd to partition a Linux volume" on page 187

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:
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.V<volser>.PART000x.NATIVE for a data partition, or LINUX.V<volser>.PART000x.SWAP for a swap partition.
Here, volser is the volume serial number assigned to the volume when ddfmfmt formatted the volume. fdasd can change the volser, too. The volser must be unique for z/OS to process it.

The x in PART000x is most likely the partition number, minus one. For example, a Linux partition such as /dev/dasd/0198/part2 would be known to z/OS as the data set LINUX.VLNX200.PART0001.NATIVE, where LNX200 is the 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:

<table>
<thead>
<tr>
<th>Data Set Names</th>
<th>Partition Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINUX.Vvolser.PART0000.type</td>
<td>/dev/dasd/yyyy/part1</td>
</tr>
<tr>
<td>LINUX.Vvolser.PART0001.type</td>
<td>/dev/dasd/yyyy/part2</td>
</tr>
<tr>
<td>LINUX.Vvolser.PART0002.type</td>
<td>/dev/dasd/yyyy/part3</td>
</tr>
</tbody>
</table>

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 zSeries 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 195.

This topic contains the following examples for your reference:
- “Example 1. DUMP FULL” on page 189
- “Example 2. DUMP FULL with CONCURRENT COPY” on page 190
- “Example 3. DUMP DATASET” on page 190
- “Example 4. COPY FULL” on page 191
- “Example 5. COPY FULL COPYVOLID ALLEXCP” on page 192
- “Example 6. RESTORE FULL” on page 192
- “Example 7. RESTORE DATASET” on page 193
- “Example 8. COPYDUMP” on page 195.
Notes:
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 4 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 4. 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 5 on page 190 shows an example of the JCL.
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 6 shows an example of the JCL.

```
//LXD2D1XX JOB ,"IBMUSER",MSGLEVEL=(1,1),TIME=(5,0),REGION=4096K,
// MSGCLASS=H,CLASS=A
//STEPT02 EXEC PGM=ADRDSSU
//SYSPRINT DD SYSSUT=
//SOURCE DD UNIT=3390, VOL=SER=LNX200, DISP=OLD
//TARGET1 DD UNIT=TAPE, VOL=(PRIVAT, SER=11111), 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 5. Sample JCL for dumping two or more output tapes at the same time.

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 7 on page 191 shows an example of the JCL.
You can also dump all of the Linux partitions. Figure 8 shows an example of the JCL.

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 9 on page 192 shows an example of the JCL.
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 10 shows an example of the JCL.

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 11 on page 193 shows an example of the JCL.
In Figure 11, observe the following:

v The Linux volume is *volser* LNX200
v The dump that was stored in data set TDS.DUMP200 is being restored to LNX200
v The SOURCE DD statement identifies where DFSMSdss is to find the information to restore
v 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 LNX200 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 12 shows an example of the JCL.

In Figure 12 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 12 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.VLNX200.PART0000.NATIVE), fdasd renames the other partitions (fdasd
subtracts one from the former name). part2 becomes part1 and part3 becomes
part2.

The data set names change, too. After fdasd deletes
LINUX.VLNX200.PART0000.NATIVE, it renames LINUX.VLNX200.PART0001.NATIVE to
LINUX.VLNX200.PART0000.NATIVE. If you then use DFSMSdss to restore the first
partition (named LINUX.VLNX200.PART0000.NATIVE), you will lose the second
partition.

You can also use the RENAMEUNCONDITIONAL keyword to change the names
of the data sets that you are restoring. Figure 13 shows an example of the JCL.

```
//LXD2S1XX JOB 'IBMUSER',MSGLEVEL=(1,1),TIME=(5,0),REGION=0M,
// MSGCLASS=A,CLASS=A
//STEP00 EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*"  
//SOURCE DD UNIT=3390,DISP=OLD,DSN=TDS.DUMP200
//TARGET DD UNIT=3390,VOL=SER=LNX300,DISP=OLD
//SYSIN DD *
RESTORE INDDNAME(SOURCE) OUTDDNAME(TARGET) -
DATASET(INCLUDE(LINUX.VLNX200.PART0001.NATIVE))
RENAMEUNCONDITIONAL(LINUX.VLNX200.PART0001,NATIVE,-
LINUX.VLNX300.PART0001))"
```

---

Figure 13. 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 14 on page 195 shows an example of the JCL you might use to restore only
the first partition, LINUX.VLNX200.PART0000.NATIVE.
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 15 shows an example of the JCL.

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 rout 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 187).

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 Chapter 19, “DFSMSdss stand-alone services,” on page 509.
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 16 on page 198) and ADRTAPB (Table 18 on page 199).

For information about ADRUFO, refer to z/OS DFSMS Installation Exits.

For information about ADREID0, refer to “ADREID0 data area” on page 604.

Format of the DFSMSdss dump data set

- 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 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 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 198.
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.

Notes:
1. For each data set, repeat items 3 on page 197 through 7.
2. For each volume of the data set, repeat items 4 on page 197 and 6.

Every record on a DFSMSdss dump data set is described by the "ADRTAPB data area" on page 199.

### ADRBMB data area

**Table 16. ADRBMB Mapping Macro**

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0)</td>
<td>0</td>
<td>CHARACTER</td>
<td>*</td>
<td>ADRBMB</td>
<td>BITMAP LAYOUT, CONTIGUOUS BITS FOR EACH TRACK, 1=OPERATE ON CORRESPONDING TRACK, 0=SKIP TRACK</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>0</td>
<td>CHARACTER</td>
<td>12</td>
<td>BMHDR</td>
<td>BITMAP HEADER</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>0</td>
<td>CHARACTER</td>
<td>4</td>
<td>BMDID</td>
<td>BLK IDENTIFIER EBCDIC &quot;BMBB&quot;</td>
</tr>
<tr>
<td>4</td>
<td>(4)</td>
<td>4</td>
<td>ADDRESS</td>
<td>4</td>
<td>BMNTPTR</td>
<td>ADDR OF NEXT BITMAP SEGMENT/0</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>4</td>
<td>UNSIGNED</td>
<td>4</td>
<td>BMNTRK</td>
<td>NUMBER OF TRACKS MAPPED BY SEGMENT</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2</td>
<td>BMNTRKTP</td>
<td>NUMBER OF TRACKS MAPPED BY SEGMENT</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>CHARACTER</td>
<td>2</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>12</td>
<td>(C)</td>
<td>0</td>
<td>BITSTRING</td>
<td>0</td>
<td>BMBITMAP</td>
<td>BITMAP LAYOUT, CONTIGUOUS BITS FOR EACH TRACK, 1=OPERATE ON CORRESPONDING TRACK, 0=SKIP TRACK</td>
</tr>
</tbody>
</table>

**Note:**

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0)</td>
<td>0</td>
<td>STRUCTURE</td>
<td>*</td>
<td>ADRCMB</td>
<td>CHUNK MAP BLOCK</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>0</td>
<td>CHARACTER</td>
<td>12</td>
<td>CMHDR</td>
<td>CHUNK MAP HEADER</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>0</td>
<td>CHARACTER</td>
<td>4</td>
<td>CMID</td>
<td>BLOCK EYECATCHER</td>
</tr>
<tr>
<td>4</td>
<td>(4)</td>
<td>4</td>
<td>ADDRESS</td>
<td>4</td>
<td>CMNTPTR</td>
<td>PTR TO NEXT CHUNKMAP</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>4</td>
<td>UNSIGNED</td>
<td>4</td>
<td>CMNCHKNK</td>
<td>NUMBER OF CHUNKS MAPPED</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2</td>
<td>CMNCHKTP</td>
<td>SHIFTED NUMBER OF CHunks</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>CHARACTER</td>
<td>2</td>
<td>*</td>
<td>RESERVED BY THIS SEGMENT</td>
</tr>
<tr>
<td>12</td>
<td>(C)</td>
<td>0</td>
<td>BITSTRING</td>
<td>0</td>
<td>CBMBITMAP</td>
<td>BITMAP LAYOUT, CONTIG BITS FOR EACH 21 CYL CHUNK TO BE PROCESSED</td>
</tr>
</tbody>
</table>

**ADRBMB constants**

**Table 17. ADRBMB Mapping Macro**

<table>
<thead>
<tr>
<th>Length</th>
<th>Type</th>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>CHARACTER</td>
<td>CMBB</td>
<td>ADRCMD</td>
<td>BLOCK IDENTIFIER</td>
</tr>
<tr>
<td>4</td>
<td>DECIMAL</td>
<td>197</td>
<td>CMAXSEG#</td>
<td>MAX NUM OF BITMAP</td>
</tr>
</tbody>
</table>
### ADRBMB cross-reference

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRBMB</td>
<td>0</td>
</tr>
<tr>
<td>ADRCMB</td>
<td>0</td>
</tr>
<tr>
<td>ADRTSMB</td>
<td>0</td>
</tr>
<tr>
<td>BMBITMAP</td>
<td>12</td>
</tr>
<tr>
<td>BMHDR</td>
<td>0</td>
</tr>
<tr>
<td>BMID</td>
<td>0</td>
</tr>
<tr>
<td>BMNPTR</td>
<td>4</td>
</tr>
<tr>
<td>BMNTRK</td>
<td>8</td>
</tr>
<tr>
<td>CMBITMAP</td>
<td>12</td>
</tr>
<tr>
<td>CMHDR</td>
<td>0</td>
</tr>
<tr>
<td>CMID</td>
<td>0</td>
</tr>
<tr>
<td>CMNCHKTP</td>
<td>8</td>
</tr>
<tr>
<td>CMNCHNK</td>
<td>8</td>
</tr>
<tr>
<td>CMNPTR</td>
<td>4</td>
</tr>
<tr>
<td>TSBMTP</td>
<td>8</td>
</tr>
<tr>
<td>TSMCHKCC</td>
<td>4</td>
</tr>
<tr>
<td>TSMHDR</td>
<td>0</td>
</tr>
<tr>
<td>TSMID</td>
<td>0</td>
</tr>
</tbody>
</table>

### ADRTAPB data area

**Table 18. ADRTAPB Mapping Macro**

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>DEC</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0)</td>
<td>(0)</td>
<td>STRUCTURE</td>
<td>16</td>
<td>ADRTAPB</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>(0)</td>
<td>CHARACTER</td>
<td>6</td>
<td>DTPSCHK</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>(0)</td>
<td>SIGNED</td>
<td>4</td>
<td>DTPSEQNO</td>
<td>SEGMENT SEQUENCE NUMBER</td>
</tr>
<tr>
<td>4</td>
<td>(4)</td>
<td>(4)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTPNOSEG</td>
<td>NUM OF SEGMENTS PER RECORD</td>
</tr>
<tr>
<td>5</td>
<td>(5)</td>
<td>(5)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTPSEQNO</td>
<td>SEGMENT NUM OF RECORD</td>
</tr>
<tr>
<td>6</td>
<td>(6)</td>
<td>(6)</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTPSEGLN</td>
<td>SEGMENT LENGTH INCL PREFIX</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>(8)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTPFFXLN</td>
<td>LENGTH OF PREFIX(CONSTANT 16)</td>
</tr>
<tr>
<td>9</td>
<td>(9)</td>
<td>(9)</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTPDMPID</td>
<td>TYPE OF DUMP ID</td>
</tr>
<tr>
<td>1...</td>
<td>1...</td>
<td>1...</td>
<td>DTPFFULD</td>
<td>80</td>
<td>80 - FULL DUMP</td>
<td></td>
</tr>
<tr>
<td>1...</td>
<td>1...</td>
<td>1...</td>
<td>DTPPARTD</td>
<td>40</td>
<td>40 - PARTIAL DUMP</td>
<td></td>
</tr>
<tr>
<td>...1.</td>
<td>...1.</td>
<td>...1.</td>
<td>DTPDASDD</td>
<td>20</td>
<td>20 - DATASET DUMP</td>
<td></td>
</tr>
</tbody>
</table>

Mapping of main prefix area (first thing in every record).
### ADRTAPB Data Area

**Table 18. ADRTAPB Mapping Macro (continued)**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0)</td>
<td></td>
<td>STRUCTURE</td>
<td>46 DTVOL</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>4</td>
<td>CHARACTER</td>
<td>4 DTVTOCB</td>
</tr>
<tr>
<td>4</td>
<td>(4)</td>
<td>4</td>
<td>CHARACTER</td>
<td>4 DTVTOCE</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>4</td>
<td>CHARACTER</td>
<td>4 DTVALPSZ</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2 DTVLOGCY</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2 DTTPKCY</td>
</tr>
<tr>
<td>12</td>
<td>(C)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2 DTVBLKSZ</td>
</tr>
<tr>
<td>11</td>
<td>(B)</td>
<td>1</td>
<td>BIT(8)</td>
<td>1 DTPRCFL1</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>1</td>
<td>BIT(8)</td>
<td>1 DTPRCID1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPVHDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPTHDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPBM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPDSNL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPTRK0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPDMPID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPDHDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPVTOC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPVOLD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPDATA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPVTRLR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPDTRLR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPVVDS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTPSBHDR</td>
</tr>
<tr>
<td>9</td>
<td>(C)</td>
<td>2</td>
<td>CHARACTER</td>
<td>2 DTVDDISP</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2 DTPDSDSP</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2 DTPENDNTR</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2 DTPENDTR</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2 DTPBWOE</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>CHARACTER</td>
<td>2 DTPDSNL</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>UNSIGNED</td>
<td>2 DTPSELFDO</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>2</td>
<td>CHARACTER</td>
<td>2 DTPDATA</td>
</tr>
</tbody>
</table>

REST OF VOLUME HEADER RECORD (follows ADRTAPB).
### ADRTAPB Data Area

#### Table 18. ADRTAPB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>(E)</td>
<td>UNSIGNED</td>
<td>DTVMAXCB</td>
</tr>
<tr>
<td>16</td>
<td>(10)</td>
<td>CHARACTER</td>
<td>DTVSERL</td>
</tr>
<tr>
<td>22</td>
<td>(16)</td>
<td>CHARACTER</td>
<td>DTVTOCI</td>
</tr>
<tr>
<td>23</td>
<td>(17)</td>
<td>CHARACTER</td>
<td>DTVTIMD</td>
</tr>
<tr>
<td>27</td>
<td>(1B)</td>
<td>CHARACTER</td>
<td>DTVTIME</td>
</tr>
<tr>
<td>31</td>
<td>(1F)</td>
<td>CHARACTER</td>
<td>DTVDEVTY</td>
</tr>
<tr>
<td>35</td>
<td>(23)</td>
<td>UNSIGNED</td>
<td>DTMODNO</td>
</tr>
<tr>
<td>36</td>
<td>(24)</td>
<td>BIT(8)</td>
<td>DTVIND1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>(25)</td>
<td>UNSIGNED</td>
<td>DTVBMSZ</td>
</tr>
<tr>
<td>39</td>
<td>(27)</td>
<td>BIT(8)</td>
<td>DTVIND2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>(28)</td>
<td>UNSIGNED</td>
<td>DTVLEN</td>
</tr>
<tr>
<td>42</td>
<td>(2A)</td>
<td>UNSIGNED</td>
<td>DTVVERNO</td>
</tr>
<tr>
<td>43</td>
<td>(2B)</td>
<td>UNSIGNED</td>
<td>DTVLVNO</td>
</tr>
<tr>
<td>44</td>
<td>(2C)</td>
<td>UNSIGNED</td>
<td>DTVXXTNO</td>
</tr>
<tr>
<td>45</td>
<td>(2D)</td>
<td>UNSIGNED</td>
<td>DTVXVTOF</td>
</tr>
</tbody>
</table>

Mapping of additional volume information (follows DTVOL).

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>0</td>
<td>DTVEND</td>
</tr>
</tbody>
</table>

Mapping of VVDS extents (in DTVEND).

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>10</td>
<td>DTVVXTNT(**)</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>10</td>
<td>DTVXXLEN</td>
</tr>
</tbody>
</table>

REST OF LOG DATA SET DUMP TAPE HEADER (follows ADRTAPB).

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>18</td>
<td>DTHDR</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTHTIMD</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>4</td>
<td>DTDHAY</td>
</tr>
<tr>
<td>4</td>
<td>CHARACTER</td>
<td>4</td>
<td>DTHTIME</td>
</tr>
</tbody>
</table>
### Table 18. ADRTAPB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTHIND2</td>
<td>DATA SET TYPE INDICATORS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTHGNVI</td>
<td>NO NON-VSAM DATA SETS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTHGVI</td>
<td>NO VSAM DATA SETS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTHGT64K</td>
<td>ON=MORE THAN 65535 DATA SETS ON VOLUME</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>9</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTHLEN</td>
<td>HEADER LEN</td>
</tr>
<tr>
<td>11</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTHVERNO</td>
<td>DFDSS VERSION NUMBER</td>
</tr>
<tr>
<td>12</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTHLVNO</td>
<td>DFDSS MODIFIC. NUMBER</td>
</tr>
<tr>
<td>13</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTHBLKSZ</td>
<td>MAX BLKSIZE</td>
</tr>
<tr>
<td>15</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTHINDS</td>
<td># DS IN LIST</td>
</tr>
<tr>
<td>17</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTHIND1</td>
<td>INDICATORS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTHFCMP</td>
<td>FILE COMPRESSED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTHUNLCD</td>
<td>UNALLOCATED SPACE DUMPED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTHSFER</td>
<td>SPHERE OPTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTHFHCMC</td>
<td>HARDWARE COMPRESSION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTHEFSAM</td>
<td>EF SAM DS HAVE DTTTRK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>

REST OF DATA SET NAME/CATALOG LIST (follows ADRTAPB).

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>45</td>
<td>DTLDSDN</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTILLEN</td>
<td>LENGTH OF DATA SET NAME</td>
</tr>
<tr>
<td>1</td>
<td>CHARACTER</td>
<td>44</td>
<td>DTLCAT</td>
<td>CATALOG NAME</td>
</tr>
</tbody>
</table>

REST OF DATA SET HEADER RECORD (follows ADRTAPB).

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>106</td>
<td>DTDSHDR</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTDLLEN</td>
<td>LENGTH OF DATA SET NAME</td>
</tr>
<tr>
<td>1</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTDCATLN</td>
<td>LENGTH OF CATALOG NAME</td>
</tr>
<tr>
<td>2</td>
<td>CHARACTER</td>
<td>2</td>
<td>DTDDSORG</td>
<td>DATA SET ORGA (FROM F1)</td>
</tr>
<tr>
<td>4</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTDOPTCD</td>
<td>DS OPTION CODE (FROM F1)</td>
</tr>
<tr>
<td>5</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTDNVOL</td>
<td>NUMBER VOLS FOR DATA SET</td>
</tr>
<tr>
<td>6</td>
<td>BIT(8)</td>
<td>1</td>
<td>DTDIND</td>
<td>DATA SET INDICATOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTDIPWD</td>
<td>PASSWORD SUPPLIED 1=YES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTDTPWD</td>
<td>TYPE PASSWORD 0=D.S. 1=CATALOG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTDRCF</td>
<td>RACF PROFILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTDRCFP</td>
<td>RACF PROFILE FLAGS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTDRCFD</td>
<td>1 = RACF DISCRETE PROFILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTDRCFG</td>
<td>1 = RACF GENERIC PROFILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTDALIAS</td>
<td>1 = USER CATALOG ALIAS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTDSPER</td>
<td>1 = SPHERE RECORD FOLLOWS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTDSMS</td>
<td>1=SMS MANAGED DATA SET</td>
</tr>
<tr>
<td>7</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTDPWD</td>
<td>PASSWORD</td>
</tr>
<tr>
<td>15</td>
<td>CHARACTER</td>
<td>44</td>
<td>DTDCAT</td>
<td>CATALOG NAME</td>
</tr>
<tr>
<td>59</td>
<td>CHARACTER</td>
<td>44</td>
<td>DTDDSN</td>
<td>DATA SET NAME</td>
</tr>
<tr>
<td>103</td>
<td>CHARACTER</td>
<td>2</td>
<td>DTDVOLCT</td>
<td>SMS VOL. CNT (FROM BCS)</td>
</tr>
<tr>
<td>103</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTDVCTD</td>
<td>VOLCOUNT FOR DATA COMPONENT OR NONVSAM DATA SET</td>
</tr>
</tbody>
</table>
### ADRTAPB Data Area

#### Table 18. ADRTAPB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>104</td>
<td>104</td>
<td>68</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTDVCTI</td>
<td>VOLCOUNFT FOR INDEX COMPONENT OR ZERO FOR NO INDEX</td>
</tr>
<tr>
<td>105</td>
<td>105</td>
<td>69</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTDIND2</td>
<td>DATA SET INDICATOR 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTDIAESP</td>
<td>AIX® &amp; PART OF A SPHERE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTDCDF</td>
<td>1=COMMON DATA FORMAT DSET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTDPDSE</td>
<td>1=PDSE DATA SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTDNTALL</td>
<td>1=DUMPED WITHOUT USING ALLD OR ALLX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTDSAI</td>
<td>1=DS ADTL INFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTDCDIF</td>
<td>1=VSAM INDEXED DATA SET DUMPED USING VALIDATE OPTION (INDEX NOT DUMPED, DATA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTDPDSET</td>
<td>CI'S IN ORDER)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DTDSMD</td>
<td>USE SYSTEM DATA MOVER</td>
</tr>
</tbody>
</table>

Mapping of sphere information (in DTSAIXS).

#### SPHERE RECORD FOR CATALOG FILTER DUMP (follows ADRTAPB).

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>STRUCTURE</td>
<td>4</td>
<td>DTSPHERE</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SIGNED</td>
<td>4</td>
<td>DTSLEN</td>
<td>LENGTH OF SPHERE RECORD</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>STRUCTURE</td>
<td>102</td>
<td>DTSINFO</td>
<td>SPHERE INFORMATION</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHARACTER</td>
<td>44</td>
<td>DTSAIXNM</td>
<td>AIX NAME</td>
</tr>
<tr>
<td>44</td>
<td>44</td>
<td>2C</td>
<td>CHARACTER</td>
<td>44</td>
<td>DTSPATHN</td>
<td>PATH NAME</td>
</tr>
<tr>
<td>88</td>
<td>88</td>
<td>58</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTSPATHA</td>
<td>PATH ATTRIBUTE</td>
</tr>
<tr>
<td>89</td>
<td>89</td>
<td>59</td>
<td>BIT(8)</td>
<td>1</td>
<td>DTSPATHF</td>
<td>PATH INFO FLAG</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>DTSPATHI</td>
<td>IF SET, PATH OWNERID IS CONTAINED IN THIS BLOCK (SEE DTSPTHON FOR DESCRIPTION)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTSPATHE</td>
<td>PATH EXPIRATION DATE IS CONTAINED IN THIS BLOCK (SEE DTSPTHEP FOR DESCRIPTION). IF NOT SET, IGNORE THE EXPIR FIELDS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTSPATHDP</td>
<td>PATH HAS PASSWORD THAT IS CONTAINED IN THIS BLOCK (SEE DTSEXPIR FOR DESCRIPTION). IF EXISTS, BEGINS AT DTSPATHD.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTSPRACF</td>
<td>PATH RACF FLAGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTSRACFD</td>
<td>1=DISCRETE PROFILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DTSRACFG</td>
<td>1=GENERIC PROFILE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>NOT USED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>90</td>
<td>5A</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTSPTHON</td>
<td>FORMAT OF OWNERID</td>
</tr>
<tr>
<td>98</td>
<td>98</td>
<td>62</td>
<td>CHARACTER</td>
<td>4</td>
<td>DTSPTHEP</td>
<td>EXPIRATION DATE FORMAT:</td>
</tr>
<tr>
<td>98</td>
<td>98</td>
<td>62</td>
<td>CHARACTER</td>
<td>3</td>
<td>DTSEXPIR</td>
<td>YYDDDF FORMAT</td>
</tr>
<tr>
<td>98</td>
<td>98</td>
<td>62</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTSEYEAR</td>
<td>YY 8 BIT IN DECIM</td>
</tr>
<tr>
<td>99</td>
<td>99</td>
<td>63</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTSERDAY</td>
<td>12-BITS DAY OF IN DECIMAL &amp; 4-BIT SIGN CHAR THAT ALLOWS TO BE UNPACKED</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>65</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTSEXNY</td>
<td>CENTURY BYTE IN DECIMAL</td>
</tr>
</tbody>
</table>

Mapping of security info table (in DTSPATHD).

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>STRUCTURE</td>
<td>52</td>
<td>DTSPASSW</td>
<td>SECURITY INFO TABLE</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTSMSTRP</td>
<td>MASTER PASSWORD. MUST HAVE IN ORDER TO HAVE ANY OTHER PASSWORDS.</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTSCNTLP</td>
<td>CONTROL INTERVAL PASSWORD</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>10</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTSPDAT</td>
<td>UPDATE PASSWORD</td>
</tr>
</tbody>
</table>
## Table 18. ADRTAPB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>(18)</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTSREADP  READ PASSWORD</td>
</tr>
<tr>
<td>32</td>
<td>(20)</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTSCODEN  CODE NAME</td>
</tr>
<tr>
<td>40</td>
<td>(28)</td>
<td>SIGNED</td>
<td>2</td>
<td>DTNSNUMAT NUMBER OF ATTEMPTS</td>
</tr>
<tr>
<td>42</td>
<td>(2A)</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTSAUTNM  ADDR OF AUTHORIZATION MOD</td>
</tr>
<tr>
<td>50</td>
<td>(32)</td>
<td>SIGNED</td>
<td>2</td>
<td>DTSAUTHR AUTHORIZATION REC LENGTH</td>
</tr>
</tbody>
</table>

### COMMON DATA FORMAT DATA SET ATTRIBUTES (follows DTDHSHDR).

- **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**

### 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 (UCBTTY4)**
- **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. ..... DTMCAVF VOLUME HAS INDEXED VTOC
  - ...1. ..... DTMftime Time stamp follows VVR
  - .... 1... DTMFBOT RLS time stamps are BWO
- **22 (16) BIT(11) POS(6) 2 * unused**
- **24 (18) UNSIGNED 1 DTM#VVR # 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 NVRS DATA SET CLUSTER.**
- **0 (0) CHARACTER 0 *
### Table 18. ADRTAPB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>12</td>
<td>DTBTMR</td>
<td>REST OF RECORD THAT MAPS THE TRACKS DUMPED (follows ADRTAPB).</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>12</td>
<td>DTBHDR</td>
<td>RESERVED</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>4</td>
<td>DTBMID</td>
<td>ID OF THE BLOCK = BMBB</td>
</tr>
<tr>
<td>4</td>
<td>SIGNED</td>
<td>4</td>
<td>DTBADDR</td>
<td>@ OF NEXT BMBB SEGMENT COPIED FROM ADRBMB BLOCK</td>
</tr>
<tr>
<td>8</td>
<td>UNSIGNED</td>
<td>4</td>
<td>DTBTRK#</td>
<td># OF TRKS MAPPED IN BMBB SEGMENT</td>
</tr>
<tr>
<td>8</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTBTRK#P</td>
<td># OF TRKS MAPPED IN BMBB SEGMENT</td>
</tr>
<tr>
<td>10</td>
<td>CHARACTER</td>
<td>2</td>
<td>*</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>

Mapping of the bitmap data (follows DTBTMR).

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>*</td>
<td>DTBBITM</td>
<td>BITMAP</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>24</td>
<td>DTTTRK</td>
<td>MAPS THE TRACK</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>16</td>
<td>DTTHDR</td>
<td>HEADER FOR EACH TRACK SEGMENT</td>
</tr>
<tr>
<td>0</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTTTRKLN</td>
<td>LENGTH OF DATA ON TRK</td>
</tr>
<tr>
<td>2</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTTTRKID</td>
<td>TRACK INDICATORS</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>DTTIOER</td>
<td>I/O ERROR ON TRK</td>
</tr>
<tr>
<td>.1</td>
<td></td>
<td></td>
<td>DTTTROVF</td>
<td>LAST REC ON TRK IS OVFL REC (maintained for restore compatibility only, do not set)</td>
</tr>
<tr>
<td>..</td>
<td></td>
<td></td>
<td>DTTTCMP</td>
<td>IF ON, TRACK COMPRESSED</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td>DTTVFST</td>
<td>FIRST VVDS RECORD</td>
</tr>
<tr>
<td>....</td>
<td></td>
<td></td>
<td>DTTINVT</td>
<td>INVALID TRACK FORMAT</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td>DTTSTAT</td>
<td>USER STATISTICAL RECORD</td>
</tr>
<tr>
<td>3</td>
<td>UNSIGNED</td>
<td>4</td>
<td>DTTCCHH</td>
<td>CCHH OF TRACK 285VD</td>
</tr>
<tr>
<td>7</td>
<td>SIGNED</td>
<td>4</td>
<td>DTTLRCNT</td>
<td>LR COUNT FOR THE DS (THIS FIELD IS FILLED IN ON THE LAST DATA CA TRACKS FOR A VS DUMPED BY VSAM I/O)</td>
</tr>
<tr>
<td>11</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTTXDLEN</td>
<td>EXTRA ENC DATA LEN</td>
</tr>
<tr>
<td>12</td>
<td>CHARACTER</td>
<td>4</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>16</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTT50DAT</td>
<td>RCRD 0 DATA, ONLY ON 1ST SEG</td>
</tr>
<tr>
<td>24</td>
<td>CHARACTER</td>
<td>0</td>
<td>DTTBODY</td>
<td>R1- RN RECORDS ON TRACK</td>
</tr>
</tbody>
</table>

Count field format (in track image data).

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>8</td>
<td>DTTCKD</td>
<td>CNT, KEY &amp; DATA FIELDS ON TRK</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTTCNT</td>
<td>COUNT FIELD</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>5</td>
<td>DTTCCCHHR</td>
<td>CCHHR OF RECORD</td>
</tr>
<tr>
<td>0</td>
<td>UNSIGNED</td>
<td>4</td>
<td>DTTCCCHH</td>
<td>CCHH OF RECORD 285VD</td>
</tr>
<tr>
<td>4</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTTCRCRD</td>
<td>R OF RECORD</td>
</tr>
<tr>
<td>5</td>
<td>UNSIGNED</td>
<td>1</td>
<td>DTTKELEN</td>
<td>KEY LENGTH</td>
</tr>
<tr>
<td>6</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTTDATLN</td>
<td>DATA LENGTH</td>
</tr>
</tbody>
</table>

Key or Data field mapping (follows count field).

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>*</td>
<td>DTTKD</td>
<td>KEY AND DATA FIELDS</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>
### ADRTAPB Data Area

#### Table 18. ADRTAPB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>6</td>
<td>DTRTLR</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>6</td>
<td>DTRSERL</td>
<td>VOLUME SERIAL OF FILE (USED FOR PHYSICAL DUMP ONLY) V26</td>
</tr>
<tr>
<td>0</td>
<td>SIGNED</td>
<td>4</td>
<td>DTRRECNT</td>
<td>NUMBER OF LOGICAL RECORDS DUMPED (USED FOR LOGICAL DATA SET DUMP ONLY) V26</td>
</tr>
<tr>
<td>4</td>
<td>CHARACTER</td>
<td>2</td>
<td>*</td>
<td>RESERVED - LOGICAL DUMP</td>
</tr>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>4</td>
<td>DTSMDHDR</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTSDLEN</td>
<td>LEN OF SELF DESC REC</td>
</tr>
<tr>
<td>2</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTSDTYPE</td>
<td>TYPE OF SELF DESC REC</td>
</tr>
<tr>
<td>3</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTSDIND1</td>
<td>SELF DESC REC FLAGS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1...</td>
<td>DTSDLAST LAST SELF DESC REC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.111</td>
<td>RESERVE</td>
</tr>
<tr>
<td>4</td>
<td>CHARACTER</td>
<td>0</td>
<td>DTSDEND</td>
<td>START SELF DESC REC ON A WORD BOUNDARY</td>
</tr>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>356</td>
<td>DTCIBPH</td>
<td>CIPHER BLOCK</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>4</td>
<td>DTCFLAG1</td>
<td>ENCRYPTION TYPES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1...</td>
<td>DTCCCTDES CLRTDES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.1...</td>
<td>DTCCA128 CLRAES128</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...1</td>
<td>DTCEDES ENCTDES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.111</td>
<td>RESERVE</td>
</tr>
<tr>
<td>1</td>
<td>CHARACTER</td>
<td>3</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>4</td>
<td>CHARACTER</td>
<td>16</td>
<td>DTCDATAS</td>
<td>SAMPLE DATA</td>
</tr>
<tr>
<td>20</td>
<td>CHARACTER</td>
<td>64</td>
<td>DTCRSAL</td>
<td>RSA LABEL</td>
</tr>
<tr>
<td>84</td>
<td>CHARACTER</td>
<td>256</td>
<td>DTCRDKL</td>
<td>RSA CIPHD DATA LABEL</td>
</tr>
<tr>
<td>340</td>
<td>SIGNED</td>
<td>4</td>
<td>DTICNCT</td>
<td>ITERATION COUNT</td>
</tr>
<tr>
<td>344</td>
<td>CHARACTER</td>
<td>8</td>
<td>DTCSALT</td>
<td>SALT FOR ADRPWKEY</td>
</tr>
<tr>
<td>352</td>
<td>SIGNED</td>
<td>4</td>
<td>DTCRDKLN</td>
<td>RSA CIPHD DATA LAB LN@DDE</td>
</tr>
</tbody>
</table>

COMPRESSION DICTIONARY BLOCK. Immediately follows a self-describing record header

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>16</td>
<td>DTCDCNT</td>
<td>FOLLOWS A DTSDHDR</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>16</td>
<td>DTCFLDS</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>SIGNED</td>
<td>4</td>
<td>DTCMAXD</td>
<td>MAX DICT SIZE</td>
</tr>
<tr>
<td>4</td>
<td>CHARACTER</td>
<td>12</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>*</td>
<td>DTCDICT</td>
<td>EXPANSION DICTIONARY AFTER DTCDCNT</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>0</td>
<td>*</td>
<td>AREA</td>
</tr>
</tbody>
</table>

EXTENDED VOLUME RECORD FOUND IN FIRST TAPE RECORD

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>52</td>
<td>DTSDEVOL</td>
<td>EXTENDED VOL SD REC</td>
</tr>
<tr>
<td>0</td>
<td>SIGNED</td>
<td>4</td>
<td>DTELOGCYL</td>
<td># OF CYLINDERS - EAV</td>
</tr>
<tr>
<td>4</td>
<td>UNSIGNED</td>
<td>2</td>
<td>DTELCYL</td>
<td>LARGE UNIT REGION</td>
</tr>
<tr>
<td>6</td>
<td>CHARACTER</td>
<td>2</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>8</td>
<td>UNSIGNED</td>
<td>4</td>
<td>DTCBITM#</td>
<td>SIZE OF CHUNK MAP</td>
</tr>
<tr>
<td>12</td>
<td>CHARACTER</td>
<td>4</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>16</td>
<td>UNSIGNED</td>
<td>8</td>
<td>DTEVBLOCKSIZE</td>
<td>OUTPUT BLOCKSIZE</td>
</tr>
<tr>
<td>16</td>
<td>UNSIGNED</td>
<td>4</td>
<td>DTEVBLKSZ</td>
<td>HIGH WORD BLKSIZE</td>
</tr>
<tr>
<td>20</td>
<td>UNSIGNED</td>
<td>4</td>
<td>DTEVBLKSZ</td>
<td>LOW WORD BLKSIZE WHEN DTVBLKSZ=0</td>
</tr>
<tr>
<td>24</td>
<td>CHARACTER</td>
<td>28</td>
<td>*</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>
## ADRTAPB Data Area

### Table 18. ADRTAPB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>52</td>
<td>DTSDEHDR</td>
<td>EXT TAPE SD RECORD</td>
</tr>
<tr>
<td>0 (0)</td>
<td>UNSIGNED</td>
<td>8</td>
<td>DTEHBLOCKSIZE</td>
<td>OUTPUT BLOCKSIZE</td>
</tr>
<tr>
<td>0 (0)</td>
<td>UNSIGNED</td>
<td>4</td>
<td>DTEHHIBLKSZ</td>
<td>HIGH WORD BLKSIZE</td>
</tr>
<tr>
<td>4 (4)</td>
<td>UNSIGNED</td>
<td>4</td>
<td>DTEHLBLKSZ</td>
<td>LOW WORD BLKSIZE WHEN DTHBLKSZ=0</td>
</tr>
<tr>
<td>8 (8)</td>
<td>CHARACTER</td>
<td>44</td>
<td>*</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>64</td>
<td>BFRPREFIX</td>
<td>BUFFER PREFIX AREA@V22H</td>
</tr>
<tr>
<td>0 (0)</td>
<td>UNSIGNED</td>
<td>2</td>
<td>BPSCOC</td>
<td>OFFSET IN BUFFER FROM BEGINNING OF R0 DATA TO COUNT FIELD IN ERROR</td>
</tr>
<tr>
<td>2 (2)</td>
<td>UNSIGNED</td>
<td>2</td>
<td>BPSCOD</td>
<td>OFFSET IN BUFFER FROM BEGINNING OF R0 DATA TO DATA FIELD IN ERROR</td>
</tr>
<tr>
<td>4 (4)</td>
<td>SIGNED</td>
<td>4</td>
<td>BPHWMRK</td>
<td>ADDRESS OF NEXT AVAILABLE BYTE (HIGH WATER MARK OF USED STORAGE)</td>
</tr>
<tr>
<td>8 (8)</td>
<td>CHARACTER</td>
<td>56</td>
<td>BPWORKA</td>
<td>WORK AREA FOR OPTIMIZE@V22H</td>
</tr>
<tr>
<td>64 (40)</td>
<td></td>
<td>0</td>
<td>BPBODY</td>
<td>LOCATION OF REST OF BUFFER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>*</td>
<td>DSAUTHR</td>
<td>AUTHORIZATION RECORD.</td>
</tr>
<tr>
<td>0 (0)</td>
<td>CHARACTER</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>68</td>
<td>DTCHKSUM</td>
<td>CHECKSUM RECORD</td>
</tr>
<tr>
<td>0 (0)</td>
<td>CHARACTER</td>
<td>1</td>
<td>DTCKTYPE</td>
<td>TYPE OF CHECKSUM</td>
</tr>
<tr>
<td>1</td>
<td>111 1111</td>
<td>*</td>
<td>DTCK_MD5</td>
<td>MD5 HASH</td>
</tr>
<tr>
<td>1 (1)</td>
<td>CHARACTER</td>
<td>3</td>
<td>*</td>
<td>UNUSED</td>
</tr>
<tr>
<td>4 (4)</td>
<td>CHARACTER</td>
<td>64</td>
<td>DTCKHASH</td>
<td>HASH VALUE</td>
</tr>
<tr>
<td>4 (4)</td>
<td>CHARACTER</td>
<td>16</td>
<td>DTCK_MD5HASH</td>
<td>MD5 HASH VALUE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>*</td>
<td>DTLDSCMP</td>
<td>COMPRESSED DATA SET NAME</td>
</tr>
<tr>
<td>0 (0)</td>
<td>CHARACTER</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

Mapping of sphere information (in DTSAIXS).

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>*</td>
<td>DTSAIXS</td>
<td>AIX SPHERE INFO</td>
</tr>
<tr>
<td>0 (0)</td>
<td>CHARACTER</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>*</td>
<td>DTSPATHD</td>
<td>START PATH SECURITY INFO</td>
</tr>
<tr>
<td>0 (0)</td>
<td>CHARACTER</td>
<td>0</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

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" DTSDLHRE | TAPE HDR EXTENSION SELF-DESC REC TYPE

CHECKSUM RECORD

---

Chapter 13. Format of the DFSMSdss dump data set  207
ADRTAPB Data Area

Table 18. ADRTAPB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;1&quot;</td>
<td>&quot;01&quot;</td>
<td>&quot;CHAR HEX&quot;</td>
<td>&quot;05&quot;</td>
<td>DTSDCHECKSUM</td>
<td>CHECKSUM SELF-DESC REC TYPE</td>
<td></td>
</tr>
<tr>
<td>&quot;2&quot;</td>
<td>&quot;02&quot;</td>
<td>&quot;DECIMAL&quot;</td>
<td>&quot;01&quot;</td>
<td>DTPVERC</td>
<td>Version</td>
<td></td>
</tr>
</tbody>
</table>

ADRTAPB constants

Table 19. ADRTAPB Mapping Macro

<table>
<thead>
<tr>
<th>Length</th>
<th>Type</th>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CONSTANT</td>
<td>X'01'</td>
<td>DTSDCIPH</td>
<td>CIPHER BLOCK SELF DESCRIBING RECORD CONSTANT</td>
</tr>
<tr>
<td>1</td>
<td>CONSTANT</td>
<td>X'02'</td>
<td>DTSDCDCT</td>
<td>COMPRESSION DICTIONARY SELF DESCRIBING RECORD CONSTANT</td>
</tr>
<tr>
<td>1</td>
<td>CONSTANT</td>
<td>X'03'</td>
<td>DTSVDOLE</td>
<td>EXTENDED VOLUME SELF-DESC REC TYPE</td>
</tr>
<tr>
<td>1</td>
<td>CONSTANT</td>
<td>X'04'</td>
<td>DTSHDHRE</td>
<td>TAPE HEADER EXTENSION SELF-DESC REC TYPE</td>
</tr>
</tbody>
</table>

ADRTAPB cross-reference

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRTAPB</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BFRPREFIX</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BPBODY</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>BPHWMRK</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>BPSOCO</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BPSCOD</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BPWORKA</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DSAUTHR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTBADDR</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTBBITM</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTBBMID</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTBHDIR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTBTMR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTBTRK#</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DTBTRK#P</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DTC#DIRB</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DTCB1TM#</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DTCAC128</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>DTCCTDES</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>DTCDDATAS</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>DTCDDCT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTCDFATT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTCDFIND</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTCDICT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTCETDES</td>
<td>0</td>
<td>20</td>
</tr>
</tbody>
</table>
### ADRTAPB Data Area

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTCFLAG1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTCFLDS</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTCHURPN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTCICNT</td>
<td>154</td>
<td></td>
</tr>
<tr>
<td>DTCIPHB</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTCMAXD</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTCRDKL</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>DTCRDKLN</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>DTCRSAL</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>DTCSALT</td>
<td>158</td>
<td></td>
</tr>
<tr>
<td>DTDAIXSP</td>
<td>69 80</td>
<td></td>
</tr>
<tr>
<td>DTDALIAS</td>
<td>6 04</td>
<td></td>
</tr>
<tr>
<td>DTDCAT</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>DTDCATLN</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DTDCDF</td>
<td>69 40</td>
<td></td>
</tr>
<tr>
<td>DTDDSN</td>
<td>3B</td>
<td></td>
</tr>
<tr>
<td>DTDDSORG</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DTDDIND</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>DTDDIND2</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>DTDPWD</td>
<td>6 80</td>
<td></td>
</tr>
<tr>
<td>DTDIRACF</td>
<td>6 20</td>
<td></td>
</tr>
<tr>
<td>DTDLLEN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTDNOIDX</td>
<td>69 04</td>
<td></td>
</tr>
<tr>
<td>DTDSNTALL</td>
<td>69 10</td>
<td></td>
</tr>
<tr>
<td>DTDSNVOL</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>DTDOPTCD</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTDPDSE</td>
<td>69 20</td>
<td></td>
</tr>
<tr>
<td>DTDPDSET</td>
<td>69 02</td>
<td></td>
</tr>
<tr>
<td>DTDPDSEX</td>
<td>4 80</td>
<td></td>
</tr>
<tr>
<td>DTDPWD</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>DTDRACFD</td>
<td>6 10</td>
<td></td>
</tr>
<tr>
<td>DTDRACFG</td>
<td>6 08</td>
<td></td>
</tr>
<tr>
<td>DTDRACFP</td>
<td>6 18</td>
<td></td>
</tr>
<tr>
<td>DTDSAI</td>
<td>69 08</td>
<td></td>
</tr>
<tr>
<td>DTDSAID</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTDSAIDDL</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTDSAIR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTDSDM</td>
<td>69 01</td>
<td></td>
</tr>
<tr>
<td>DTDSHDR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTDSMS</td>
<td>6 01</td>
<td></td>
</tr>
<tr>
<td>DTDSFER</td>
<td>6 02</td>
<td></td>
</tr>
<tr>
<td>DTDPFWD</td>
<td>6 40</td>
<td></td>
</tr>
<tr>
<td>DTDVCTD</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>DTDVCTI</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>DTDVOLCT</td>
<td>67</td>
<td></td>
</tr>
</tbody>
</table>
## ADRTAPB Data Area

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTEDSBLKSIZE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTEDSBLOCKSIZE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTEDSHIBLKSIZE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTELACYL</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTELOGCYL</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTEvBLKSIZE</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>DTEvBLOCKSIZE</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>DTEVHIBLKSIZE</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>DTHBLKSIZE</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>DTHDAY</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTHDR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTHDFSAM</td>
<td>11</td>
<td>08</td>
</tr>
<tr>
<td>DTHFCMP</td>
<td>11</td>
<td>80</td>
</tr>
<tr>
<td>DTHFHCMP</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>DTHGNV1</td>
<td>8</td>
<td>80</td>
</tr>
<tr>
<td>DTHGTr4K</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>DTHGV1</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>DTHIND1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>DTHIND2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DTHLEN</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>DTHLVLNO</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>DTHNDS</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>DTHSFER</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>DTHTIMD</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTHTIME</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTHUNLHC</td>
<td>11</td>
<td>40</td>
</tr>
<tr>
<td>DTHVERNO</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>DTLCAT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DTLDCOMP</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTLDSN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTLLEN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTM#DSCE</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>DTM#EXT</td>
<td>1A</td>
<td></td>
</tr>
<tr>
<td>DTM#VVRD</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>DTMBWOT</td>
<td>16</td>
<td>08</td>
</tr>
<tr>
<td>DTMCVAF</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>DTMDEVTY</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>DTMIND</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>DTMLOGCY</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>DTMMAXCY</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>DTMMINI</td>
<td>16</td>
<td>40</td>
</tr>
<tr>
<td>DTMMODNO</td>
<td>1B</td>
<td></td>
</tr>
<tr>
<td>DTMTIME</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>DTMTRKCP</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>DTMTRKCY</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>
## ADRTAPB Data Area

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTMVDATA</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTMVIRT</td>
<td>16</td>
<td>80</td>
</tr>
<tr>
<td>DTMVOL</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTMVOLSZ</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>DTMVSERL</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTPBODY</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>DTPBTM</td>
<td>A</td>
<td>40</td>
</tr>
<tr>
<td>DTPBWOE</td>
<td>B</td>
<td>08</td>
</tr>
<tr>
<td>DTPDA5DD</td>
<td>9</td>
<td>20</td>
</tr>
<tr>
<td>DTPDATA</td>
<td>A</td>
<td>08</td>
</tr>
<tr>
<td>DTPDDDSP</td>
<td>B</td>
<td>40</td>
</tr>
<tr>
<td>DTPDDISP</td>
<td>B</td>
<td>80</td>
</tr>
<tr>
<td>DTPDMPID</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>DTPDSHDR</td>
<td>A</td>
<td>20</td>
</tr>
<tr>
<td>DTPDNSL</td>
<td>A</td>
<td>40</td>
</tr>
<tr>
<td>DTPDTRLR</td>
<td>A</td>
<td>04</td>
</tr>
<tr>
<td>DTPDUMPC</td>
<td>B</td>
<td>04</td>
</tr>
<tr>
<td>DTPENDKR</td>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>DTPENDNT</td>
<td>B</td>
<td>20</td>
</tr>
<tr>
<td>DTPFULLD</td>
<td>9</td>
<td>80</td>
</tr>
<tr>
<td>DTPLOGCL</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>DTPNOSEG</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTPPARTD</td>
<td>9</td>
<td>40</td>
</tr>
<tr>
<td>DTPFFXLN</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DTPRCFL1</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>DTPRCID1</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>DTPSCHK</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTPSEGLN</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>DTPSEGNO</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>DTPSELFDF</td>
<td>B</td>
<td>02</td>
</tr>
<tr>
<td>DTPSEQNO</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTPSPHDR</td>
<td>A</td>
<td>01</td>
</tr>
<tr>
<td>DTPTHDR</td>
<td>A</td>
<td>80</td>
</tr>
<tr>
<td>DTPTRK0</td>
<td>A</td>
<td>20</td>
</tr>
<tr>
<td>DTPVER#</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>DTPVHDR</td>
<td>A</td>
<td>80</td>
</tr>
<tr>
<td>DTPVOLD</td>
<td>A</td>
<td>10</td>
</tr>
<tr>
<td>DTPVTOC</td>
<td>A</td>
<td>10</td>
</tr>
<tr>
<td>DTPVTRLR</td>
<td>A</td>
<td>04</td>
</tr>
<tr>
<td>DTPVVDS</td>
<td>A</td>
<td>02</td>
</tr>
<tr>
<td>DTTRECNT</td>
<td>0</td>
<td>c</td>
</tr>
<tr>
<td>DTRSERL</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTRTRLR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTSAIXNM</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTSAIXS</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
### ADRTAPB Data Area

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTSAUTHR</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>DTSAUTNM</td>
<td>2A</td>
<td></td>
</tr>
<tr>
<td>DTSCNTLP</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DTSCODEN</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>DTSEDSD</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTSDEND</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTSDEVOL</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTSDHDR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTDIND1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DTSDLAST</td>
<td>3 80</td>
<td></td>
</tr>
<tr>
<td>DTSDLLEN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTSJTYPE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DTSERDAY</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>DTSEXCNY</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>DTSEXPIR</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>DTSEYEAR</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>DTSINFO</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTSLEN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTMSTRP</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTNUMAT</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>DTSPASSW</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTSPATHA</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>DTSPATHD</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTSPATHE</td>
<td>59 40</td>
<td></td>
</tr>
<tr>
<td>DTSPATHF</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>DTSPATHI</td>
<td>59 80</td>
<td></td>
</tr>
<tr>
<td>DTSPATHN</td>
<td>2C</td>
<td></td>
</tr>
<tr>
<td>DTSPATHP</td>
<td>59 20</td>
<td></td>
</tr>
<tr>
<td>DTSHERE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTSPRACF</td>
<td>59 18</td>
<td></td>
</tr>
<tr>
<td>DTSPTHEP</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>DTSPTHON</td>
<td>5A</td>
<td></td>
</tr>
<tr>
<td>DTSRACFD</td>
<td>59 10</td>
<td></td>
</tr>
<tr>
<td>DTSRACFG</td>
<td>59 08</td>
<td></td>
</tr>
<tr>
<td>DTSREADP</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>DTSPDAT</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>DTTBODY</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>DTTCCCHH</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTTCCCHH</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DTTCCCHHR</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTTCKD</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTTCNT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTTCCCRD</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>DTTDATLN</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>DTTHDR</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
## ADRTAPB Data Area

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTTINVNT</td>
<td>2</td>
<td>08</td>
</tr>
<tr>
<td>DTTIOER</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>DTTKD</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTTKELEN</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>DTLRCNT</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>DTR0DAT</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>DTTSTAT</td>
<td>2</td>
<td>04</td>
</tr>
<tr>
<td>DTTTCMP</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>DTTTRK</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTTTRKID</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DTTTRKLN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTTTROVF</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>DTTVFRST</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>DTTXDLEN</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>DTVBLKSZ</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>DTVBMSZ</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>DTVCVPOL</td>
<td>24</td>
<td>10</td>
</tr>
<tr>
<td>DTVCVAF</td>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>DTVDAY</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>DTVDEVTY</td>
<td>1F</td>
<td></td>
</tr>
<tr>
<td>DTVEND</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTVFCMP</td>
<td>24</td>
<td>08</td>
</tr>
<tr>
<td>DTVFHCMP</td>
<td>27</td>
<td>02</td>
</tr>
<tr>
<td>DTVGGVNI</td>
<td>27</td>
<td>20</td>
</tr>
<tr>
<td>DTVGV</td>
<td>27</td>
<td>10</td>
</tr>
<tr>
<td>DTVIND1</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>DTVIND2</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>DTVLEN</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>DTVLNVI</td>
<td>27</td>
<td>80</td>
</tr>
<tr>
<td>DTVLOGCY</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DTVLF</td>
<td>24</td>
<td>02</td>
</tr>
<tr>
<td>DTVLV</td>
<td>27</td>
<td>40</td>
</tr>
<tr>
<td>DTVVLNO</td>
<td>2B</td>
<td></td>
</tr>
<tr>
<td>DTVMAXCB</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>DTVMINI</td>
<td>24</td>
<td>40</td>
</tr>
<tr>
<td>DTVMODNO</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>DTVOL</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTVOLSZ</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DTVSERL</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>DTVSMS</td>
<td>27</td>
<td>08</td>
</tr>
<tr>
<td>DTVSMSI</td>
<td>27</td>
<td>04</td>
</tr>
<tr>
<td>DTVTIMD</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>DTVTIME</td>
<td>1B</td>
<td></td>
</tr>
<tr>
<td>DTVTOCB</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTVTOCE</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
ADRTAPB Data Area

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTVTRKCY</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>DTVUNLCD</td>
<td>24</td>
<td>04</td>
</tr>
<tr>
<td>DTVVERNO</td>
<td>2A</td>
<td></td>
</tr>
<tr>
<td>DTVVIRT</td>
<td>24</td>
<td>80</td>
</tr>
<tr>
<td>DTVVTOCI</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>DTVVVDS</td>
<td>24</td>
<td>01</td>
</tr>
<tr>
<td>DTVVXLEN</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTVVXTNO</td>
<td>2C</td>
<td></td>
</tr>
<tr>
<td>DTVVXTNT</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>DTVVXTOF</td>
<td>2D</td>
<td></td>
</tr>
</tbody>
</table>
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 (ADRDSSU). 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 20 on page 240.

Sample JCL

```bash
//PATCH JOB...
//*
).*
//* SAMPLE JCL TO SET THE FLAGS IN ADRPATCH. *
//* *
//************************************************************
//ZAP EXEC PGM=AMASPZAP,PARM='IGNIDRFULL'
//SYSPRINT DD SYSOUT=* 
//SYSLIB DD DISP=SHR,DSN=LIBNAME.LINKLIB 
//SYSIN DD *
  NAME ADRDSSU ADRPATCH 
   VER offset value REP offset value/*
```

As an alternative to using the above JCL to set flags in ADRPATCH, you can customize certain DFSMSdss functions by temporarily setting patch bytes during DFSMSdss processing through a SET PATCH command. For information about the SET command, see “Controlling task processing” on page 502.

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 215) 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 215) as follows:

```
//SYSIN DD *
    NAME ADRDSSU ADRPATCH
    VER 09 00
    REP 09 FF
```

You can set a slip trap and use generalized trace facility (GTF) as follows:

1. Use AMBLIST to locate the displacement (xxxx) of label ADRSLIP1 within ADRDSSU. The displacement of label ADRSLIP1 is used in setting the slip trap.

```
//AMBLIST JOB...
//*
//******************************************************************************
//*
// SAMPLE JCL TO LOCATE THE DISPLACEMENT OF LABEL ADRSLIP1
//*
// WITHIN ADRDSSU. THE DISPLACEMENT OF ADRSLIP1 IS NEEDED
//*
// TO SET THE SLIP TRAP.
//*
//******************************************************************************
//LISTIT EXEC PGM=AMBLIST
//SYSPRINT DD SYSOUT*
//SYSLIB DD DISP=SHR,DSN=LIBNAME.LINKLIB
//SYSIN DD *
//*
```

2. Set a slip trap from the system console. The displacement label of ADRSLIP1 obtained with AMBLIST is ‘xxxx’. Registers 7 and 8 must be used as shown below:

   ```
   SLIP SET,IF,ACTION=TRACE,TRDATA=(STD,7R?,8R?),
   PVTMOD=(ADRDSSU,xxxx),JOBNAME=nnnnnnn,END
   ```

3. Start the GTF trace at the system console:
   a. Enter `S GTF`. 

b. Respond \texttt{TRACE=SLIP,USR} to message AHL125A (“SPECIFY TRACE OPTIONS”).
c. Respond \texttt{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.

\textbf{Notes:}
1. The duplicate key error occurs only after the first record with a duplicate key gets written to the data set. Second and subsequent records with the same key are recognized as duplicates.
2. Only the first 65 535 bytes of each duplicate record can be written to the trace data set.

\section*{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 \texttt{X'0A'} in ADRPATCH to the number of seconds to wait. Setting the field to \texttt{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 215) as follows:

\begin{verbatim}
//SYSIN DD *
   NAME ADRSSU ADRPATCH
   VER 0A 0000
   REP 0A 0078
\end{verbatim}

To disable lockout detection, use the SET PATCH command or modify the sample JCL (see “Sample JCL” on page 215) as follows:

\begin{verbatim}
//SYSIN DD *
   NAME ADRSSU ADRPATCH
   VER 0A 0000
   REP 0A FFFF
\end{verbatim}

\section*{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:

\begin{itemize}
\item \texttt{X'0D'} An indicator that new wait/retry values are specified. Any nonzero value is valid.
\item \texttt{X'0E'} The new wait time in seconds. Valid values are \texttt{X'00'} through \texttt{X'FF'} (0 - 255).
\item \texttt{X'0F'} The new retry count. Valid values are \texttt{X'00'} through \texttt{X'FF'} (0 - 255).
\end{itemize}

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:

\begin{itemize}
\item Use the SET PATCH command to set the wait/retry time dynamically
\item Modify the sample JCL (see “Sample JCL” on page 215) as follows:
\end{itemize}
Using CONVERTV on data sets with a revoked user ID in the RESOWNER field (OY59957)

DFSMSdss lets you perform CONVERTV operations on data sets that have a revoked user ID, in the RESOWNER field in the profile. The byte at offset X'11' in ADRPATCH can be used to give that resource owner authority to the SMS constructs.

To use ADRPATCH to set the revoked RESOWNER on, set the byte at offset X'11' in module ADRPATCH to a X'FF'. You can use the SET PATCH command to set the patch byte dynamically or modify the sample JCL (see "Sample JCL" on page 215) as follows to set the patch byte permanently:

```
//SYSIN DD *
NAME ADRSSU ADRPATCH
VER 11 00
REP 11 FF
```

Notes:
1. Catalog APAR OY56724 must be applied for this fix to work for VSAM data sets.
2. Normal DFSMSdss COPY and RESTORE processing is not changed to bypass MGMTCLAS and STORCLAS authorization checking. See "Bypassing storage and management class authorization checking during RESTORE (OY65348)" on page 220.

Restoring inconsistent PDSE data sets (OY60301)

If during a logical restore operation, DFSMSdss cannot tell if the data set is a PDS or a PDSE, it issues message ADR793E, indicating that the data set is an inconsistent PDSE. Setting the flag at offset X'12' in ADRPATCH tells DFSMSdss how to process these data sets during the restore. The settings are listed below:

**X'01'** DFSMSdss tries to restore the data set as a PDSE. Before the attempt, DFSMSdss issues message ADR794W to warn you that DFSMSdss assumes that the data set to be restored is a PDSE.

Notes:
1. If the data set is a PDS and you try to restore it as a PDSE, it will be unusable.
2. If you try to restore the data set to an unlike device, the restore fails and DFSMSdss issues message ADR792E, indicating to which device type the data set must be restored.
3. If you try to restore the data set to a preallocated target data set, the restore fails because DFSMSdss does not recognize the preallocated data set as usable.

**X'02'** DFSMSdss tries to restore the data set as a PDSE. 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.
DFSMDss Patch Area

Any setting other than X’01’ or X’02’

DFSMDss 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 215) as follows:

//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 12 00
REP 12 01

Changing default protection status during RESTORE (PN37489)

By default, DFSMDss maintains the protection status of the data set during a logical RESTORE. If the source data set was protected by RACF, a component of the Security Server for z/OS, at dump time, the target data set is RACF-indicated at restore time.

This function is affected by setting the flag at offset X’13’ in ADRPATCH. The settings are listed below:

X’00’

DFSMDss functions as previously described.

Any setting other than X’00’

DFSMDss 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 215) as follows:

//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 13 00
REP 13 FF

Restoring or copying undefined, multivolume SMS-managed data sets (OY63818)

Messages ADR709E and IGD17040I might be received when copying or restoring the following types of non-VSAM, multivolume data sets:

• Empty, multivolume data sets dumped with the ALLEXCP keyword
• Undefined data sets
• TTR-organized BDAM data sets
• Data sets with BLKSIZE=0

This function is affected by setting the flag at offset X’14’ in ADRPATCH. The settings are listed below:

X’00’

DFSMDss 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.
DFSMSdss Patch Area

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 215) as follows:

```
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 14 00
  REP 14 FF
```

Bypassing backup-while-open processing (OY63531)

When DFSMSdss encounters a data set marked as backup-while-open (BWO) eligible, DFSMSdss processes it accordingly. If the BWO indication is on erroneously, the BWO processing may be bypassed for a logical data set dump.

This function is affected by setting the flag at offset X'15' in ADRPATCH. The settings are listed below:

- X'00'  BWO processing is performed.
- Any setting other than X'00'  BWO processing is bypassed.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 15 00
  REP 15 FF
```

Bypassing storage and management class authorization checking during RESTORE (OY65348)

For a logical RESTORE and a physical data set restore, storage class and management class authorization checking is normally conducted. This checking can be bypassed to let the storage administrator restore data sets that are protected by data set profiles whose RESOWNER field contains a user ID that has been revoked.

This function is affected by setting the flag at offset X'16' in ADRPATCH. The settings are listed below:

- X'00'  DFSMSdss functions normally and conducts authorization checking.
- Any setting other than X'00'  Authorization checking is bypassed.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 16 00
  REP 16 FF
```
Issuing notification for tape and migrated data sets (OY66092)

On a logical data set copy or dump, a data set can be specified with a partially qualified name. If that name resolves to a data set cataloged to tape or to a migrated data set, DFSMSdss does not issue a warning message that the data set was not selected.

This function is affected by setting the flag at offset X'17' in ADRPATCH. The settings are listed below:

- X'00' DFSMSdss functions normally and issues no warning message.
- Any setting other than X'00' DFSMSdss issues a warning message.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```assembler
//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 17 00
REP 17 FF
```

Using RESET with concurrent copy (OY65555)

The RESET keyword specifies that DFSMSdss resets the data-set-changed indicator for data sets after the dump is complete. By default, the RESET keyword is ignored when the CONCURRENT keyword is also specified with the DUMP command.

Setting the byte at offset X'18' in ADRPATCH to a nonzero value causes DFSMSdss to perform RESET processing when both the RESET and CONCURRENT keywords are specified with the DUMP command. In this case, the data-set-changed indicator is reset after concurrent copy initialization is complete and before the data set is written to the output.

To dynamically enable RESET processing with concurrent copy, use the SET PATCH command. To enable RESET processing with concurrent copy permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```assembler
//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 18 00
REP 18 FF
```

After the data-set-changed indicator has been reset, if the concurrent copy dump is unsuccessful for any reason, the data-set-changed indicator for the data set remains reset even though you might not have a usable dump of the data. For this reason, you must carefully evaluate the risks of using this patch to enable RESET processing with concurrent copy. Rather than using this patch to accomplish incremental backup in DFSMSdss with concurrent copy, use DFSMShsm instead.

In the event that the data-set-changed indicators have been reset and the dump subsequently fails, your recovery action can be either of the following:

- Turn the data-set-changed indicators back on by using AMASPZAP.
- Dump the data again. Because the data-set-changed indicators are still off, you must not specify “BY(DSCHA,EQ,YES)” in the dump command.
Forcing RESTORE after message ADR482E (OY67532)

If message ADR727E was received while DFSMSdss was dumping an SMS-managed user catalog with aliases, the dump tape might be unusable and message ADR482E might be received during RESTORE. If you set the flag at offset X'19' in ADRPATCH on, DFSMSdss forces the restore to continue after receiving message ADR482E.

Note: If you use the AMASPZAP program to set the patch permanently, turn off this patch after restoring the problem dump tape, so that legitimate message ADR482E condition are detected.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```plaintext
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 19 00
  REP 19 FF
```

Restoring VSAM KSDS or VRRDS after messages ADR789W, ADR364W, and ADR417W (OY67942)

Prior to the fix of APAR OY67724, DFSMSdss sometimes (when using the CONCURRENT and VALIDATE options) created dumps with empty track records for imbedded sequence set tracks and with zero in the total record count dumped for the data set. The sequence set records are not necessary when restoring data sets dumped with VALIDATE. Without patch byte X'1A' on the following messages are generated:
- ADR364W and ADR417W (because of the empty tracks)
- ADR789W (because of the missing record count)

and the restore fails.

If message ADR789W, or message ADR364W and message ADR417W, or all three messages are received during a logical restore of a KSDS or VRRDS from a dump created while using CONCURRENT and VALIDATE either by keyword or by default, you can retrieve your data sets by setting the flag at offset X'1A' in ADRPATCH and rerunning the restore job. The settings are listed below:

- X'00'
  DFSMSdss functions normally. It fails to restore data sets that have errors in the dump.

- Any setting other than X'00'
  During a logical RESTORE of a KSDS or VRRDS, if DFSMSdss detects an empty record in the dump or a missing count of the total number of records dumped for the data set, or both, it restores all of the data for the data set in the dump and keeps the resulting data set. You still get message ADR789W with a valid output record count. You can compare this valid output record count to the record count from message ADR788I from the dump job to ensure DFSMSdss has retrieved all the data.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:
Restoring VSAM data sets with expiration date of 1999365 (OW00780)

Prior to the fix of APAR OW00780, DFSMSdss sometimes mishandled VSAM expiration dates during logical data set DUMP and RESTORE operations. Non-SMS VSAM data sets that had an expiration date of 1999365 were dumped and restored with no expiration date (an expiration date of 0000000).

The byte at offset X'1B' of module ADRPATCH has been defined to allow special processing for VSAM data sets with an expiration date of 1999365 that were dumped without the fix for OW00780. The possible settings for the flag byte are listed below:

X'00' Non-SMS VSAM data sets that had an expiration date of 1999365 and that were dumped without the fix for OW00780 are restored with no expiration date.

X'FF' Any non-SMS VSAM data set that had an expiration date of 1999365 and that was dumped without the fix for OW00780 are restored with an expiration date of 1999365. However, non-SMS VSAM data sets that originally had no expiration date are also restored with an expiration date of 1999365.

Note: This flag byte should only be used when restoring VSAM data sets dumped without the APAR OW00780 fix and with an expiration date of 1999365.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```
//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 1A 00
REP 1A FF
```

Restoring VSAM data sets with expiration dates beyond 2000 (OW00780)

Prior to the fix of APAR OW00780, DFSMSdss sometimes mishandled VSAM expiration dates during logical data set DUMP and RESTORE operations. Data sets that had expiration dates in the year 2000 or beyond were dumped and restored as though they had expired in the 1900s.

The byte of offset X'1C' of module ADRPATCH has been defined to allow special processing for VSAM data sets with expiration dates in the year 2000 or beyond that were dumped without the fix of OW00780. The possible settings for the flag byte are listed below:

X'00' Any VSAM data sets that expired in the years 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 in the year 2000 or beyond.
DFSMSdss Patch Area

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 215) as follows:

```
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 1C 00
  REP 1C FF
```

Changing default insertion of EOF track during COPY with ALLDATA specified (OW15003)

When multivolume sequential data sets are copied to a single-volume like device and ALLDATA is specified, the default action inserts an explicit EOF track in the target data set, when required. A patch is provided that allows an installation to change this default. If the patch byte at offset X'1D' of module ADRPATCH is set to X'FF', no explicit EOF track is added to the target data set.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 1D 00
  REP 1D FF
```

Using RESET or UNCATALOG in a logical data set dump (PN60114)

For a logical data set dump operation, use of the RESET or UNCATALOG keyword causes the enqueue on a data set to be held until all data sets are dumped. DFSMSdss does not reset the data-set-changed indicator or uncatalog the data set until after all data sets are dumped.

This function is affected by setting the flag at offset X'1E' in ADRPATCH. The settings are listed below:

- X'00': DFSMSdss functions normally as described above.
- Any setting other than X'00': DFSMSdss resets the data-set-changed indicator or uncatalog the data set when the data set is dumped. The enqueue on a data set is released after DFSMSdss has completed resetting the data-set-changed indicator or uncataloging the data set.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:
Changing secondary allocation quantity in format 1 DSCB for PDSE data sets (OW07755)

PTFs UW06768 and UW06769 for APAR OW04199 caused the target PDSE secondary allocation quantity (DS1SCAL3) in the Format 1 DSCB to increase or decrease by a ratio of "blocks per track of blocksize" divided by "blocks per track of 4096" during data set COPY and RESTORE processing. As part of the fix to this problem, three pairs of 4-byte fields at offset X'20' through X'37' in ADRPATCH are provided for installations to change the incorrect secondary allocation quantity of an unpreallocated target PDSE during COPY and RESTORE processing. The secondary allocation quantity of the source PDSE is not affected by the patch.

Note: Use this patch to change the incorrect secondary allocation quantity resulting from PTFs UW06768 and UW06769. Set the high threshold value with caution to avoid changing the secondary allocation quantity of PDSE data sets that are not affected by PTFs UW06768 and UW06769.

To use this function when the source PDSE is allocated in cylinders, you must modify the 4-byte field at each of the following offsets:

X'20' A nonzero, 4-byte value indicates a threshold in cylinders. DFSMSdss compares the secondary allocation quantity of the source PDSE to this threshold. If the secondary allocation quantity is greater than this threshold, the value set at offset X'24' is used as the secondary allocation quantity in the calculation for the target PDSE.

A zero, 4-byte value indicates this function is not activated.

X'24' This value is used as the secondary allocation quantity in cylinders, if applicable.

To use this function when the source PDSE is allocated in tracks, you must modify the 4-byte field at each of the following offsets:

X'28' A nonzero, 4-byte value indicates a threshold in tracks. DFSMSdss compares the secondary allocation quantity of the source PDSE to this threshold. If the secondary allocation quantity is greater than this threshold, the value set at offset X'2C' is used as the secondary allocation quantity in the calculation for the target PDSE.

A zero, 4-byte value indicates this function is not activated.

X'2C' This value is used as the secondary allocation quantity in tracks, if applicable.

To use this function when the source PDSE is allocated in blocks, you must modify the 4-byte field at each of the following offsets:

X'30' A nonzero, 4-byte value indicates a threshold in blocks. DFSMSdss compares the secondary allocation quantity of the source PDSE to this threshold. If the secondary allocation quantity is greater than this threshold, the value set at offset X'34' is used as the secondary allocation quantity in the calculation for the target PDSE.

A zero, 4-byte value indicates this function is not activated.
DFSMSdss Patch Area

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 215) to cause DFSMSdss to do the following:

- Use 250 cylinders as the secondary allocation quantity when the source PDSE is allocated in cylinders and has a secondary allocation quantity greater than 5000 cylinders.
- Use 2500 tracks as the secondary allocation quantity when the source PDSE is allocated in tracks and has a secondary allocation quantity greater than 50 000 tracks.
- Use 12 500 blocks as the secondary allocation quantity when the source PDSE is allocated in blocks and has a secondary allocation quantity greater than 250 000 blocks.

//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 20 00000000 /* Verify value is 0 */
REP 20 00001388 /* If > 5000 cylinders */
VER 24 00000000 /* Verify value is 0 */
REP 24 000000FA /* Then use 250 cyls */
VER 28 00000000 /* Verify value is 0 */
REP 28 0000C350 /* If > 50000 tracks */
VER 2C 00000000 /* Verify value is 0 */
REP 2C 000000FA /* Then use 2500 tracks */
VER 30 00000000 /* Verify value is 0 */
REP 30 00030090 /* If > 250000 blocks */
VER 34 00000000 /* Verify value is 0 */
REP 34 000030D4 /* Then use 12500 blks */

NOTE: As an alternative, you can use the SET PATCH command to set the patch bytes dynamically.

Changing reference date default settings during data set COPY and RESTORE processing (OW12011)

During RESTORE and COPY operations, source and target data set reference dates remain the same, unless the data set is renamed. However, if you rename the restored or copied data set, the current date (TODAY) replaces the reference date in the target data set's DS1REFD field.

DFSMSdss provides four patch bytes that allow you to change the method of setting the DS1REFD field. Below are the patch bytes:

- If the patch byte at offset X’38’ of module ADRPATCH is set to X’FF’, the reference date of the target data set that is restored without a rename is set to the current date.
- If the patch byte at offset X’39’ of module ADRPATCH is set to X’FF’, the reference date of the target data set that is restored and renamed is set to the source data set's reference date.
- If the patch byte at offset X’3A’ of module ADRPATCH is set to X’FF’, the reference date of the target data set that is copied without a rename is set to the current date.
- If the patch byte at offset X’3B’ of module ADRPATCH is set to X’FF’, the reference date of the target data set that is copied and renamed is set to the source data set's reference date.
DFSMSdss Patch Area

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 215) as follows:

```plaintext
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 38 00
  REP 38 FF
  VER 3A 00
  REP 3A FF
```

**NOTE:** As an alternative, you can use the SET PATCH command to set the patch bytes dynamically.

### Changing default protection processing during COPY (OW10314)

By default, DFSMSdss issues message ADR757E during COPY with DELETE specified if the data set is RACF-indicated, but a discrete profile is not associated with the data set.

This function is affected by setting the flag at offset X'3C' in ADRPATCH. The settings are listed below:

- **X'00'** DFSMSdss functions normally as described above.
- **Any setting other than X'00'** DFSMSdss does not fail the COPY operation with message ADR757E, but allows the data set to be processed and issues either message ADR759W or ADR771W, depending on the SAF return codes.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```plaintext
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 3C 00
  REP 3C FF
```

### Bypassing management and storage class access checks during COPY (PN72592)

A COPY operation might not be successful when the owner of the data set does not have read access to the STORCLAS or MGMTCLAS routines that allocate the target data set, even though the ADMINISTRATOR keyword is specified. The ADMINISTRATOR keyword gives access to the data set, not to the automatic class selection (ACS) routines that are used for STORCLAS or MGMTCLAS processing.

This function is affected by setting the flag at offset X'3D' in ADRPATCH. The settings are listed below:

- **X'00'** DFSMSdss functions normally as described above.
- **Any setting other than X'00'** When the ADMINISTRATOR keyword is specified, DFSMSdss causes SMS to bypass access authorization checking to the ACS routines for STORCLAS and MGMTCLAS when allocating the target data set during a COPY operation.
DFSMSdss Patch Area

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 215) as follows:

```plaintext
//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 3D 00
REP 3D FF
```

Changing default handling of invalid tracks created during data set COPY and RESTORE processing (OW08174)

When invalid tracks are created during COPY and RESTORE processing, message ADR367E is issued and the invalid tracks are erased from the target volume, regardless of whether CANCELERROR is specified. The copy or restore of the data set ends and the target data set is deleted. The COPY or RESTORE operation continues with the next data set.

A patch byte is provided that allows you to change the default handling of invalid tracks created during COPY and RESTORE processing. If the patch byte at offset X'3E' of module ADRPATCH is set to X'FF', CANCELERROR specification will determine the action taken. When CANCELERROR is specified, the action is to issue message ADR367E and erase the invalid track from the target volume. The restore or copy of the data set receiving the error is terminated and the target data set is deleted. The RESTORE or COPY processing continues with the next data set.

When CANCELERROR is not specified, the action is to issue message ADR366W and leave the invalid track on the volume. The RESTORE or COPY operation continues.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```plaintext
//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 3E 00
REP 3E FF
```

Forcing RESTORE to the same volumes as the source VSAM data set (OW07077)

If a user does not specify output volumes, you can force the RESTORE operation to attempt to return a VSAM data set assigned to an SMS storage class with guaranteed space to the same set of primary volumes that the source data set had occupied when dumped.

This function is affected by setting the flag at offset X'3F' in ADRPATCH. The settings are listed below:

- **X'00'**: Unless the user specifies output volumes, RESTORE processing does not pass the source volume list to SMS, and the restored data set probably does not occupy the same primary volumes that the source data set occupied when it was dumped. Because this allows the use of any volumes in the storage group, there is less chance of a failure due to lack of space on volumes.

- **Any setting other than X'00'**: Unless the user specifies output volumes,
RESTORE processing passes the source volume list to SMS, forcing the restored data set to occupy the same primary volumes that the source data set had occupied when it was dumped, provided that it is assigned to an SMS storage class with guaranteed space. If it will not fit, RESTORE fails processing.

Notes:
1. DFSMSdss cannot ensure that the order of volumes are maintained during a restore of a multivolume data set. SMS determines volume order.
2. This patch does not support data sets with candidate space volumes. The restore of such a data set will fail with an ADR709E message.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```plaintext
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 3F 00
  REP 3F FF
```

### Modifying number of volumes allocated for SMS data sets during logical RESTORE and COPY (OW15880)

When processing SMS data sets, the total number of volumes DFSMSdss allocates for the target data set is the same as the total number of volumes that were allocated for the source data set, unless the source data set was multivolume and output volumes are specified. When the source data set was multivolume and a list of output volumes is supplied with OUTDDNAME or OUTDYNAM, the number of volumes DFSMSdss allocates is the number of volumes in the volume output list.

The number of output volumes allocated for SMS data sets can be modified by setting an enabling flag at offset X'40' in ADRPATCH and a count value at offset X'41' in ADRPATCH. After the enabling flag is set, the following are true:

- When an output volume list is specified and the patch count value is not zero, the allocated number of volumes is the lesser of either the number of volumes in the output list or the patch count value.
- When an output volume list is specified and the patch count value is zero, the allocated number of volumes is the lesser of either the number of volumes in the output list, or the allocated number of volumes that were for the source data set.
- When no output volume list is specified, the allocated number of volumes is the greater of either the number of volumes that were allocated for the source data set, or the patch count value.

The patch bytes are defined as follows:

- Offset X'40'
  - X'00' Use current DFSMSdss SMS allocation rules
  - X'20' Modify DFSMSdss SMS allocation rules
- Offset X'41'
  - The hexadecimal representation of the number of volumes to be used for SMS data set allocation. Any value between 0 and 59 (X'00' through X'3B') can be specified. If a value larger than 59 is specified, the value of 59 (X'3B') is used.
Notes:

1. For any keyed VSAM data set, there must be sufficient space for a primary extent on each volume that is to contain data. No space is allocated on candidate volumes.

2. For any guaranteed space keyed VSAM data set, there also must be sufficient volumes in the target storage group to provide volume serial numbers for each primary and candidate volume.

3. This patch does not modify the number of volumes allocated for data sets whose DSORG is PO (for example, PDS, PDSE and HFS), and does not modify the number of volumes allocated for VSAM linear data sets.

4. When this patch is activated, the MAKEMULTI keyword is ignored.

To cause DFSMSdss to use ten volumes for SMS data set allocation, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```
//SYSIN DD *
   NAME ADRDSSU ADRPATCH
   VER 40 00
   REP 40 20
   VER 41 00
   REP 41 0A
```

NOTE: As an alternative, you can use the SET PATCH command to set the patch bytes dynamically.

---

Dumping a keyed VSAM data set that has data CAs without corresponding index CIs (OW17877)

When a KSDS is logically dumped with DFSMSdss using the VALIDATE option, a check is performed to determine if there are data control areas (CAs) without corresponding index control intervals (CIs). If there are missing index CIs, ADR970E is issued and the dump of this data set fails.

This function is affected by setting the flag at offset X'42' in ADRPATCH. The settings are listed below:

- **X'00'**: DFSMSdss functions normally as described above.
- **X'01'**: DFSMSdss issues ADR985W instead of ADR970E when a possible missing index CI condition is detected during logical data set DUMP using the VALIDATE option.
- **Any setting other 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.
DFSMSdss Patch Area

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 215) as follows:

```plaintext
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  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 ADRPATCH. The settings are listed below:

- **X'00'**: DFSMSdss functions normally as described above.
- **Any setting other than X'00'**: DEFRAG moves selected extents, even when the data set DS1CPOIT flag is set on. Message ADR252I is issued when the patch byte is set on to indicate that the installation is overriding normal DEFRAG processing.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```plaintext
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 43 00
  REP 43 FF
```

### Setting the percentage to overallocate target data set space (OW27837)

When restoring an extended-format non-VSAM data set to a device which does not support extended-format, the data set is converted to a nonextended-format. During this type of conversion, DFSMSdss is not able to calculate the exact amount of space that the target data set will require. Therefore, it is possible for the target data set to be underallocated, causing RESTORE to fail with message ADR910E, return code 40000004, reason code 0000000D. To avoid underallocating the target data set, a new patch byte has been created. The hex value set in this patch byte is the percentage to overallocate by. This function is affected by setting the flag at offset X'44' in ADRPATCH.

To use ADRPATCH to overallocate the target data set by a decimal 10%, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```plaintext
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 44 00
  REP 44 0A
```
DFSMSdss Patch Area

Notes:
1. As an alternative, you can use the SET PATCH command to set the patch byte dynamically.
2. Set this patch only for those data sets that have failed to convert. Reset the patch to zero after the failing data set is restored, so that other converted data sets are not needlessly overallocated. Also, the required percentage of overallocated space can vary by data set. In other words, some data sets might require an overallocation of 10%, while some require additional space. Adjust the overallocation percentage, as needed.

Bypassing RLS processing (OW32817)

For SMS-managed VSAM data sets, DFSMSdss performs serialization that provides data integrity during logical dump and copy in both the RLS and the non-RLS environments. This serialization includes communications with VSAM RLS to determine the type of enqueues to be performed, based on the type of access that the data set is currently being used for by other jobs.

You can bypass the communication between DFSMSdss and VSAM RLS, which results in DFSMSdss performing non-RLS serialization, regardless of how the data set is currently being accessed. Use of this function can compromise data integrity when the data set is being accessed through RLS by some other job while the dump or copy is being performed. Use this function only if you are willing to tolerate the exposure of not having data integrity in order to force the successful completion of the operation or to prevent updates to the data set from failing during the operation.

When the communication between DFSMSdss and VSAM RLS is bypassed, communication with CICS® is also bypassed. Therefore, use this function should only in an environment where forward recovery logging and forward recovery is managed entirely by the application.

This function is affected by setting the flag at offset X'45' in ADRPATCH. The settings are listed below:

X'00'
DFSMSdss functions normally. The communications with VSAM RLS are performed and DFSMSdss provides RLS or non-RLS serialization accordingly.

Any setting other than X'00'
DFSMSdss bypasses RLS processing. The communications with VSAM RLS are not performed and DFSMSdss performs non-RLS serialization.

To set the flag to on dynamically, use the SET PATCH command. To set the flag to on permanently, modify the sample JCL (see “Sample JCL” on page 215) as follows:

```bash
//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 45 00
REP 45 FF
```
Changing creation date default settings during logical data set COPY and RESTORE (OW19618)

During RESTORE and COPY operations, source and target data set creation dates remain the same, unless the data set is renamed. However, if you rename the restored or copied data set, the current date (TODAY) replaces the creation date in the target data set’s DS1CREDT field.

DFSMSdss provides four patch bytes that allow you to change the method of setting the DS1CREDT field. These patch bytes apply only to data sets that are allocated by DFSMSdss (not 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 215) as follows:

```
//SYSIN DD *
  NAME ADRDSSU ADRPATCH
  VER 46 00
  REP 46 FF
  VER 48 00
  REP 48 FF
```

Notes:
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.
DFSMSdss Patch Area

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 215) as shown below:

```plaintext
//SYSIN DD *
NAME ADRDSSU ADRPATCH
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 ADRPATCH. 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'44C' 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 215) as follows:

```plaintext
//SYSIN DD *
NAME ADRDSSU ADRPATCH
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
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 ADRPATCH. 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 215) as follows:

```sql
//SYSIN DD *
NAME ADRDSSU ADRPATCH
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.

For more information about FCWITHDRAW and volume initialization processing, see “FCWITHDRAW” on page 405.

---

**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
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 ADRPATCH. 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 215) as follows:

```
//SYSIN DD *
NAME ADRDSSU ADRPATCH
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 215) as follows:
DFSMSdss Patch Area

//SYSIN DD *
NAME ADRDSSU 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 (OA23805)

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.

\[ \text{Any setting other than } X'00' \]

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 215) 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 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 ADRPATCH. 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 215) as follows:

```
//SYSIN DD *
NAME ADRDSSU ADRPATCH
VER 59 00
REP 59 FF
```

ADRPTCHB data area

Table 20 lists the mapping of flags in ADRPTCHB.

### Table 20. ADRPTCHB Mapping Macro

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0)</td>
<td>0</td>
<td>STRUCTURE</td>
<td>4096</td>
<td>ADRPTCHB</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>0</td>
<td>CHARACTER</td>
<td>8</td>
<td>PTCHKEY</td>
<td>USE PREALLOCATED VSAM</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>8</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PTCHARRY</td>
<td>USE PREALLOCATED VSAM</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>8</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBUVSPRE</td>
<td>USE PREALLOCATED VSAM</td>
</tr>
<tr>
<td>9</td>
<td>(9)</td>
<td>9</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PDUPKEY</td>
<td>DUPLICATE VSAM KEY</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>10</td>
<td>UNSIGNED</td>
<td>2</td>
<td>PBTIMEOUT</td>
<td>TIMEOUT CONSTANT</td>
</tr>
<tr>
<td>12</td>
<td>(C)</td>
<td>12</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBADINFO</td>
<td>VARIABLE USED FOR ADTL INFO XT@NA03602 IN ADRRTDSC &amp; IN ADRFDSRL</td>
</tr>
<tr>
<td>13</td>
<td>(D)</td>
<td>13</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBWAITFG</td>
<td>WAIT/RETRY FLAG FOR VTOC/VVDS ENQ OR RESERVE</td>
</tr>
<tr>
<td>14</td>
<td>(E)</td>
<td>14</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBWAIT#</td>
<td>WAIT TIME FOR VTOC/VVDS ENQ OR RESERVE</td>
</tr>
<tr>
<td>15</td>
<td>(F)</td>
<td>15</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBRETRY#</td>
<td>RETRY COUNT FOR VTOC/VVDS ENQ OR RESERVE</td>
</tr>
<tr>
<td>16</td>
<td>(10)</td>
<td>16</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBRESERV</td>
<td>RESERVE BYTE TO ACCOUNT FOR OFFSET ERRORS</td>
</tr>
<tr>
<td>17</td>
<td>(11)</td>
<td>17</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBREVoke</td>
<td>GIVE REVOKED DS OWNER ACCESS TO SMS CONSTRUCTS</td>
</tr>
<tr>
<td>18</td>
<td>(12)</td>
<td>18</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBBPDSE</td>
<td>BROKEN PDSE RESTORE FLAG</td>
</tr>
<tr>
<td>19</td>
<td>(13)</td>
<td>19</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBNRACFI</td>
<td>NO INDIC. FOR LOG. REST.</td>
</tr>
<tr>
<td>20</td>
<td>(14)</td>
<td>20</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBNMVNCV</td>
<td>NO MV ALLOC IF ALSO UNDEFINED DATA SET</td>
</tr>
<tr>
<td>21</td>
<td>(15)</td>
<td>21</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBYPBPWO</td>
<td>BYPASS BWO</td>
</tr>
<tr>
<td>22</td>
<td>(16)</td>
<td>22</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBNACSAU</td>
<td>NO ACS AUTH. CHECKING DURING RESTORE</td>
</tr>
<tr>
<td>23</td>
<td>(17)</td>
<td>23</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBSLECT</td>
<td>WARNING MESSAGE FOR DATA SETS NOT SELECTED.</td>
</tr>
<tr>
<td>24</td>
<td>(18)</td>
<td>24</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBCCRESE</td>
<td>DO RESET WITH CONCURRENT COPY</td>
</tr>
<tr>
<td>25</td>
<td>(19)</td>
<td>25</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBNO482E</td>
<td>RESTORE DS FROM DUMP TAPE WHICH HAD MSGADR727E</td>
</tr>
<tr>
<td>26</td>
<td>(1A)</td>
<td>26</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBMISCNT</td>
<td>RESTORE DS FROM DUMP TAPE DUMPED WITH CC PRIOR TO INSTALLATION OF FIX FOR OY67724</td>
</tr>
<tr>
<td>27</td>
<td>(1B)</td>
<td>27</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBX9365</td>
<td>TREAT ALL F1 DSCB EXP DATE 1999.365 FROM DUMP AS ACTUAL EXP DATE WHEN DEFINING TARGET</td>
</tr>
<tr>
<td>28</td>
<td>(1C)</td>
<td>28</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBEX2000</td>
<td>TREAT ALL F1 DSCB SMALL EXP DATES FROM DUMP AS 2NNN EXP DATES WHEN DEFINING TARGET</td>
</tr>
<tr>
<td>29</td>
<td>(1D)</td>
<td>29</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBEOFNO</td>
<td>DO NOT INSERT EOF DURING COPY ALLDATA FROM MULTIVOLUME TO SINGLE-VOLUME DATASET</td>
</tr>
<tr>
<td>30</td>
<td>(1E)</td>
<td>30</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBBYLENY</td>
<td>BYPASS PN27748 CODE WHICH DELAYS RESET UNCAT AND HOLDS LONGER ENQUEUES</td>
</tr>
<tr>
<td>31</td>
<td>(1F)</td>
<td>31</td>
<td>UNSIGNED</td>
<td>1</td>
<td>*</td>
<td>RESERVED BYTE</td>
</tr>
</tbody>
</table>
### Table 20. ADRPTCHB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>(20)</td>
<td>20</td>
<td>UNSIGNED</td>
<td>4</td>
<td>PBCYLHI</td>
<td>HI THRESHOLD FOR CHECKING DS1SCAL3 IN CYL ALLOCATION</td>
</tr>
<tr>
<td>36</td>
<td>(24)</td>
<td>24</td>
<td>UNSIGNED</td>
<td>4</td>
<td>PBCYLQTY</td>
<td>CHANGE DS1SCAL3 TO THIS VALUE IF IT IS &gt; PBCYLHI</td>
</tr>
<tr>
<td>40</td>
<td>(28)</td>
<td>28</td>
<td>UNSIGNED</td>
<td>4</td>
<td>PBTRKHI</td>
<td>HI THRESHOLD FOR CHECKING DS1SCAL3 IN TRK ALLOCATION</td>
</tr>
<tr>
<td>44</td>
<td>(2C)</td>
<td>2C</td>
<td>UNSIGNED</td>
<td>4</td>
<td>PBTRKQTY</td>
<td>CHANGE DS1SCAL3 TO THIS VALUE IF IT IS &gt; PBTRKHI</td>
</tr>
<tr>
<td>48</td>
<td>(30)</td>
<td>30</td>
<td>UNSIGNED</td>
<td>4</td>
<td>PBBLKHI</td>
<td>HI THRESHOLD FOR CHECKING DS1SCAL3 IN BLK ALLOCATION</td>
</tr>
<tr>
<td>52</td>
<td>(34)</td>
<td>34</td>
<td>UNSIGNED</td>
<td>4</td>
<td>PBBLKQTY</td>
<td>CHANGE DS1SCAL3 TO THIS VALUE IF IT IS &gt; PBBLKHI</td>
</tr>
<tr>
<td>56</td>
<td>(38)</td>
<td>38</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBROREFD</td>
<td>DS1REFD RESTORE OLD 0 = USE OLD DATE 1 = USE TODAY'S DATE</td>
</tr>
<tr>
<td>57</td>
<td>(39)</td>
<td>39</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBRNREFD</td>
<td>DS1REFD RESTORE NEW 1 = USE OLD DATE 0 = USE TODAY'S DATE</td>
</tr>
<tr>
<td>58</td>
<td>(3A)</td>
<td>3A</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBCOREFD</td>
<td>DS1REFD COPY OLD 0 = USE OLD DATE 1 = USE TODAY'S DATE</td>
</tr>
<tr>
<td>59</td>
<td>(3B)</td>
<td>3B</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBCNREFD</td>
<td>DS1REFD COPY NEW 1 = USE OLD DATE 0 = USE TODAY'S DATE</td>
</tr>
<tr>
<td>60</td>
<td>(3C)</td>
<td>3C</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBCDELNP</td>
<td>DEGRADE 'E' MSG TO 'W' IF COPY DEL NO TGT PROFILE</td>
</tr>
<tr>
<td>61</td>
<td>(3D)</td>
<td>3D</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBNACSAC</td>
<td>NO ACS AUTHORIZATION CHECKING DURING COPY</td>
</tr>
<tr>
<td>62</td>
<td>(3E)</td>
<td>3E</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBINVTOK</td>
<td>DON'T ERASE INVALID TRACK DURING COPY/RESTORE</td>
</tr>
<tr>
<td>63</td>
<td>(3F)</td>
<td>3F</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBSRCVOL</td>
<td>PASS SOURCE PRIMARY VOLUMES FROM WHICH DS WAS DUMPED</td>
</tr>
<tr>
<td>64</td>
<td>(40)</td>
<td>40</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBVLOPT</td>
<td>'PATCHABLE' VOLCOUNT OPTION FLAG DATA SET</td>
</tr>
<tr>
<td>65</td>
<td>(41)</td>
<td>41</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBVCVAL</td>
<td>NUMBER OF VOLUMES TO USE FOR OUTPUT DATA SET</td>
</tr>
<tr>
<td>66</td>
<td>(42)</td>
<td>42</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBMSCIOK</td>
<td>OK TO HAVE MISSING INDEX CI IN KEYED VSAM DATA SET - VALIDATE</td>
</tr>
<tr>
<td>67</td>
<td>(43)</td>
<td>43</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBRESV60</td>
<td>WAS FOR DEFRAG TO MOVE CHECKPOINT INDICATED DATA SET</td>
</tr>
<tr>
<td>68</td>
<td>(44)</td>
<td>44</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBFUDGE</td>
<td>OVER ALLOCATE DATA SET BY % IN FUDGE FACTOR</td>
</tr>
<tr>
<td>69</td>
<td>(45)</td>
<td>45</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBNORLS</td>
<td>DO NOT DO RLS QUIESCE</td>
</tr>
<tr>
<td>70</td>
<td>(46)</td>
<td>46</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBroCRED</td>
<td>DS1CRED RESTORE OLD 0 = USE OLD DATE 1 = USE TODAY'S DATE</td>
</tr>
<tr>
<td>71</td>
<td>(47)</td>
<td>47</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBRCRED</td>
<td>DS1CRED RESTORE NEW 1 = USE OLD DATE 0 = USE TODAY'S DATE</td>
</tr>
<tr>
<td>72</td>
<td>(48)</td>
<td>48</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBCORED</td>
<td>DS1CRED COPY OLD 0 = USE OLD DATE 1 = USE TODAY'S DATE</td>
</tr>
<tr>
<td>73</td>
<td>(49)</td>
<td>49</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBNCRED</td>
<td>DS1CRED COPY NEW 1 = USE OLD DATE 0 = USE TODAY'S DATE</td>
</tr>
<tr>
<td>74</td>
<td>(4A)</td>
<td>4A</td>
<td>UNSIGNED</td>
<td>1</td>
<td>*</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>

*Note:* The following 6 fields starting at offset X'20' are intended for installations to “adjust” target PDSE DS1SCAL3 which may have been incorrectly altered by PE-OW04199. The logic is: IF 2nd_qty > PBCYLHI THEN 2nd_qty = PBCYLQTY. Any non-zero value in PBCYLHI, PBTRKHI, or PBBLKHI will activate the “adjustment code” for the corresponding allocation type in ADRNEWDS.
### DFSMSdss Patch Area

Table 20. ADRPTCHB Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>75 (4B)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBVDPDSE</td>
<td>VALIDATE PDSE 1 = ENABLE VALIDATION 0 = DISABLE VALIDATION</td>
</tr>
<tr>
<td>76 (4C)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBMXPTSK</td>
<td>MAXIMUM NUMBER OF ACTIVE PARALLEL SUBTASKS X’01’ THROUGH X’FF’ = USER SPECIFIED THRESHOLD VALUE X’00’ = DFSMSdss DETERMINED THRESHOLD</td>
</tr>
<tr>
<td>77 (4D)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBNOINIT</td>
<td>CHANGE DEFAULT INIT PROCESSING DURING DUMP FCWITHDRAW 1 = DO NOT INIT VOLUME 0 = INIT DUMP SOURCE VOLUME WHEN APPLICABLE</td>
</tr>
<tr>
<td>78 (4E)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBMOVLL</td>
<td>OVERRIDE DEFRAg TO MOVE LINKLIST-INDICATED DATA SET</td>
</tr>
<tr>
<td>79 (4F)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBFCDEF</td>
<td>OVERRIDE FAST REPLICATION DEFAULT SETTING TO NONE</td>
</tr>
<tr>
<td>80 (50)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBDCRTBD</td>
<td>REBUILD COMPRESSION DICTIONARY AFTER THIS MANY BAD COMPRESSIONS</td>
</tr>
<tr>
<td>81 (51)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBCMPTH</td>
<td>POOR COMPRESSION THRESHOLD VALUE 1-100</td>
</tr>
<tr>
<td>82 (52)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBDSCHA</td>
<td>OVERRIDE RESET OF DSCHA BIT ON FULL OR PART RESTORE 0 = DS1DSCHA RESET 1 = DSCHA NOT RESET</td>
</tr>
<tr>
<td>83 (53)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBCHKVVR</td>
<td>DOUBLE CHECK THE DSHURBA VALUE FOR LDS DATA SET</td>
</tr>
<tr>
<td>84 (54)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBCSIE5</td>
<td>ENABLE USAGE OF THE CATALOG SEARCH INTERFACE FOR GENERIC FILTER</td>
</tr>
<tr>
<td>85 (55)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBCPVP01</td>
<td>1 = TREAT RESTORE AS A CPVOL - ONLY BY THE SET PATCH CMD</td>
</tr>
<tr>
<td>86 (56)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBRESDS</td>
<td>TREAT ESDS AS LDS WHEN ATTEMPTING LOGICAL RESTORE OF INVALID DB2 ESDS</td>
</tr>
<tr>
<td>87 (57)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>PBMSGTIME</td>
<td>ADD TIMESTAMPS TO MESSAGES IN EACH OF THE VARIOUS CLASSES</td>
</tr>
</tbody>
</table>

### ADRPTCHB cross-reference

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRPTCHB</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PBADINFO</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>PBBLKHI</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>PBBLKQTY</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>PBPOSE</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>PBBYLENQ</td>
<td>1E</td>
<td></td>
</tr>
<tr>
<td>PBByPBPWc</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>PBCRESE</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>PBCDELNP</td>
<td>3C</td>
<td></td>
</tr>
<tr>
<td>PBCMPTH</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>PBCNCREED</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>PBCNREFD</td>
<td>3B</td>
<td></td>
</tr>
<tr>
<td>PBCOCREED</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>PBCOREFD</td>
<td>3A</td>
<td></td>
</tr>
<tr>
<td>PBCYLHI</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>PBCYLQTY</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>PBDCRTBD</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>PBDSCHA</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>PBDSCHB</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>PBUPKKEY</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

DFSMSdss Patch Area

242  z/OS V1R13.0 DFSMSdss Storage Administration
## DFSMSdss Patch Area

<table>
<thead>
<tr>
<th>Name</th>
<th>Hex Offset</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBEOFNO</td>
<td>1D</td>
<td></td>
</tr>
<tr>
<td>PBEX2000</td>
<td>1C</td>
<td></td>
</tr>
<tr>
<td>PBFCDEF</td>
<td>4F</td>
<td></td>
</tr>
<tr>
<td>PBFUDGE</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>PBINVTOK</td>
<td>3E</td>
<td></td>
</tr>
<tr>
<td>PBMISCNT</td>
<td>1A</td>
<td></td>
</tr>
<tr>
<td>PBMOVLL</td>
<td>4E</td>
<td></td>
</tr>
<tr>
<td>PBMSCIOK</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>PBMXPTSK</td>
<td>4C</td>
<td></td>
</tr>
<tr>
<td>PBNACSAC</td>
<td>3D</td>
<td></td>
</tr>
<tr>
<td>PBNACSAU</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>PBNMVNCV</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>PBNOINIT</td>
<td>4D</td>
<td></td>
</tr>
<tr>
<td>PBNORLS</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>PBN0482F</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>PBNRACFI</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>PBNSELECT</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>PBRESERV</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>PBRESV60</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>PBRETRY#</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>PBREVOK</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>PBRNCRE</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>PBRNRFD</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>PBROCRE</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>PBROREFD</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>PBSRCVOL</td>
<td>3F</td>
<td></td>
</tr>
<tr>
<td>PBTIMOUT</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>PBTRKTH</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>PBTRKQTY</td>
<td>2C</td>
<td></td>
</tr>
<tr>
<td>PBUVSXPRE</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>PBVCVAL</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>PBVDPDSE</td>
<td>4B</td>
<td></td>
</tr>
<tr>
<td>PBVOLOPT</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>PBWAIT#</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>PBWAITFG</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>PBWFUMOD</td>
<td>4A</td>
<td></td>
</tr>
<tr>
<td>PBX99365</td>
<td>1B</td>
<td></td>
</tr>
<tr>
<td>PTCHARRY</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>PTCHEYE</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
DFSMSdss Patch Area
# Part 2. DFSMSdss Storage Administration Reference

## Chapter 15. Specifying DFSMSdss commands

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command syntax</td>
<td>253</td>
</tr>
<tr>
<td>How many subkeywords are allowed?</td>
<td>254</td>
</tr>
<tr>
<td>Specifying subkeywords in a command data set</td>
<td>254</td>
</tr>
<tr>
<td>How to read syntax diagrams</td>
<td>255</td>
</tr>
<tr>
<td>JCL that you need</td>
<td>257</td>
</tr>
<tr>
<td>How to control DFSMSdss through PARM information in the EXEC statement</td>
<td>258</td>
</tr>
<tr>
<td>Examples of invoking DFSMSdss with JCL</td>
<td>261</td>
</tr>
<tr>
<td>Moving a data set</td>
<td>261</td>
</tr>
<tr>
<td>Dumping a data set</td>
<td>262</td>
</tr>
<tr>
<td>Restoring a data set</td>
<td>262</td>
</tr>
</tbody>
</table>

## Chapter 16. DFSMSdss filtering—choosing the data sets you want processed

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How DFSMSdss filters data sets</td>
<td>263</td>
</tr>
<tr>
<td>Virtual storage access method (VSAM) data set considerations</td>
<td>263</td>
</tr>
<tr>
<td>Filtering by data set names</td>
<td>264</td>
</tr>
<tr>
<td>Using an asterisk in partially qualified data set names</td>
<td>264</td>
</tr>
<tr>
<td>Using a percent sign in partially qualified data set names</td>
<td>264</td>
</tr>
<tr>
<td>Examples of fully and partially qualified data set names</td>
<td>265</td>
</tr>
<tr>
<td>Fully qualified data set names</td>
<td>265</td>
</tr>
<tr>
<td>Partially qualified data set names using **</td>
<td>265</td>
</tr>
<tr>
<td>More partially qualified data set names</td>
<td>265</td>
</tr>
<tr>
<td>Relative generation filtering</td>
<td>266</td>
</tr>
<tr>
<td>Filtering by data set characteristics</td>
<td>266</td>
</tr>
<tr>
<td>Some examples of the BY keywords</td>
<td>269</td>
</tr>
<tr>
<td>Standard catalog search order</td>
<td>270</td>
</tr>
<tr>
<td>Broken data set considerations</td>
<td>270</td>
</tr>
</tbody>
</table>

## Chapter 17. Syntax—DFSMSdss function commands

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What DFSMSdss commands do</td>
<td>271</td>
</tr>
<tr>
<td>Building the stand-alone IPL-able core image</td>
<td>271</td>
</tr>
<tr>
<td>Using DUMP and RESTORE for backup and recovery</td>
<td>271</td>
</tr>
<tr>
<td>Moving data with COPY</td>
<td>271</td>
</tr>
<tr>
<td>Converting to and from Storage Management Subsystem (SMS) with CONVERTV</td>
<td>272</td>
</tr>
<tr>
<td>Managing space with COMPRESS, CONSOLIDATE, DEFRAG, and RELEASE</td>
<td>272</td>
</tr>
<tr>
<td>Using COPY for partitioned data set (PDS) and partitioned data set extended (PDSE) conversions</td>
<td>272</td>
</tr>
<tr>
<td>Printing for diagnostic purposes with PRINT</td>
<td>273</td>
</tr>
<tr>
<td>BUILDSA command for DFSMSdss</td>
<td>273</td>
</tr>
<tr>
<td>BUILDSA syntax</td>
<td>274</td>
</tr>
<tr>
<td>Explanation of BUILDSA command keywords</td>
<td>274</td>
</tr>
<tr>
<td>ADMINISTRATOR</td>
<td>274</td>
</tr>
<tr>
<td>INDDNAME</td>
<td>274</td>
</tr>
</tbody>
</table>

## Additional Commands

- IPL ........................................................................................................ 274
- OPERCNSL ............................................................................................ 275
- OUTDDNAME .......................................................................................... 276
- OUTDYNAM ............................................................................................ 276
- BUILDSA command examples .................................................................. 276
  - Example 1: core image using the default parameters .................... 276
  - Example 2: core image for IPL from tape .................................... 277
  - Example 3: core image for IPL from DASD .................................... 278
  - Example 4: core image for IPL from DASD with OPERCNSL option ...... 278
- CGCREATED command for DFSMSdss .................................................... 278
- CGCREATED syntax .............................................................................. 279
- Explanation of CGCREATED command keywords ................................ 279
  - ACCESSVOLUME .................................................................................. 279
  - FCCGVVERIFY .................................................................................. 280
- COMPRESS command for DFSMSdss .................................................... 280
- COMPRESS syntax ................................................................................ 280
- Explanation of COMPRESS command keywords .................................. 281
  - ADMINISTRATOR .............................................................................. 281
  - BY ....................................................................................................... 282
  - DDNAME .............................................................................................. 282
  - DYNALLOC ......................................................................................... 282
  - DYNAM ............................................................................................... 282
  - EXCLUDE ............................................................................................. 282
  - FILTERDD ......................................................................................... 283
  - INCLUDE .............................................................................................. 283
  - PASSWORD .......................................................................................... 284
  - WAIT ................................................................................................. 284
  - Example of compress operations .................................................. 285
- CONSOLIDATE command for DFSMSdss .............................................. 285
- CONSOLIDATE command syntax ......................................................... 285
- Explanation of CONSOLIDATE command keywords ............................. 287
  - ADMINISTRATOR .............................................................................. 287
  - BY ....................................................................................................... 287
  - CANCELERERROR .............................................................................. 288
  - DATASET ............................................................................................ 288
  - DEBUG ................................................................................................ 288
  - DYNAALLOC ....................................................................................... 289
  - EXCLUDE ............................................................................................. 290
  - FASTREPLICATION ............................................................................. 290
  - FCTOPPRCPRIORITY ......................................................................... 291
  - FILTERDD ......................................................................................... 292
  - FORCECP ............................................................................................ 292
  - INCLUDE .............................................................................................. 292
  - MAXTIME ............................................................................................ 293
  - PASSWORD .......................................................................................... 293
  - PHYSINDDNAME ............................................................................... 294
  - PHYSINDYNAM .................................................................................. 294
  - PROCESS ............................................................................................. 294
  - WAIT ................................................................................................. 294
  - WRITECHECK ..................................................................................... 295
- Example of a CONSOLIDATE operation ................................................ 295
- CONVERTV command for DFSMSdss ................................................... 296
Data integrity considerations for full or tracks dump operation ........................................ 424
Format of the output data set .................................................................................................. 424
Examples of full and tracks dump operations ........................................................................ 424
  Example 1: a data set dump .................................................................................................. 425
  Example 1A: a full dump operation ...................................................................................... 425
  Example 1B: a tracks dump operation ................................................................................ 425
  Example 1C: full volume dump operation with CONCURRENT ....................................... 425
Examples of physical data set dump operations .................................................................... 425
  Example 2: depicting DASD volume DUMP ....................................................................... 426
  Example 2A: using the INCLUDE subkeyword ................................................................. 426
  Example 2B: using the INCLUDE and EXCLUDE subkeywords ...................................... 426
  Example 2C: using the INCLUDE, EXCLUDE, and BY subkeywords .......................... 426
  Example 2D: with filtering data in a data set ..................................................................... 426
  Example 2E: with passwords in the input stream ............................................................... 427
  Example 2F: with passwords in a data set ......................................................................... 427
  Example 2G: wait for data sets if they or other data sets with the same name are in use by other jobs ................................................................. 427
  Example 2H: clearing volumes of uncataloged data sets .................................................. 427
Examples of logical data set dump operations ..................................................................... 428
  Example 1: dumping data sets constantly in use ............................................................... 428
  Example 2: dumping a user catalog and its aliases ........................................................... 428
  Example 3: logical data set dump operation with catalog filtering .................................. 428
  Example 4: logical data set dump operation with VTOC filtering .................................. 429
  Example 5: logical data set dump operation for Storage Management Subsystem (SMS) 430
  Example 6: logical dump operation with CONCURRENT ................................................ 430
  Example 7: clearing volumes of uncataloged data sets ..................................................... 430
PRINT command for DFSMSdss ......................................................................................... 431
PRINT syntax ....................................................................................................................... 431
Explanations of PRINT command keywords ................................................................. 432
  ADMINISTRATOR ............................................................................................................. 433
  ALLDATA ......................................................................................................................... 433
  CPVOLUME ..................................................................................................................... 433
  DATALength .................................................................................................................... 433
  DATASET ......................................................................................................................... 433
  DYNALLOC ....................................................................................................................... 434
  ERRORTRACKS ............................................................................................................... 434
  INDDNAME ..................................................................................................................... 434
  INDYNAM ......................................................................................................................... 434
  KEYLENGTH ................................................................................................................... 435
  OUTDDNAME .................................................................................................................. 435
  PASSWORD ....................................................................................................................... 435
  SHARE .............................................................................................................................. 436
  TOLERATE ....................................................................................................................... 436
  TRACKS ............................................................................................................................. 436
  VTOC ................................................................................................................................. 437
  WAIT ................................................................................................................................. 438
Examples of print operations ................................................................................................. 438
  Example 1: printing a range of tracks ............................................................................... 438
  Example 2: printing a component of a Virtual Storage Access Method (VSAM) cluster .... 439
  Example 3: printing a data set ......................................................................................... 439
RELEASE command for DFSMSdss .................................................................................. 440
RELEASE syntax for physical processing ......................................................................... 441
RELEASE syntax for logical processing .......................................................................... 442
Explanations of RELEASE command keywords .......................................................... 443
  ADMINISTRATOR ............................................................................................................. 443
  BY ................................................................................................................................... 446
  DDNAME .......................................................................................................................... 446
  DYNALLOC ....................................................................................................................... 446
  DYNAM .............................................................................................................................. 446
  EXCLUDE .......................................................................................................................... 446
  INCAT ................................................................................................................................ 446
  INCLUDE ............................................................................................................................ 446
  LOGDDNAME .................................................................................................................. 446
  LOGDYNAM ..................................................................................................................... 446
  MINEQTY .......................................................................................................................... 447
  MINTRACKSUNUSED ...................................................................................................... 447
  ONLYINCAT ..................................................................................................................... 447
  PASSWORD ....................................................................................................................... 447
  PROCESS ........................................................................................................................... 448
  SELECTMULTI ................................................................................................................ 448
  SPHERE ............................................................................................................................ 448
  STORGRP ........................................................................................................................ 449
  WAIT ................................................................................................................................ 449
Example of a release operation ............................................................................................ 450
RESTORE command for DFSMSdss .................................................................................. 451
Special considerations for RESTORE ............................................................................... 452
Data integrity considerations for full or tracks restore operations .................................... 452
RESTORE FULL and RESTORE TRACKS command syntax ....................................... 453
RESTORE DATASET command syntax for logical data set ............................................. 454
RESTORE DATASET command syntax for physical data set ........................................... 457
Explanations of RESTORE command keywords ......................................................... 458
  ADMINISTRATOR ............................................................................................................. 459
  AUTORELBLOCKADDRESS ......................................................................................... 460
  BY ................................................................................................................................... 460
  BYPASSACS ................................................................................................................... 460
  CANCELERROR ............................................................................................................... 460
  CATALOG or RECATALOG ............................................................................................. 461
  COPYVOLID .................................................................................................................... 462
  CPVOLUME ...................................................................................................................... 462
  DATASET .......................................................................................................................... 463
  DELETECATALOGENTRY ............................................................................................... 463
  DYNALLOC ....................................................................................................................... 465
  EXCLUDE .......................................................................................................................... 466
  FILTERDD ........................................................................................................................ 466
  FORCE ............................................................................................................................... 466
  FORCECP .......................................................................................................................... 467
  FREESPACE ...................................................................................................................... 467
  FULL .................................................................................................................................. 467
  IMPORT .............................................................................................................................. 468
  INCLUDE ............................................................................................................................ 469
  INDDNAME ..................................................................................................................... 469
  KEYPASSWORD ............................................................................................................... 469
LOGICALVOLUME ........... 470
MAKEMULTI ............. 470
MENTITY ................ 471
MGMTCLAS ............... 471
NOPACKING .............. 471
NULLMGMTCLAS ........... 472
NULLSTORCLAS ............ 472
OUTDDNAME .............. 472
OUTDYNAM ................ 472
OUTTRACKS ............... 473
PASSWORD ................ 473
PERCENTUTILIZED ............ 474
PROCESS ................. 474
PURGE .................... 475
REBLOCK .................. 475
RECATALOG ............... 476
REBLOCKADDRESS ......... 476
RENAME .................... 476
RENAMEUNCONDITIONAL ..... 479
SHARE ....................... 482
SPHERE ..................... 482
RSA ......................... 483
STORCLAS ................. 483
TGTALLOC .................. 483
TGTGDS ...................... 484
TOLERATE ................. 485
TRACKS ...................... 485
TTRADDRESS ............... 486
VOLCOUNT .................. 486
WAIT ........................ 488
WRITECHECK ............... 489

DFSMSdss RESTORE process 490
Assignment of class names by using the
RESTORE and COPY commands 492
Examples of full and tracks restore operations 493
Example 1: DASD volume-restore operation 493
Example 1A: full restore operation 493
Example 1B: tracks restore operation 493
Example 1C: tracks RESTORE—restore to
different tracks 494
Examples of physical data set restore operations 494
Example 2: label tape volume RESTORED to
DASD 494
Example 2A: using the INCLUDE
subkeyword to restore all data sets on a
dump tape 495
Example 2B: using the INCLUDE and
EXCLUDE subkeywords 495
Example 2C: using the INCLUDE,
EXCLUDE, and BY subkeywords 495
Example 2D: with filtering data in a data set 495
Example 2E: using the LOGICALVOLUME
and REPLACE keywords 496
Example 2F: using the REPLACE and
RENAME keywords 496
Example 2G: using the REPLACE and
RENAMEUNCONDITIONAL keywords 496
Example 2H: restore all data sets 496
Example 2I: restore data sets 497
Examples of logical data set restore operations 497
Example 1: logical data set
RESTORE—output volumes not specified 497
Example 2: logical restore of an unmovable
data set 498
Example 3: logical data set dump, followed
by a restore to an unlike device 498
Example 4: dump and restore for storage
management subsystem (SMS) conversion 499
Example 5: using the RENAME keyword to
restore a VSAM data set 499
Example 6: using the RECATALOG keyword
for a z/OS data set 500

Chapter 18. Syntax—auxiliary commands 501
Writing to the operator for DFSMSdss 501
WTO command 501
WTOR command for DFSMSdss 501
Scheduling tasks 502
SERIAL command for DFSMSdss 502
PARALLEL command for DFSMSdss 502
Controlling task processing 502
SET command—setting condition codes and
patch bytes 503
SET command for DFSMSdss 503
IF-THEN-ELSE command sequence for
DFSMSdss—using condition codes 504
IF-THEN-ELSE command sequence 505
Creating a null command 506
Continuation rules for IF-THEN-ELSE
command sequencing 506
Nesting IF commands 507
Examples of controlling task processing for
the IF-THEN-ELSE command 507
Common continuation errors 507
EOJ command—ending your DFSMSdss step 508

Chapter 19. DFSMSdss stand-alone services 509
Preparing to run the stand-alone services program 509
Running stand-alone services in 370 mode 509
Running stand-alone services in XA or ESA
mode 510
Running stand-alone services with a predefined
console 511
Using a tape library 511
IPLing and restoring from a tape library 512
Identifying procedures to mount and
demount tapes using the IBM Tape Library
Stand-Alone Device setup features 512
Using an automatic cartridge loader 514
Controlling command sequence processing 514
IPLing and running the Stand-Alone Services
Program 514
IPLing Stand-Alone Services 516
IPL example 517
Interpreting wait-state codes 518
RESTORE—restoring a formatted dump tape 520
RESTORE command syntax 521
Required parameters 521
Optional keywords 522
RESTORE command examples 523
TAPECNTL—rewinding and unloading a tape 525
Chapter 20. Data security and authorization checking

Chapter 21. Data integrity—serialization

Chapter 22. Application programming interface
Chapter 15. Specifying DFSMSdss commands

This topic is divided into the following sections:

- "Command syntax" describes syntax requirements.
- "How many subkeywords are allowed?" on page 254 explains the allowable number of subkeywords that you can specify in the input (command) stream.
- "Specifying subkeywords in a command data set" on page 254 explains how to use keywords in a command data set.
- "How to read syntax diagrams" on page 255 explains the syntax conventions used in this book.
- "JCL that you need" on page 257 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 506, 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.

```
  COPYDump
  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.

```
  BLK
  TRK
  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.

```
  CATalog
  RECATalog
```

- 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 TGETA1loc(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 258 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.

Notes:
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 138 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.
5. The output data set must be a standard format sequential data set and cannot use any extended-format features, such as compression.

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](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 (ADRnnnx). 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, SYSDUMP, 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.

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.

MSGTIME=I|W|E|T
Add timestamps to the requested message classes: I (informational), W
(warning), E (error) or T (terminating). Any combination of the classes is
accepted.

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=nnnn
nnnn 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 (1KB 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 nth 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.

UTILMSG=YES|NO|ERROR
This parameter controls the output of messages from auxiliary programs invoked by DFSMSdss (including ICKDSF, IDCAMS, IEBCOPY, IEBISAM, 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 ERR0R, messages are copied only if the auxiliary return program returns an error code to DFSMSdss. The default is ERR0R.

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.

Notes:
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 \texttt{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 \texttt{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=\texttt{YES}'. The default is \texttt{ZBUFF64R=\texttt{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:

```verbatim
// EXEC PGM=ADRDSSU,
// PARM='PAGENO=8,LINECT=57,SIZE=500K,UTILMSG=YES'

// EXEC PGM=ADRDSSU,
// PARM='TYPE=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 \texttt{z/OS MVS JCL Reference}.

**Related reading:** For additional information about the ADRUFO parameter list and the ADRUXIT exit, see the \texttt{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.

```jcl
//MYJOB 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=3390,VOL=(PRIVATE,SER=222222),DISP=OLD
//SYSIN DD *
COPY DATASET(INCLUDE(MYDATASET)) -
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:

```jcl
//MYJOB 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,VOL=SER=TAPE01,
// LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=MYDATASET.BACKUP
//SYSIN DD *
DUMP DATASET(INCLUDE(MYDATASET)) -
INDNAME(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 (MYDATASET) 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.

```jcl
//MYJOB JOB accounting information,REGION=nnnnK
//STEP1 EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=A
//TAPE DD UNIT=3480,VOL=SER=TAPE01,
// LABEL=(1,SL),DISP=(OLD,KEEP),DSNAME=MYDATASET.BACKUP
//DASD DD UNIT=3380,VOL=(PRIVATE,SER=111111),DISP=OLD
//SYSIN DD *
RESTORE DATASET(INCLUDE(MYDATASET)) -
INDNAME(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 describes how DFSMSdss filters data sets by data set name and by data set characteristic.
- “Filtering by data set names” on page 264 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 266 lists the data set characteristics on which filtering can be done.
- “Standard catalog search order” on page 270 identifies the standard order in which DFSMSdss searches for a data set name.

See Chapter 24, “Data set attributes,” on page 637 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 ways:

- Applying inclusion and exclusion criteria to fully or partially qualified data set names.
  - If you specify inclusion criteria (INCLUDE), DFSMSdss tentatively selects all data sets that fit any of the 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(**)). For the DEFRAG command, INCLUDE(**) is always implied.
  - If you specify exclusion criteria (EXCLUDE), DFSMSdss surveys the data sets that are selected with the inclusion criteria. It then rejects the data sets that fit any of the new criteria.
- Applying data set characteristics (BY) criteria to the data sets that are tentatively selected with the inclusion and exclusion criteria. 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 DEFRAG 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
Filtering

component is selected, the index component for the cluster is also selected. Again, the one exception is the DEFRAG 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 IAM.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:

```
        schar,op,(arg)
```

Where Represents

schar The selection characteristics:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOC</td>
<td>Allocation type (cylinder, track, block, absolute track, or movable)</td>
</tr>
<tr>
<td>CATLG</td>
<td>Whether the data set is currently cataloged or not (using the standard catalog search order)</td>
</tr>
<tr>
<td>CREDIT</td>
<td>Creation date (absolute or relative)</td>
</tr>
</tbody>
</table>
Filtering by Data Set Characteristics

**DSCHA** Whether the data-set-changed flag is on or off

**DSORG** Data set organization (SAM, PAM, PDS, PDSE, BDAM, HFS, EXCP, ISAM, 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 21 summarizes the permissible combinations of **schar**, **op**, and **arg**.

<table>
<thead>
<tr>
<th>schar</th>
<th>op</th>
<th>arg</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOC</td>
<td>EQ</td>
<td>CYL (cylinder allocation)</td>
<td>If MOV is picked, the data sets to be processed cannot be allocated as any of the following:</td>
</tr>
<tr>
<td></td>
<td>NE</td>
<td>TRK (track allocation)</td>
<td>• PSU (physical sequential unmovable)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLK (block length allocation)</td>
<td>• POU (partitioned organization unmovable)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ABSTR (absolute track allocation)</td>
<td>• DAU (BDAM unmovable)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOV (movable data sets)</td>
<td>• ABSTR (absolute tracks)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• ISAM (indexed sequential access method).</td>
</tr>
</tbody>
</table>

**COMPRESS command:** If MOV is picked, only data sets allocated as POU cannot be compressed.
### Table 21. BY Keywords (continued)

<table>
<thead>
<tr>
<th>schar</th>
<th>op</th>
<th>arg</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTI</td>
<td>EQ</td>
<td>YES (or 1)</td>
<td>YES: DFSMSdss processes only multivolume data sets. NO: DFSMSdss processes only single-volume data sets. <strong>DEFRAg command:</strong> 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. <strong>COMPRESS command:</strong> Because DFSMSdss assumes the data set is single volume, you do not need to specify MULTI. <strong>Note:</strong> Single volume data sets that have been allocated but have never been opened and are not cataloged may be selected as multivolume.</td>
</tr>
<tr>
<td>NO (or 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CATLG</td>
<td>EQ</td>
<td>YES (or 1)</td>
<td>YES: Only currently cataloged data sets are processed. NO: Only uncataloged data sets are processed. <strong>DUMP command:</strong> The CATLG filter is valid only when used with an input volume list (INDD, INDY, LOGINDD, LOGINDY, STORGRP). <strong>COPY command:</strong> Because DFSMSdss assumes the data set is single volume, you do not need to specify MULTI. <strong>Note:</strong> Single volume data sets that have been allocated but have never been opened and are not cataloged may be selected as multivolume.</td>
</tr>
<tr>
<td>NO (or 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREDT</td>
<td>LT</td>
<td>yyyddd(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 000001) is the first day of 2000. <em>n; Current date (run date), modified by an optional 1-digit to 4-digit positive or negative number of days, n. For example, <em>;5 is five days before this job is run. 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. <strong>Note:</strong> 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 CREDT or REFDT filter. The value used in the CREDT or REFDT filter should be for data sets at least two days old: (CREDTLE,</em>,-2) or (REFDTLE,</em>,-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.</td>
<td></td>
</tr>
<tr>
<td>EXPDT</td>
<td>GT</td>
<td>yyyddd(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 000001) is the first day of 2000. <em>n; Current date (run date), modified by an optional 1-digit to 4-digit positive or negative number of days, n. For example, <em>;5 is five days before this job is run. 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. <strong>Note:</strong> 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 CREDT or REFDT filter. The value used in the CREDT or REFDT filter should be for data sets at least two days old: (CREDTLE,</em>,-2) or (REFDTLE,</em>,-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.</td>
<td></td>
</tr>
<tr>
<td>REFDT</td>
<td>GE</td>
<td>yyyddd(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 000001) is the first day of 2000. <em>n; Current date (run date), modified by an optional 1-digit to 4-digit positive or negative number of days, n. For example, <em>;5 is five days before this job is run. 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. <strong>Note:</strong> 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 CREDT or REFDT filter. The value used in the CREDT or REFDT filter should be for data sets at least two days old: (CREDTLE,</em>,-2) or (REFDTLE,</em>,-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.</td>
<td></td>
</tr>
<tr>
<td>DSCHA</td>
<td>EQ</td>
<td>YES (or 1)</td>
<td>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.</td>
</tr>
<tr>
<td>NE (or 0)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSORG</td>
<td>EQ</td>
<td>SAM</td>
<td><strong>COMPRESS command:</strong> Because DFSMSdss assumes data set organization, you do not need to specify it. <strong>Note:</strong> The selective characteristic of DSORG can be specified more than once.</td>
</tr>
<tr>
<td>NE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATACLAS</td>
<td>EQ</td>
<td>An appropriate SMS class name.</td>
<td></td>
</tr>
<tr>
<td>MGMTCLAS</td>
<td>NE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STORCLAS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Filtering by Data Set Characteristics**
**Filtering by Data Set Characteristics**

Table 21. BY Keywords (continued)

<table>
<thead>
<tr>
<th>schar</th>
<th>op</th>
<th>arg</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTNT</td>
<td>LT</td>
<td>nnnnnnn (1-digit to 8-digit decimal</td>
<td>nnnnnnn is the number of used or allocated extents (EXTNT) or used or</td>
</tr>
<tr>
<td>FSIZE</td>
<td>GT</td>
<td>number, from 0 to 99999999)</td>
<td>allocated tracks (FSIZE). RESTORE command: The data set that was</td>
</tr>
<tr>
<td></td>
<td>EQ</td>
<td></td>
<td>dumped determines the number of used or allocated extents or tracks.</td>
</tr>
<tr>
<td></td>
<td>NE</td>
<td></td>
<td><strong>DUMP and COPY commands:</strong></td>
</tr>
<tr>
<td></td>
<td>GE</td>
<td></td>
<td>• For non-VSAM data sets:</td>
</tr>
<tr>
<td></td>
<td>LE</td>
<td></td>
<td>– If ALLDATA or ALLEXCP is specified, FSIZE is equal to the allocated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- tracks, and EXTNT is equal to the allocated extents.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– If ALLDATA or ALLEXCP is not specified, FSIZE is equal to the used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>tracks, and EXTNT is equal to the used extents.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> A specification of FSIZE,EQ,0 can be used to select PDSE data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sets that have no members (that is, have no used directory blocks),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>and a specification of FSIZE,NE,0 can be used to exclude PDSE data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>sets that have no members.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• For VSAM data sets, FSIZE is always equal to the allocated tracks,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>and EXTNT is always equal to the allocated extents.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Logical data set COPY, DUMP, and RESTORE commands:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• FSIZE criteria are applied once for the entire data set on all</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>volumes that the data set resides on. For logical processing of HFS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>files with TYPRUN NORUN, FSIZE equals all of the allocated space.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For logical processing of HFS files without TYPRUN NORUN, the used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>space represents the FSIZE unless ALLDATA was specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• EXTNT criteria are applied to non-VSAM data sets once for the data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>set on all volumes that the data set resides on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• EXTNT criteria are applied to each VSAM data component on all</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>volumes that the VSAM data component resides on. If a VSAM data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>component is selected by EXTNT filtering, the index component for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the cluster is automatically selected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Physical data set DUMP and RESTORE commands:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• FSIZE criteria are applied once for each volume being processed of a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>non-VSAM or VSAM data set. For HFS data sets, FSIZE equates to the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>allocated space, not the space actually used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• EXTNT criteria are applied once for each volume being processed of a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>non-VSAM data set or VSAM data component. If a VSAM data component</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>residing on a volume is selected by EXTNT filtering, the index</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>component (if any) residing on the same volume for the cluster is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>automatically selected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• FSIZE and EXTNT can be used to select certain volumes of a multivolume</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>data set without selecting all of the volumes that the data set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>resides on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>DEFRAG commands:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• For VSAM data sets, EXTNT and FSIZE criteria are applied at the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VSAM component level for the volume being processed only. EXTNT and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FSIZE filtering are performed for both VSAM data and index</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>components; VSAM index components are not automatically selected if</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the data component for the cluster is selected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• For non-VSAM data sets, EXTNT and FSIZE criteria are applied for the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>volume being processed only. For HFS data sets, FSIZE equates to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the allocated space, not the space actually used.</td>
</tr>
</tbody>
</table>

Notes:
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
Filtering by Data Set Characteristics

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. 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:

- BUILDSA
- CGCREATED
- COMPRESS
- CONSOLIDATE
- CONVERTV
- COPY
- COPYDUMP
- DEFRAG
- DUMP
- PRINT
- RELEASE
- RESTORE


What DFSMSdss commands do

DFSMSdss commands can perform a variety of functions.

Building the stand-alone IPL-able core image

The DFSMSdss BUILDSA 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 zSeries partitions and volumes, see Chapter 12, “Dumping and restoring Linux for zSeries partitions and volumes,” on page 185.

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.

BUILDDSA command for DFSMSdss

Use the BUILDDSA command to build the IPL-able core image for the Stand-Alone Services program. With the BUILDDSA 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 BUILDDSA function allocates temporary data sets if needed for BUILDDSA processing. These data sets are deleted when the BUILDDSA operation is complete. System-generated, temporary data set names are used.

The BUILDDSA function invokes the linkage editor (or Binder) utility. Specify UTILMSG=YES when you run the BUILDDSA function, and keep the output as a debugging reference when you run the Stand-Alone Services program.

The BUILDDSA 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.

Notes:
1. BUILDDSA 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 BUILDDSA 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 BUILDDSA task ends without processing input or output.
BUILDDSA Command for DFSMSdss

BUILDDSA syntax

Explanation of BUILDDSA command keywords

This section describes the keywords for the BUILDDSA command.

ADMINISTRATOR

ADMINISTRATOR specifies that you are a DFDSS-authorized or DFSMSdss-authorized administrator for the BUILDDSA 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 BUILDDSA command authorization levels” on page 526.

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 **BUILD**S**A** 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.V

**Note:**

**OPERCNSL**

**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 BUILDSA command. For more information about the predefined console, see "Running stand-alone services with a predefined console" on page 511.

**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.Volser on this volume and places the core image into this data set. If the SYS1.ADR.SAIPLD.Volser 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.

**BUILDSA 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.

```plaintext
//BUILDSA JOB accounting information,REGION= nnnnK
//STEP1 EXEC PGM=ADDRSSU,PARM='UTILMSG=YES'
//SAMODS DD DSN=SYS1.SADRYLIB,DISP=SHR
//CARDDD DD DSN=ADDRSA.IPLC,UNIT=3390,
//     DISP=(NEW,KEEP),VOL=SER=339001,
//     SPACE=(TRK,(40,5)),
//     DCB=(DSORG=PS,RECFM=F,BLKSIZE=80,LRECL=80)
//SYSPRINT DD SYSPUNCH
//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.

```plaintext
//COPYSA JOB accounting information
//STEP1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSPUNCH
//SYSUT1 DD DSN=ADDRSA.IPLC,DISP=(,KEEP),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.

```plaintext
//BUILDSA JOB accounting information,REGION= nnnnK
//STEP1 EXEC PGM=ADDRSSU,PARM='UTILMSG=YES'
//SAMODS DD DSN=SYS1.SADRYLIB,DISP=SHR
//TAPEDD DD DSN=ADDRSA.IPLT,UNIT=3480,LABEL=(,NL),
//     DISP=(NEW,KEEP),VOL=SER=TAPE01,
//     DCB=(DSORG=PS,RECFM=U,BLKSIZE=32760,LRECL=32760)
//SYSPRINT DD SYSPUNCH
//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'
//SYSPRT DD SYSOUT=A
//SAMODS DD DSN=SYS1.SADRLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//BUILD -
  INDDR(SAMODS) -
  OUTDYNM(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=ADRDSSU,PARM='UTILMSG=YES'
//SYSPRT DD SYSOUT=A
//SAMODS DD DSN=SYS1.SADRLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//BUILD -
  INDDR(SAMODS) -
  OUTDYNM(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®.
CGCREATED Command for DFSMSdss

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 135.

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 31.

For additional information about RACF FACILITY class profiles, refer to z/OS Security Server RACF Security Administrator's Guide.

CGCREATED syntax

```
CGCREATED ACCESSVOLUME (volser) | A |
```

**A: Optional keywords::**

- FCCGVERIFYfrzstate(volser)
- FCCGVFY(volser)

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.
CGCREATED Command for DFSMSdss

FCCGVERIFY

`volser` Specifies the volume serial number of the FlashCopy Consistency Group verification volume.

FCCGVERIFYFRZSTATE specifies the volume serial number of the verification volume 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.

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**

```
COMPRESS A FDD (ddn) ADMINistrator DYNALLOC
PASsword (ddn) WAIT (2,2)
PSWD dsn /pswd
```

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 539
COMPRESS Command for DFSMSdss

**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 266 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 15.

**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**


COMPRESS Command for DFSMSdss

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 15.

EXCLUDE

exclude (dsn)

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

filterdd (ddn)

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

include (dsn)

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 264 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.
COMPRESS Command for DFSMSdss

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 217.

Example of compress operations

The following example compresses a selected partitioned data set.

```
//JOB1  JOB accounting information,REGION=nnnnK
//STEP1  EXEC PGM=ADRDSSU
//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 293.

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:

```
```
CONSOLIDATE Command for DFSMSdss

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 539.

BY

BY (schar, op, arg)
CONSOLIDATE Command

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 266.

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 167.

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 263.

Note: Using the DATASET keyword requires that you also specify the FILTERDD, INCLUDE, EXCLUDE, or BY keywords.
You can use the DEBUG keyword as a diagnostic tool. Specify DEBUG with one of the following sub-keywords:

**TRACE**
Suggests that DFSMSdss is to print messages that identify the relocated extents.

**FRMSG**
Suggests 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.

**Notes:**
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 enqueuing, be used as the serialization method for relocating data set extents. Use DYNALLOC when you require cross-system serialization in a JES3 environment.

**Notes:**
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**

```
EXCLUDE(dsns)
```

The EXCLUDE keyword specifies one or more data sets (dsns) 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 287.

**Note:** You must use FILTERDD when you specify more than 255 entries on the INCLUDE, EXCLUDE, or BY keywords.

**FASTREPLICATION**

```
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.

FCTOPPRCPRIMARY

The FCTOPPRCPRIMARY 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 operation.

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.

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 or Global Copy pair currently in full duplex state, goes into a duplex pending state when the FlashCopy relationship is established. When Metro Mirror or Global Copy completes the copy operation, the Metro Mirror or Global Copy pair goes to full duplex state. To prevent Metro Mirror or Global Copy pairs from going to duplex pending state during FlashCopy operation, you must specify FCTOPPRCPrimary(PRESMIRREQ).

Notes:
1. Using FCTOPPRCPRIMARY might require RACF authorization.
2. When FlashCopy is not used to perform the CONSOLIDATE operation, the FCTOPPRCPRIMARY keyword is ignored.
CONSOLIDATE Command

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.

FILTERDD

```
FILTERDD
```

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

```
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

```
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.
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.

\[ \text{MAXTIME} \left( \text{nummins} \right) \]

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.

\[ \text{PASword} \left( \text{ddn} \right) \]

\[ \text{PSWD} \left( \text{dsn/pswd} \right) \]

\[ \text{ddn} \]

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.

\[ \text{dsn/pswd} \]

\[ \text{dsn} \]

\[ \text{pswd} \]

\[ \text{ddn} \]

\[ \text{dsn} \]

\[ \text{pswd} \]

\[ \text{ddn} \]

\[ \text{dsn} \]

\[ \text{pswd} \]

Notes:
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.
CONSOLIDATE Command

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**

```
PHYSINDDName(ddn)
```

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)
```

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.

**Notes:**
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 PHYSINDDNAME 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.

**Notes:**
1. SYS1.VVDS and SYS1.VTOCIX data sets are not processed.
2. To use PROCESS(SYS1), you might require RACF authorization.

**WAIT**

```
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.
**CONsolidate Command**

**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 217.

**WRITECHECK**

The WRITECHECK keyword specifies that the data set being processed is to be verified for successful completion.

**Notes:**
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)) -
    PHYSINDAME(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 identif 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 31.

CONVERTV command syntax

The syntax of the CONVERTV command is:

```
CONVERTV
```

A: The syntax of optional keywords with CONVERTV SMS is::

```
CATalog
INCAT(catname)
```

```
DName(ddn)
```

```
DYNAM(volser,unit)
```

```
SELECTMulti(ALL)
```

```
SELm(ANY, FIRST)
```

```
FORCECP(days)
```

```
PREPARE
```

```
TEST
```

```
SMS
```

```
NONSMS
```

```
A
```

```
SELECTMulti(ALL)
```

```
SELECTMulti(ALL)
```

```
SELM(ANY, FIRST)
```

```
CONVERTV Command for DFSMSdss
```

Guideline: Proper RACF security authorization might be required.

For additional information about RACF security authorization, see the Chapter 5, “Protecting DFSMSdss functions,” on page 31.

CONVERTV command syntax

The syntax of the CONVERTV command is:

```
CONVERTV
```

A: The syntax of optional keywords with CONVERTV SMS is::

```
CATalog
INCAT(catname)
```

```
DName(ddn)
```

```
DYNAM(volser,unit)
```

```
SELECTMulti(ALL)
```

```
SELm(ANY, FIRST)
```

```
FORCECP(days)
```

```
PREPARE
```

```
TEST
```

```
SMS
```

```
NONSMS
```

```
A
```

```
SELECTMulti(ALL)
```

```
SELECTMulti(ALL)
```

```
SELM(ANY, FIRST)
```

```
CONVERTV Command for DFSMSdss
```

Guideline: Proper RACF security authorization might be required.

For additional information about RACF security authorization, see the Chapter 5, “Protecting DFSMSdss functions,” on page 31.
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.

**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
CONVERTV Command

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(days)

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 297.

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 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.
CONVERTV Command

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.IFCAT.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.IFCAT.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.
CONVERTV Command

Example 3: using the CONVERTV command to convert from SMS

```
//JOB1   JOB accounting information,nnnnK
//STEP1  EXEC PGM=A0RDSUU
//SYSPRINT DD SYSPRINT=*
//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.

For more information about using the COPY command, see "Backup with concurrent copy" on page 42 and "Moving data sets with concurrent copy" on page 107.

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 zSeries file system (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.
COPY Command

- 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:
B: Optional Keywords Used for Logical Data Sets:

- ADMINistrator
- ALLData (dsn)
- ALLData (*)
- AUTORELBlockaddress
- AUTOREL九龙
- BYPASSACS (dsn)
- CANCELerror
- CATALOG
- RECATalog(newcatname)
- RECATalog(*)
- CHECKvloc
- CICSVRBACKUP
- CONCurrent
- CC PREFERRED NOTIFYCONCurrent
- ANYPREF NOTIFYCC
- CACHEpreferred
- CPref
- VIRTUALpreferred
- VPref
- REQUIRED
- ANYREQ
- CACHEDrequired
- CRcl
- VIRTUALREQUIRED
- VReq
- None
- STANDARD
- STD
- CONVERT (PDSE(dsn))
- PDS (dsn)
- PDS(dsn)
- PDSE (dsn)
- DELETE
- DEBUG (FRMSG (MINimal))
- DETAILED
- DYNALloc
- FCNOCOPY
- FCNOCOPYTOCOPY
- FCNOCTC
- FASTREPLICATION (PREFERRED)
- FASTREPLICATION
- FR
- REQUIRED (1)
- None (2)
- FCTOPPRCPRIPrimary
- FORCEP (days)
- FREESPACE (ci)
- FPS:
- INCAT (catname)
- ONLYINCAT
Notes:

1. Do not use the FASTREPlcation (REQuired) keyword with the
COPY Command

CONCURRENT(ANYPREF | ANYREQ | VIRTUALPREF | VIRTUALREQ | CACHEPREF | CACHEREQ) keyword.

2. Do not use the FASTREPLICATION (NONE) keyword with the FCNOCOPY or FCSTOPPPRPri mary 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

BYPASSACS

CHECKvtoc

DELete

DEBUG

DYNALloc

FCNC

FCNOCOPY

FCNOCOPYTOCOPY

FCNTC

FASTREPllication

FORCE

FCTOPPRCPrimary

FORCECP
days

MGMTCLAS

NULLMGMTCLAS

NMC

PASsword

PROCESS

PURGE

SYS1

UNDEFINEDSORG

UNDEFINEDSORG

UNDEFINEDSORG

SYS1

PSWD

dsn /pswd

CATalog

CATalog(newcatname)

RECATalog(*)

RECATalog(*)

RECATalog

SUMmarized

DETAILED

DTL

MINimal

CHECKvtoc

DELete

DEBUG

DYNALloc

FCNC

FCNOCOPY

FCNOCOPYTOCOPY

FCNTC

FASTREPllication

FORCE

FCTOPPRCPrimary

FORCECP
days

MGMTCLAS

NULLMGMTCLAS

NMC

PASsword

PROCESS

PURGE

SYS1

UNDEFINEDSORG

UNDEFINEDSORG

UNDEFINEDSORG

SYS1

PSWD

dsn /pswd

CATalog

CATalog(newcatname)

RECATalog(*)

RECATalog(*)

RECATalog

SUMmarized

DETAILED

DTL

MINimal

CHECKvtoc

DELete

DEBUG

DYNALloc

FCNC

FCNOCOPY

FCNOCOPYTOCOPY

FCNTC

FASTREPllication

FORCE

FCTOPPRCPrimary

FORCECP
days

MGMTCLAS

NULLMGMTCLAS

NMC

PASsword

PROCESS

PURGE

SYS1

UNDEFINEDSORG

UNDEFINEDSORG

UNDEFINEDSORG

SYS1

Notes:
1. Do not use the FASTREPllication (REQuired) keyword with the CONCURRENT(ANYPREF | ANYREQ | VIRTUALPREF | VIRTUALREQ | CACHEPREF | CACHEREQ) keyword.
2. Do not use the FASTREPlcation (NONE) keyword with the FCNOCOPY or FCTOPPRCPPrimary keywords.

COPY FULL and COPY TRACKS Syntax
COPY Command

C: Optional Keywords with COPY FULL:

- ADMINistrator
- ALLData
- ALLExcp
- CANcelerror
- CHECKvtoc
- CONCurrent
- CC
- PREFERRED
- ANYPREF
- CACHEpreferred
- CPref
- VIRTUALpreferred
- VPref
- REQUIRED
- ANYREQ
- CACHEREQuired
- CReq
- VIRTUALREQuired
- VReq
- NONE
- STANDARD
- STD
- COPYVolid
- CPYY
- DEBUG
- FRMSG
- MINimal
- SUMMARY
- DETAIL
- DTL
- FASTREplication
- PREFerred
- FastReplication
- REQUIRED
- (1)
- NOne
- DUMPCONDITIONing
COPY Command

D: Optional Keywords with COPY TRACKS:

- ADMINistrator
- CANcelerror
- CHECKvtoc
COPY Command

Notes:

1. Do not use the FASTREPLICATION (NONE) keyword with the FCFULLVOLUMERELATION, FCNOCOPY, FCSETGTOK, or FCTOPPFRCPRIORARY 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 539.
COPY Command

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 22 on page 370 and Table 23 on page 371 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 22 on page 370 and Table 23 on page 371 for more information.

Notes:
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
COPY Command

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 22 on page 370 and Table 23 on page 371 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.

AUTORELBLOCKADDRESS
AUTORELBLOCKADDRESS specifies that direct access data sets be automatically processed by relative block address rather than by track-track-record (TTR). The data sets must be accessed with an optional services code (OPTCD) setting. This setting indicates the data sets are organized by relative block address.

Notes:
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-spanned record formats or standard user labels.

For more information about the OPTCD, see z/OS DFSMS Macro Instructions for Data Sets.

BY

BY specifies that the data sets selected up to this point, by the processing of the INCLUDE and EXCLUDE keywords, are to be filtered further. To select the data set, all BY criteria must be met. See “Filtering by data set characteristics” on page 266 for a full discussion of schar, op, arg, and for more information about BY filtering.

Note: You must use FILTERDD when you have more than 255 entries in INCLUDE, EXCLUDE, or BY list keywords.

BYPASSACS

BYPASSACS specifies that the automatic class selection (ACS) routines are not invoked to determine the target data set class names. To specify BYPASSACS, RACF authorization may be required.

dsn Specifies a fully or partially qualified data set name.

If a data set is being renamed, the old name must be specified.

For more information about RACF authorization, see Chapter 5, “Protecting DFSMSdss functions,” on page 31.

For information about the assignment of class names using the COPY command, see “Assignment of class names by using the RESTORE and COPY commands” on page 492.
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 167.

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.

Notes:
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.

Notes:
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 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 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.

Notes:
1. CICSVRBACKUP is intended to be used with CICSVR. The minimum required CICSVR release is Version 3 Release 1. To use CICSVRBACKUP, the CICSVR server address space must be active.
2. CICSVRBACKUP applies to COPY DATASET for logical data set processing only.
3. CICSVR manages VSAM base clusters 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 prior to recovery and rebuilds the reusable AIXs after recovery, you need not copy the alternate indexes. DFSMSdss ignores the CICSVRBACKUP keyword when copying non-VSAM data sets.
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.

**Notes:**

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 58.

For more information about virtual concurrent copy and working space data sets, see "Performance considerations" on page 57 and **z/OS DFSMS Advanced Copy Services**.
COPY Command

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

Notes:
1. If the target data set is a PDSE, it must be SMS-managed.

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.

Notes:
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 demount 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.
COPY Command

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 263 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(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
COPY Command

Example 2: Data set copy

ADR918I (ttt)-mmmmm(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)-mmmmm(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)-mmmmm(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)-mmmmm(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)-mmmmm(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
COPY Command

Example 3: Data set copy, target data set is non-SMS-managed

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR918I</td>
<td>Fast replication could not be used for data set TEST.SRC.KSDS1, return code 14</td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 7 - VERSION 1 FC RELATION EXISTS</td>
<td></td>
</tr>
<tr>
<td>2 VOLUMES WERE REJECTED FOR QFRVOLS VOLUME REASON CODE 8 - MAX ESS FC RELATIONS</td>
<td></td>
</tr>
<tr>
<td>2 VOLUMES WERE REJECTED FOR QFRVOLS VOLUME REASON CODE 9 - FLASHCOPY NOT SUPPORTED</td>
<td></td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR DFMSDSS REASON CODE 1 - INSUFFICIENT SPACE</td>
<td></td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR DFMSDSS REASON CODE 2 - NO FREE DSCB IN THE VTOC</td>
<td></td>
</tr>
<tr>
<td>2 VOLUMES WERE REJECTED FOR DFMSDSS REASON CODE 3 - VOLUME IS SMS MANAGED</td>
<td></td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR DFMSDSS REASON CODE 4 - LSPACE MACRO FAILED WHILE CALCULATING FREE SPACE</td>
<td></td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR DFMSDSS REASON CODE 8 - DADSM FAILURE OCCURRED WHILE ALLOCATING THE DATA SET ON THE VOLUME</td>
<td></td>
</tr>
</tbody>
</table>

Example 4: Data set copy, target data set is SMS-managed

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR918I</td>
<td>Fast replication could not be used for data set TEST.SRC.KSDS1, return code 15</td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR QFRVOLS VOLUME REASON CODE 7 - VERSION 1 FC RELATION EXISTS</td>
<td></td>
</tr>
<tr>
<td>2 VOLUMES WERE REJECTED FOR QFRVOLS VOLUME REASON CODE 8 - MAX ESS FC RELATIONS</td>
<td></td>
</tr>
<tr>
<td>2 VOLUMES WERE REJECTED FOR QFRVOLS VOLUME REASON CODE 9 - FLASHCOPY NOT SUPPORTED</td>
<td></td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR DFMSDSS REASON CODE 1 - INSUFFICIENT SPACE</td>
<td></td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR DFMSDSS REASON CODE 2 - NO FREE DSCB IN THE VTOC</td>
<td></td>
</tr>
<tr>
<td>2 VOLUMES WERE REJECTED FOR DFMSDSS REASON CODE 3 - VOLUME IS SMS MANAGED</td>
<td></td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR DFMSDSS REASON CODE 4 - LSPACE MACRO FAILED WHILE CALCULATING FREE SPACE</td>
<td></td>
</tr>
<tr>
<td>1 VOLUME WAS REJECTED FOR DFMSDSS REASON CODE 8 - DADSM FAILURE OCCURRED WHILE ALLOCATING THE DATA SET ON THE VOLUME</td>
<td></td>
</tr>
</tbody>
</table>

Example 5: Full volume or tracks copy

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR918I</td>
<td>Fast replication could not be used for volume SRCV01 because the source and target devices do not provide compatible fast replication functions</td>
</tr>
<tr>
<td>VOLUME SRCV01 SUPPORTS DATA SET FLASHCOPY</td>
<td></td>
</tr>
<tr>
<td>VOLUME TGTV01 SUPPORTS SNAPSHOT</td>
<td></td>
</tr>
</tbody>
</table>

In examples 4 and 5, DEBUG(FRMSG(SUM)) and DEBUG(FRMAG(DTL)) result in the same level of informational message being issued.
Notes:
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)-nnnnn-yyyy, 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)-nnnnn-yyyy, 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
Example 4: Full volume or tracks copy

Notes:
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.

Notes:
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).
COPY Command

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 114.

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.”

Notes:
1. Do not use the DUMPCONDITIONING keyword with the COPYVOLID keyword.
COPY Command

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.

Notes:
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.

**Notes:**

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.
COPY Command

**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.

**Notes:**

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 FCCGCREEZE option requires the specified devices support the FlashCopy Consistency Group function.

**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.
COPY Command

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.

Notes:
1. Do not specify FCFASTREVERSERESTORE with any of the following keywords:
   - FCTOPPRCPRIMARY (PRESMIRPREF | PRESMIRREQ)
   - FASTREPLICATION (PREFERRED | NONE)
   - CONCURRENT, FCFREEZE, FCINCREMENTAL, FCINCREMENTALLAST, FCINCRVERIFY, FCNOCOPYTOCOPY, FCWAIT
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, 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.

Notes:
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
COPY Command

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/INDYDYNAM 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.

Notes:
1. Do not specify the FCINCREMENTAL keyword with any of the following keywords:
   - DATASET
   - FCINCREMENTALAST
   - 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 relationship is limited to one full volume relationship per volume. However, an Incremental FlashCopy relationship can coexist with other non-incremental 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 330 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.

Notes:
1. Do not specify the FCINCREMENTALLAST keyword with any of the following keywords:
   - DATASET
   - FCINCREMENTAL
   - CONCURRENT (ANYPREF | ANYREQ | CACHEPREF | CACHEREQ | VIRTUALPREF | VIRTUALREQ)
   - FASTREPLICATION(PREFERRED | NONE)
COPY Command

- 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 330 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
**COPY Command**

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.

**Notes:**
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 with existing FlashCopy no-background copy (NOCOPY) relationships 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.

**Notes:**
1. Do not specify the FCNOCOPYTOCOPY keyword with any of the following keywords:
   - DELETE
   - FASTREPLICATION(REQUIRED | PREFERRED | NONE)
   - FCCGFREEZE
   - FCINCREMENTAL | FCINCREMENTALLAST
   - FCNOCOPY
2. FCNOCOPYTOCOPY operation does not create a new copy of the source data.
COPY Command

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(FAILrelation)
```

FCSETGTOK specifies that when FlashCopy is used to perform a full volume copy operation, the target volume can be a space efficient volume. This type of FlashCopy relationship is called a *space efficient FlashCopy*. You can use 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 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.
COPY Command

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 531.

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 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
```

Notes:
1. Space efficient FlashCopy 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 might also be appropriate for longer term copies, if the source and target volumes are not frequently updated.
2. For a FlashCopy request, you must also specify the FCNOCOPY keyword when the intended target is a space efficient volume. Otherwise, your request will fail.
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 31.

For more information about RACF FACILITY class profiles, see “Protecting DFSMSdss functions with RACF FACILITY class profiles” on page 33.
FCTOPPRCPrimary 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 operation.

**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.

**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 or Global Copy pair currently in full duplex state, goes into a duplex pending state when the FlashCopy relationship is established. When Metro Mirror or Global Copy completes the copy operation, the Metro Mirror or Global Copy pair goes to full duplex state. To prevent Metro Mirror or Global Copy pairs from going to duplex pending state during FlashCopy operation, you must specify FCTOPPRCPrimary(PRESMIRREQ).

**Notes:**

1. Using FCTOPPRCPRIMARY might require RACF authorization.
2. Do not specify the FCTOPPRCPrimary keyword with the PRESMIRPREF and PRESMIRREQ subkeywords, if you specify the FCSETG TOK keyword with FAILRELATION.
3. Do not specify the FCTOPPRCPrimary keyword with the FASTREPLICATION(NONE) keyword.
4. When FlashCopy is not used to perform the copy operation, the FCTOPPRCPrimary keyword is ignored.
COPY Command

5. 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 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.

FCWAIT

```
FCWAIT(0,0)
```

```
FCWAIT(numsecs,numretries)
```

**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.

**Notes:**

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

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.

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 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.
COPY Command

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 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 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.
COPY Command

INCLUDE

```
INCLUDE(<dsn>)
```

*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 264. 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

```
INDDNAME(<ddn>)
```

*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 LOGINDYNAM 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

```
INDYNAM(<volser>,<unit>)
```


The INDYNAME keyword specifies the input volumes that are to be dynamically allocated and copied. The volume must be both mounted and online. 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. To ease JCL coding and command input without appreciably increasing run time, use the INDYNAME 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 INDYNAME) and also allows you to specify SELECTMULTI (which cannot be done with INDDNAME or INDYNAME).

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 344.

Refer to the description of SELECTMULTI in "LOGINDYNAM."

LOGINDYNAME

LOGINDYNAME 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 (*).

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
COPY Command

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:

Notes:
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 20, "Data security and authorization checking," on page 529.

**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.

**Notes:**

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 492 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.

\textit{dsn} \hspace{1cm} \text{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 402.

NULLMGMTCLAS
See "MGMTCLAS" on page 345.

NULLSTORCLAS
See "STORCLAS" on page 355.

ONLYINCAT
See "INCAT" on page 340.

OUTDDNAME

\textit{ddn} \hspace{1cm} \text{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 OUTFDYNAM for additional information.
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.

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**

**PSWD**
COPY Command

**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.
Notes:
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 22 on page 370 and Table 23 on page 371 for the action taken by DFSMSDss.
COPY Command

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 31.

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.

Notes:
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 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

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 the DASDCALC macro to determine the optimal block size for the target. The reblocking method used, DFSMSdss or DASDCALC, is presented to the installation reblock exit.

For more information about DFSMSdss processing of reblockable data sets, see

**RECATALOG**

See "CATALOG" on page 315.

**RELBLOCKADDRESS**

RELBLOCKADDRESS 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 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 access 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.

**Notes:**

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 CICS\-generated new name instead of the new name that you specified. See “CICSVRBACKUP” on page 317 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. For a physical data set copy, the RENAMEUNCONDITIONAL keyword is unsupported for VSAM data sets.

For more information about renaming, see "RENAME" on page 476.

**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.

**Notes:**
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
COPY Command

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.

Notes:
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.

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.
COPY Command

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 492 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 LOGINDNAM, LOGINDYNAM and STORGRP keywords” on page 344.

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
Notes:
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.

Notes:
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 90.
**COPY Command**

**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.

**Notes:**
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.

**Notes:**
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 21, “Data integrity—serialization,” on page 559.

**TRACKS**

TRACKS specifies ranges of tracks to be copied (that is, a tracks copy).
COPY Command

\[ c1,h1 \] 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:

<table>
<thead>
<tr>
<th>Specified</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Syntax error</td>
</tr>
<tr>
<td>c1</td>
<td>c1,0,c1,maximum head number</td>
</tr>
<tr>
<td>c1,h1</td>
<td>c1,h1,c1,maximum head number</td>
</tr>
<tr>
<td>c1,h1,c2</td>
<td>c1,h1,c2,maximum head number</td>
</tr>
<tr>
<td>c1,,c2</td>
<td>Syntax error</td>
</tr>
<tr>
<td>h1</td>
<td>Syntax error</td>
</tr>
<tr>
<td>c1,h1,X’FFFFFFF’</td>
<td>c1,h1,maximum cylinder number for the volume, maximum head number</td>
</tr>
</tbody>
</table>

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 22.

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(nnn)  nnn 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 nnn 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 nnn, whichever is less.
- If nnn 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 nnn, 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.

Notes:

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.
COPY Command

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(2,2)

WAIT(—numsecs—,—numretries—)

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 217.
WRITECHECK specifies that the data copied is to be verified for successful completion. This keyword increases the overall elapsed time.

Notes:
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

Example 1A: A Full 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" and "Example 1B: A Tracks Copy Operation."

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
```
COPY Command

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

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

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.
Example 4: A Data Set Copy with DELETE and RENAMEU Options

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

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...
COPY Command

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

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.

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.
COPY Command

Example 8: A Data Set Copy Using CONVERT PDSE

Example 9: A Data Set Copy with CONCURRENT

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 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
COPY Command

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

```plaintext
//DSSJOB JOB accounting information,REGION=nnnnK
//COPYSTEP EXEC PGM=ADDRDSSU
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
COPY DATASET(INCLUDE(OMVS.SB.MVS090.JV390.HFS, -
   OMVS.SB.MVS090.XML.HFS)) -
   RENAME((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 22 on page 370 and Table 23 on page 371 describe the functions of the ALLDATA and ALLEXCP keywords during a data set copy to LIKE and UNLIKE devices, respectively.
### COPY Command

**Table 22. 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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X X Y N X X N X X</td>
<td>X X X N N N N N X</td>
<td>Y Y Y Y Y Y N N N</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N X Y X X N N X</td>
<td>Y Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>X X N N N N N</td>
<td>Y Y Y Y Y Y Y Y</td>
<td>1 2 3 4 5 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 23. 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.

<table>
<thead>
<tr>
<th>Empty data set?</th>
<th>EOF as first record?</th>
<th>BLKSIZE=0 data set?</th>
<th>Undefined DSORG?</th>
<th>Sequential data set? (a)</th>
<th>Load module? (b)</th>
<th>Target tracksize &gt; source?</th>
<th>PROCEDURE (UNDEF) used?</th>
<th>ALLDATA(*) used?</th>
<th>ALLEXCP used?</th>
<th>DFSMSdss action is:</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>N</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>N</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>N</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>N</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>N</td>
<td>X</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>X</td>
<td>Y</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>Y</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Notes:
- (a) 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.
- (b) 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.

Notes:
1. Extra dump tapes can be used for such things as disaster recovery backup or distribution of dumped data (for example, a newly generated system).
2. 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
COPYDUMP Command

**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 EXECPGM=ADDRSSU
//SYSPRINT DD SYSOUT=A
//BACKUP DD UNIT=3480, VOL=SER=TAPE05, DISP=OLD,
  //DSNAME=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)).
COPYDUMP Command

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

```
DEFRAG DDName (ddn)
DYNAM (volser,unit)
ADMINistrator

BY (LIST (shar,op,(arg)))

CONSolidate

DEBUG (FRMSG (MINimal),TRACE,DETAILED,DTL)

EXClude (LIST (dsn))

FASTREPlication (PREFerred)
```
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:
DEFAG Command for DFSMSdss

- The RACF FACILLITY 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 539.

**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 266 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.
DEFRAG Command for DFSMSdss

DDNAME

```
DDName(ddn)
```

Specifies the name of the DD statement, `ddn`, that describes the volume to be processed.

DEBUG

```
DEBUG--(FRMSG--(MINimal--SUMmarized--DETAILED--DTL)--TRACE)
```

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 an informational message to explain why fast replication or Preserve Mirror operation could not be used during the DEFRAG operation. The DEBUG keyword overrides the DEBUG=FRMSG parameter 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.

If you specify DEBUG(FRMSG), you must also specify one of the following sub-keywords:

FRMSG(MINIMAL)

Specifies that DFSMSdss is to issue an informational message with a minimal level of information. The following examples show the messages that are issued when you use this keyword:

Examples:

```
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.

```
ADR918I (ttt)-mmmm(yy), FAST REPLICATION COULD NOT BE USED FOR VOLUME SRCV01, RETURN CODE 3
```

Return code 3 indicates that the source device is not eligible for fast replication at this time.
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:

Examples:

ADR918I (ttt)-mmmmm(yy), FAST REPLICATION COULD NOT BE USED FOR VOLUME SRCVO1, 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)-mmmmm(yy), FAST REPLICATION COULD NOT BE USED FOR VOLUME SRCVO1, RETURN CODE 3 VOLUME SRCVO1 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.

Guideline: 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 regarding 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:

ADR918I (ttt)-mmmmm(yy), FAST REPLICATION COULD NOT BE USED FOR VOLUME SRCVO1, RETURN CODE 3 VOLUME SRCVO1 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
**DEFRAG Command for DFSMSdss**

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(volser,unit)
```

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**

```
EXCLUDE("\nLIST("dsn"), "\nODName="ddn")
```

EXCLUDE specifies the list of data sets to be excluded from processing. This list is used for both DYNAM and DEFRAG commands.

- **dsn** Specifies the data set name to be excluded.
- **ddn** Specifies the data set name to be excluded.
**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 158.

**FASTREPLICATION**

- **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**

FCTOPPRCPrimary 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.

**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 or Global Copy pair currently in full duplex state, goes into a duplex pending state when the FlashCopy relationship is established. When Metro Mirror or Global Copy completes the copy operation, the Metro Mirror or Global Copy pair goes to full duplex state. To prevent Metro Mirror or Global Copy 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.

**Notes:**

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 31.

For additional information about RACF FACILITY class profiles see, “Protecting DFSMSdss functions with RACF FACILITY class profiles” on page 33.

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 "DEFRAO options" on page 159

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 \textit{WAIT} option” on page 561 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.

Notes:
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.
DEFRAG Command for DFSMSdss

MAXTIME

\[\text{MAXTIME} \rightarrow (\text{nummins})\]

nummins

Specifies the maximum number of minutes (0-9999 in decimal) a DEFRAG operation can run. A value of 0 is ignored.

Notes:
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 MAXIME is specified, the DEFRAG command terminates and DFSMS issues syntax error message ADR139E.

MMOVPCt

\[\text{MMOVPCt} \rightarrow (n, p, \text{PASSDelay}(nnn))\]

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 \(\text{MMOVPCt}(n,p)\). PASSDELAY is meaningful only if \(p\) is greater than one.

nnn

This 1- to 4-digit number specifies the time delay in milliseconds (1/1000 of a second). When \(\text{MMOVPCt}(n,p)\) 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 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.

Notes:
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 383 or MMOVPCT on page 385.

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 DASD VOL 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 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.
DEFRAG Command for DFSMSdss

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 217.

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=nmmnK
//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" and "Example 1B: with the names of excluded data sets in a data set."

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

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 16 on page 390 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:

```plaintext
//STEPT010 EXEC PGM=ADRDSSU,PARM='RACFLOG=YES,TRACE=YES'
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *
  DEFRAG DYNAM(D95060)  */
/*
```
DEFrag Command

Figure 16. 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:
DEFrag Command

ADR213I (001)-DFANL(01), 2007.226 12:50:17 ENDING STATISTICS ON C9NS05:
+STATISTICS+  +VOLUME+  +TRKAREA+
DATA SET EXTENTS RELOCATED  00000037  00000033
TRACKS RELOCATED  00000312  00001007
FREE CYLINDERS  00008690  00006526
FREE TRACKS  00000022  00000022
FREE EXTENTS  00000005  00000004
LARGEST FREE EXTENT (CYL,TRK)  00063426.03  00063426.03
FRAGMENTATION INDEX  0.050  0.010

Figure 17. 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 basic sequential data set, a large format sequential data set or an extended format sequential data set. The storage medium for a basic sequential data set can be a tape or DASD. The storage medium for the large format or extended format sequential data set must be DASD. You can dump data sets, an entire volume, or ranges of tracks.

Note: 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.

DFSMSdss offers two 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.

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.

Integrated catalog facility catalogs should not have a high-level qualifier of SYSCTLG because this causes DFSMSdss to treat them as CVOLs.

For more information about using the DUMP command, see Chapter 6, “Managing availability with DFSMSdss,” on page 35.

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
DUMP Command

- 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.

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 and DUMP TRACKS syntax

```
DUMP

DUMP FULL

DUMP TRACKS

A: Optional Keywords with DUMP FULL:
```

1. ADMINistrator
2. ALLData
3. ALLData
4. ALLExcp
5. ALLX
6. CANCELerror
7. CHECKvtoc
8. COMPRESS
9. HWCOMPRESS

```
```
B: Optional Keywords with DUMP TRACKS:

- ADMINistrator
- CANcelerror
- CHECKvtoc
- COMPRESS
- INCOMPRESS
- CONCurrent
- CPVOLUME
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 DATASET syntax for logical data set

C: Additional Keywords with DUMP DATASET for Logical Data Set:
D: Optional Keywords with DUMP DATASET for Logical Data Set:
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 LOGINDYNAM keywords at the same time.

DUMP DATASET syntax for physical data set

```plaintext
DUMP DATASET (E) Filterdd (ddn)
```
DUMP Command

E: Additional Keywords with DUMP DATASET for Physical Data Set:

```
BY((schar,op,(arg)))
```

```
INClude(**)
```

```
EXClude(dsn)
```

```
INClude(dsn)
```

```
EXClude(dsn)
```

```
BY((schar,op,(arg)))
```

```
INClude(**)
```

```
EXClude(dsn)
```

```
INClude(dsn)
```

```
EXClude(dsn)
```

```
BY((schar,op,(arg)))
```

F: Optional Keywords with DUMP DATASET for Physical Data Sets:

```
ADMINistrator
```

```
ALLData(dsn)
```

```
ALLData(*)
```

```
ALLExcp
```

```
ALLX
```

```
CANCELerror
```

```
CHECKvtoc
```

```
COMPRESS
```

```
INCOMPRESS
```

```
CONCurrent
```

```
CC PREFERRED NOTIFY
```

```
ANYPREF NOTIFY
```

```
CACHEpreferred
```

```
CPref
```

```
VIRTUALpreferred
```

```
VPref
```

```
REQUIRED
```

```
ANYREQ
```

```
CACHERequired
```

```
CReq
```

```
VIRTUALRequired
```

```
VReq
```

```
None
```

```
STANDARD
```

```
STD
```

```
DELeate
```

```
DYNALoc
```

```
FCWITHDRAW
```

```
OPTimize(1)
```

```
OPTimize(n)
```

```
PASsword(ddn)
```

```
PSWD
```

```
process(SYS1)
```

```
PURge
```

```
PRG
```

Chapter 17. Syntax—DFSMSdss function commands 397
**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.

**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 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.

For more information about how to use the ADMINISTRATOR keyword, see “ADMINISTRATOR keyword” on page 539.
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.)

Notes:
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 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
DUMP Command

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 (schar, op, (arg))

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
266 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 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. This enables CICSVR to manage backups made by DFSMSdss.

Notes:
1. CICSVRBACKUP is intended to be used in conjunction with CICSVR. The minimum required CICSVR release is Version 3 Release 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.

For more information about using DFSMSdss DUMP to create CICSVR backups, see [CICSVR V3R1 Implementation Guide](#).

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.

Notes:
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.
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.

Notes:
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 57. 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 263 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.

Notes:
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.

Notes:
1. The serialization is of value only when the dynamic allocation/JES3 interface is not disabled.
DUMP Command

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**

ENCRIPT is a subparameter of the RSA and KEYPASSWORD keywords.

For more information, see "KEYPASSWORD" on page 409 and "RSA" on page 418.

**EXCLUDE**

```
EXCLUDE (dsn)
```

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 400 and "INCLUDE" on page 408.

**FCWITHDRAW**

```
FCWITHDRAW (1)
```

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
DUMP Command

- The VTOC tracks on the source volume of the DUMP operation are the target of a FlashCopy relationship
- 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.

Notes:
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, using the FCNOCOPY keyword on the full volume copy and the FCWITHDRAW keyword on the full volume dump leaves the target volume (of the copy) in an indeterminate state. Some tracks on the volume might contain data from the source volume; other tracks might contain residual data from the target volume that existed before the copy. This indeterminate state can cause problems when accessing the target volume following the dump, if the VTOC locations of the source volumes and the target volumes are different before the copy.

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 occupied by the dump data. Hardware assisted compression is performed on the dumped data using the zSeries 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.

INCAT
INCAT\((\text{catname})\) specifies that DFSMSdss search the user catalogs specified by the INCAT\((\text{catname})\) 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.

\textit{catname} \quad \text{Specifies a fully qualified catalog name.}

\textbf{ONLYINCAT} \quad \text{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\((\text{catname})\) 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 31.

\textbf{INCLUDE}

\textit{dsn} \quad \text{Specifies the name of a data set eligible to be dumped. Either a fully or a partially qualified data set name can be used. See “Filtering by data set names” on page 264 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.}

\textbf{Notes:}
1. You must use FILTERDD when you have more than 255 entries in a list that is created 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 400 and “EXCLUDE” on page 405.

\textbf{INDDNAME}

\textit{ddn} \quad \text{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 (\textit{ddn}) must identify only one volume serial number.}
Notes:
1. Only one ddu 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.

Notes:
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 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 (,), forward slash (/), parentheses ()), or semi-colons. DFSMSdss removes leading and trailing blanks.

ICOUNT
DUMP Command

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.

ENCRIPT

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 DFSSMSdss on and the ENCRYPT sub-parameter you use.

For more information on the ENCRYPT keyword, see “RSA” on page 418.
DUMP Command

LOGIXDDNAME

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 LOGIXDDNAME, LOGINDYNAME, and STORGRP keywords:
1. When you specify the LOGIXDDNAME, LOGINDYNAME, 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 LOGIXDDNAME, LOGINDYNAME, or STORGRP keyword will not be dumped unless you specify SELECTMULTI.

For information about the SELECTMULTI keyword, see "LOGINDYNAME."

LOGINDYNAME

LOGINDYNAME specifies that volumes that contain the data sets to be dumped using logical processing are to be dynamically allocated.

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:
- LOGIXDDNAME
- LOGINDYNAME
- STORGRP

If logical volume filtering is not used, the specification of SELECTMULTI is not accepted.
DUMP Command

**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 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 406.

Notes:
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.
**DUMP Command**

**NOTIFYCONCURRENT**
See “CONCURRENT” on page 317.

**NOVALIDATE**
See “VALIDATE” on page 423.

**ONLYINCAT**
See “INCAT” on page 407.

**OPTIMIZE**

```
OPTimize(1)
```

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 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.

**Recommendation:** To improve performance and save dump space, specify the OPTIMIZE keyword with the COMPRESS keyword.

**OUTDDNAME**

```
OUTDDname (ddn)
```

`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.

**Notes:**
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.
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. The output data set must be a standard format sequential data set and cannot use any extended-format features, such as compression.

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.

Notes:

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 \(ddn\).

PHYSINDD

PHYSINDDname \((ddn)\)  PHYSINDDname (ddn)
DUMP Command

**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**
is 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**
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.VTOCI data sets can be physically, but
not logically, dumped. SYS1.VVDS and SYS1.VTOCI 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
DFSMdss functions,” on page 31.](#)

**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 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.

**Notes:**
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 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.

**Notes:**
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 215](#).
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.

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.

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.

Notes:
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 46.

For more information about dumping HFS data sets, see “Dumping HFS data sets” on page 562.

For more information about dumping zFS data sets, see “Dumping zFS data sets” on page 563.

**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.
DUMP Command

Notes:
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

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.

Notes:
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 46. For more information about TOL(ENQF), see Chapter 21, “Data integrity—serialization,” on page 559.

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.

Notes:
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.

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 \(X'FFFFFFFF'\) (or 268435455) as the high cylinder value. This causes DFSMSdss to choose the high cylinder value to be the end of the volume.
DUMP Command

Notes:
1. The $c_2$ must be greater than or equal to $c_1$.
2. If $c_2$ equals $c_1$, $h_2$ must be greater than or equal to $h_1$.

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:

<table>
<thead>
<tr>
<th>Specified</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Syntax error</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$c_1,0,c_1$, maximum head number</td>
</tr>
<tr>
<td>$c_1,h_1$</td>
<td>$c_1,h_1,c_1$, maximum head number</td>
</tr>
<tr>
<td>$c_1,h_1,c_2$</td>
<td>$c_1,h_1,c_2$, maximum head number</td>
</tr>
<tr>
<td>$c_1,c_2$</td>
<td>Syntax error</td>
</tr>
<tr>
<td>$h_1$</td>
<td>Syntax error</td>
</tr>
<tr>
<td>$c_1,h_1,X'FFFFFFF'$</td>
<td>$c_1,h_1$,maximum cylinder number for the volume, maximum head number</td>
</tr>
</tbody>
</table>

For more information about using TRACKS during a physical dump operation, see Chapter 6, “Managing availability with DFSMSdss,” on page 35.

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.)

Notes:
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 224.
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.

Notes:
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.

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.

\( \text{numsecs} \) Specifies a decimal number from 0 to 255 that designates the interval, in seconds, between retries.

\( \text{numretries} \) 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.

\( \text{numsecs} \) 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.

\( \text{numretries} \) 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 \( \text{numsecs,numretries} \) 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.
DUMP Command

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 217.

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 35.

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

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

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.
DUMP 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=1111111),DISP=OLD
//DASD2 DD UNIT=3380,VOL=(PRIVATE,SER=2222222),DISP=OLD
//TAPE DD UNIT=3480,VOL=SER=TAPE02,
// LABEL=(1,5L),DISP=(NEW,CATLG),DSNAME=USER2.BACKUP
//SYSIN DD *
/*
```

See command input in “Example 2A: using the INCLUDE subkeyword,” “Example 2B: using the INCLUDE and EXCLUDE subkeywords,” “Example 2C: using the INCLUDE, EXCLUDE, and BY subkeywords,” “Example 2D: with filtering data in a data set,” “Example 2E: with passwords in the input stream” on page 427, “Example 2F: with passwords in a data set” on page 427 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 427.

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
```
DUMP Command

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

```plaintext
//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
//SYSLIN DD *
//DUMP LOGINDNAME(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 enqueuing (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.

```plaintext
//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
//SYSLIN DD *
//DUMP OUTDDNAME(TAPE) -
//   DS(INCLUDE(MY.USER.CAT))
/*
```

Example 3: logical data set dump operation with catalog filtering

```plaintext
//JOB3 JOB accounting information,REGION=nnnnK
//STEP1 EXEC PGM=ADRDSSU
//TAPE DD UNIT=3480, VOL=SER=TAPE04,
//   LABEL=(1,SL), DISP=(NEW,CATLG), DSNAME=USER3.BACKUP
//SYSPRINT DD SYSOUT=A
//SYSLIN DD *
//DUMP OUTDD(TAPE) -
//   DS(INCLUDE(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)

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

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

This JCL clears volumes of uncataloged data sets.

DUMP Command

Example 5: logical data set dump operation for Storage Management Subsystem (SMS)

```plaintext
//JOB5 JOB accounting information,REGION=nnnnK
//STEP1 EXEC PGMP=ADROSSU
//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
/*
```

Example 6: logical dump operation with CONCURRENT

```plaintext
//JOB6 JOB accounting information,REGION=nnnnK
//DUMPSTEP EXEC PGMP=ADROSSU
//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
/*
```

Example 7: clearing volumes of uncataloged data sets

```plaintext
//JOB7 JOB accounting information,REGION=nnnnK
//STEP1 EXEC PGMP=ADROSSU
//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)) -
 OUTDDNAME(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.

**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 20, “Data security and authorization checking,” on page 529

**PRINT syntax**

```
Print
Prt
```
PRINT Command for DFSMSdss

A: Optional Keywords with PRINT DATASET:

- ALLData
- DYNAlloc
- SHARE
- SHR

B: Optional Keywords with PRINT TRACKS:

- CPVOLUME
- TOLERate (IOError)

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 539.

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

DATASET
**PRINT Command**

*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.
PRINT Command

*unit*  Specifies the device type of a DASD volume to be printed. This parameter is optional.

**KEYLENGTH**

```
KEYlength——(—n—)
```

*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**

```
OUTDDname——(——ddn——)
```

*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.

**Notes:**
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——(——ddn——)
```

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.
PRINT Command

Note: The PASSWORD keyword is not valid for a PRINT VTOC command.

Specify 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.

\[ ddn \]

\[ dsn/pswd \]

\[ dsn \] 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 23, "Examples of the application program with the user interaction module (UIM)," on page 619.

TRACKS

TRACKS specifies ranges of tracks to be printed.

\[ c1,h1 \]

\[ c1, max head \] Specifies the cylinder and head number of the beginning of the range.

Specify hexadecimal numbers as \[ \text{X}'c1' \] or \[ \text{X}'h1' \].
**PRINT Command**

*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:

<table>
<thead>
<tr>
<th>Specified</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Syntax error</td>
</tr>
<tr>
<td><em>c1</em></td>
<td>*c1,*0,<em>c1</em>, maximum head number</td>
</tr>
<tr>
<td>*c1,<em>h1</em></td>
<td>*c1,*h1,<em>c1</em>, maximum head number</td>
</tr>
<tr>
<td>*c1,*h1,<em>c2</em></td>
<td>*c1,*h1,<em>c2</em>, maximum head number</td>
</tr>
<tr>
<td>*c1,<em>c2</em></td>
<td>Syntax error</td>
</tr>
<tr>
<td><em>h1</em></td>
<td>Syntax error</td>
</tr>
<tr>
<td>*c1,<em>h1,X’FFFFFFF’</em></td>
<td>*c1,<em>h1</em>, maximum cylinder number for the volume, maximum head number</td>
</tr>
</tbody>
</table>

**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.

\textit{numsecs} \quad \text{Specifies a decimal number from 1 to 255 that designates the interval, in seconds, between retries.}

\textit{numretries} \quad \text{Specifies a decimal number from 0 to 99 that designates the number of retries to gain control of a data set.}

The default for \textit{numsecs},\textit{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 \textit{numsecs} or \textit{numretries}.

\textbf{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.

\textbf{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 217

\textbf{Examples of print operations}

The following are examples of the PRINT command.

\textbf{Example 1: printing a range of tracks}

```c
//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 TRACKS(1,0,1,5) INDDNAME(DASD)
/*
```

Prints a hard copy of tracks 0 through 5 from cylinder 1 on volume 111111.
Example 2: printing a component of a Virtual Storage Access Method (VSAM) cluster

```
//JO83 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

```
//STEP003 EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
PRINT DATASET(PUBSEXMP.SAM.S01) INDYNAM(D9S060)
/*
```

The output shown in Figure 18 on page 440 was produced from the example above.
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 will 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.

For additional information about filtering, see Chapter 16, “DFSMSdss filtering—choosing the data sets you want processed,” on page 263.

**RELEASE syntax for physical processing**

```
RELEASE A (see Logical Processing)
FILTERDD (ddn)
```
RELEASE Command for DFSMSdss

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 539.

**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 445 and "EXCLUDE" on page 444.

**DDNAME**

DDNAME requests physical processing.
**RELEASE Command**

**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 443 and "INCLUDE” on page 445.
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

INCLUDE (dsn)

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 264. If INCLUDE is omitted (but EXCLUDE or BY is specified) or if INCLUDE(ALL) 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 443 and “EXCLUDE” on page 444.

LOGDDNAME

LOGDDname (ddn)

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.
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 LOGINDDNAME, LOGINDYNAM, or STORGRP keyword must list all the volumes that contain a part of the data set.
   • The volume list 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.

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 LOGINDDNAME, LOGINDYNAM, or STORGRP keyword must contain a part of the data set.
   • Any volume specified in the volume list that is created by the LOGINDDNAME, 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 LOGINDDNAME, 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**

The password parameter sets the password used to access the data set during the release operation. The format is:

```
PSWD (ddn/pswd)
```

Where:
- **ddn** is the data set name.
- **pswd** is the password.
RELEASE Command

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 31.

SELECTMULTI

See “LOGDDNAME” on page 445, “LOGDYNAM” on page 446, and “STORGRP” on page 449.

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 LOGINDDNAME, 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 446.

**WAIT**

WAIT specifies to DFSMSdss the length of a wait in seconds and the number of retries to obtain control of a data set.

\[\text{WAIT}(\text{numsecs}, \text{numretries})\]

\[\text{WAIT}(2,2)\]

numsecs specifies a decimal number from 0 to 255 that designates the interval, in seconds, between retries.
**RELEASE Command**

$numretries$ 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" on page 217.

**Example of a release operation**

The following is an example of a release operation on selected sequential and partitioned data sets.

```plaintext
//JOB1 JOB accounting information,REGION=nnnnK
//STEP1 EXEC PGM=ADDRDSSU
//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:

```plaintext
//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:
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(**) -
BY(DSORG EQ VSAM) /* RELEASE ONLY VSAM */-
LOGDYNAM(339000) /* DYN ALLOC VOL 339000 */-
SELECTMULTI(ANY) /*EVEN IF MULTIVOL */-
SPHERE /* RELEASE AIKES */
/*
```

**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 two 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.

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

For more information about using the RESTORE command, see [Chapter 6, “Managing availability with DFSMSdss,” on page 35].

**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 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 INDX 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.

- 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
RESTORE Command

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 and RESTORE TRACKS command syntax

A: RESTORE FULL Command Optional Keywords:

Attention
RESTORE Command

B: RESTORE TRACKS Command Optional Keywords:

C: Additional Keywords Used for Logical Data Sets:

D: Optional Keywords Used for Logical Data Sets:
Chapter 17. Syntax—DFSMSdss function commands
RESTORE Command

RESTORE DATASET command syntax for physical data set

E: Additional Keywords for Physical Data Sets:

F: Optional Keywords Used for Physical Data Sets:
**Explanation of RESTORE command keywords**

This section describes the keywords for the RESTORE command.
The ADMINISTRATOR keyword allows you to act as a DFSMSdss authorized storage administrator for the RESTORE 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 539.

**Requirement:** You must also specify the ADMINISTRATOR keyword with CPVOLUME because DFSMSdss cannot check access authorization for VM data.

**AUTORELBLOCKADDRESS**

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.

**Notes:**
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*. 

" ADMINIST rator"
RESTORE Command

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 266.
- “EXCLUDE” on page 466.
- “INCLUDE” on page 469.

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.

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 31. For more information about data set names, see “Filtering by data set names” on page 264.

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:
RESTORE Command

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 215.

CATALOG or RECATALOG

| CATALOG | RECATALOG(newcatname) | 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.

Notes:
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.

8. For multivolume non-VSAM data sets, use IDCAMS DEFINE NONVSAM.

**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.

**Notes:**

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**
**RESTORE Command**

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
    DS
```

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 263.

**DELETECATALOGENTRY**

```
DELETECATALOGENTRY
    DELCATE
```

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 indicated by the catalog 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.

Notes:
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 volser(s) in the catalog do not match either the source volser(s) from the dump tape or the target volser(s) 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**

```plaintext
EXCLUDE (dsn)
```

*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 460
- “INCLUDE” on page 469

**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**

```plaintext
FILTERDD (ddn)
```

*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**

```plaintext
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.
RESTORE Command

Notes:
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(days)
```

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(CI,CA)
```

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.

Notes:
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.
REPLACE Command

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 452.

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 DELETECATALOGENTRY parameter to request that DFSMSdss perform a DELETE NOSCRATCH operation for you.
    Attention: Do not use the DELETECATALOGENTRY 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.

Notes:
1. IMPORT should be RACF-protected.
2. IMPORT is only supported for logical data set restore.
3. DELETECATALOGENTRY 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 31.

For more information about using the IMPORT keyword, see “Protecting the usage of DFSMSdss” on page 531.

For more information about using the DELETECATALOGENTRY keyword to successfully restore a data set with the old data set name, see Chapter 6, “Managing availability with DFSMSdss,” on page 35.

**INCLUDE**

```include
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 264. 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 460.
- “EXCLUDE” on page 466.

**INDDNAME**

```inddname
inddname (ddn)
```

*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
KEYPASSWORD (-password-)
```

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.
RESTORE Command

Note: The KEYPASSWORD password that is specified in your input command stream is not printed in the SYSPRINT output.

LOGICALVOLUME

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.
RESTORE Command

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 345.

NULLSTORCLAS
See "STORCLAS" on page 355.

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."

OUTDYNAM

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 ddname 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.
**RESTORE Command**

**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.

**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.

**Notes:**
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**
RESTORE Command

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 31.

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
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 the DASDCALC macro to determine the optimal block size for the
target. The reblocking method used, DFSMSdss or DASDCALC, 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 165.

**RECATALOG**
See “CATALOG” on page 315.

**RELBLOCKADDRESS**

RELBLOCKADDRESS 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 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 access 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
unreadable.

**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 cannot be renamed during a physical data set restore. 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.

Prefix (`pfx`) 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.

Old name (`on`) Specifies the old name to be used as filtering criteria for matching data set names on the target volume.

New name (`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 DSNNAME has one level, then restore with old name.
- A.** First level of DSNNAME replaced by A.
- A.B.** First two levels of DSNNAME replaced by “A.B”.
- *.A.** Second level of DSNNAME replaced by A.
- **.BCD Last level of DSNNAME replaced by BCD.
RESTORE Command

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 264.

“INCLUDE” on page 469.
RENAMENU CONDITIONAL

RENAMENU CONDITIONAL 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.

RENAMENU CONDITIONAL is not supported for VSAM 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.

Notes:
1. RENAME and RENAMENU CONDITIONAL are mutually exclusive; you cannot specify these keywords together. RENAMENU CONDITIONAL is not supported for physical restore of VSAM data sets, and, if specified, the data sets will not be restored.
2. RENAMENU CONDITIONAL 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.

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 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 pfx, on, and nn under “RENAME” on page 476.
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.

Notes:

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.

For more information about the REPLACE keyword, see “Special considerations 
    for RESTORE” on page 452.

**REPLACEUNCONDITIONAL**

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 can not be specified together.
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 already associated with the preallocated data set remains the same.

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.

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.
RESTORE Command

Notes:
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

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 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.

**Notes:**
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 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.

**Notes:**
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 21, “Data integrity—serialization,” on page 559.

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’.

Notes:
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’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:

<table>
<thead>
<tr>
<th>Specified</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Syntax error</td>
</tr>
<tr>
<td>c1</td>
<td>c1,0,c1, maximum head number</td>
</tr>
<tr>
<td>c1,h1</td>
<td>c1,h1,c1, maximum head number</td>
</tr>
<tr>
<td>c1,h1,c2</td>
<td>c1,h1,c2, maximum head number</td>
</tr>
<tr>
<td>c1,c2</td>
<td>Syntax error</td>
</tr>
</tbody>
</table>
**RESTORE Command**

Syntax error

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 452 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 22.

### 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.

**Notes:**

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.
RESTORE Command

Notes:
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(2,2)

WAIT(numsecs,numretries)

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.
RESTORE Command

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 217.

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 24 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 24. Physical Data Set Restore Actions on SMS-Managed Data Sets**

<table>
<thead>
<tr>
<th></th>
<th>Non-VSAM Physical Restore</th>
<th>VSAM Physical Restore</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENAME</td>
<td>N N Y Y Y Y - - - - N N Y - Y Y - - N N N</td>
<td></td>
</tr>
<tr>
<td>RENUNC</td>
<td>N N - - - - Y Y Y Y N N - Y - - Y Y N N N</td>
<td></td>
</tr>
<tr>
<td>REPLACE</td>
<td>Y Y N N Y Y N N Y Y N N N N X - - - - Y N -</td>
<td></td>
</tr>
<tr>
<td>OLD DATA SET</td>
<td>Y N Y N Y N Y N Y N Y N X Y Y Y N Y N N</td>
<td></td>
</tr>
<tr>
<td>EXISTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEW NAME ON</td>
<td>- - N X N - - - - - - - - Y Y - - - - - -</td>
<td></td>
</tr>
<tr>
<td>VOLUME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overlay old data set</td>
<td>T - - - - - - - - - - - - - - T - -</td>
<td></td>
</tr>
<tr>
<td>ALLOC with new name on USERVOL and restore</td>
<td>- - T - T - T T T T - - - - - - - - -</td>
<td></td>
</tr>
<tr>
<td>ALLOC with old name</td>
<td>- T - T - T - - - - - T - - - T - - - - T</td>
<td></td>
</tr>
<tr>
<td>Do not restore</td>
<td>- - - - - - - - - - - - - - T - T T T - T -</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- Y = Yes
- T = Action taken
- N = No
- X = Doesn't matter
- _ = Not Applicable
Figure 19 on page 491 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.

**Note:** When following the path that includes REPLACEU, the flow diagram applies only to a logical restore operation.
Figure 19. Restore Actions on Non-VSAM Data Sets. These actions apply to a data set RESTORE command with RENAME, RENAMEUNCONDITIONAL, REPLACE and REPLACEUNCONDITIONAL keywords.
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 20 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 20 only addresses how the management and storage class names are assigned, not how the storage group name is assigned or how volumes are selected.

Notes to Figure 20:

Figure 20. DFSMSdss Target Class Selection

END
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 35.

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,”
- “Example 1B: tracks restore operation,”
- “Example 1C: tracks RESTORE—restore to different tracks” on page 494.

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

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 425 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

For command input, see:

- “Example 2A: using the INCLUDE subkeyword to restore all data sets on a dump tape” on page 495.
- “Example 2B: using the INCLUDE and EXCLUDE subkeywords” on page 495.
- “Example 2C: using the INCLUDE, EXCLUDE, and BY subkeywords” on page 495.
- “Example 2D: with filtering data in a data set” on page 495.
- “Example 2E: using the LOGICALVOLUME and REPLACE keywords” on page 496.
- “Example 2F: using the REPLACE and RENAME keywords” on page 496.
- “Example 2G: using the REPLACE and RENAMEUNCONDITIONAL keywords” on page 496.
- “Example 2H: restore all data sets” on page 496.
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 RENUNC 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

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 enqueuing (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

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:

```bash
//SYSSN DD *
//SYSSN DD *
//SYSSN DD *
```
### Example 2: logical restore of an 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

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=ADDRSSU
//SYSPRINT DD SYSSUT**
//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)
/*
//SYSPRINT DD SYSSUT**
//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 initial part of “Example 4: dump and restore for storage management subsystem (SMS) conversion” 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 SYSSUT**
//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” 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=ADDRSSU
//TAPE DD UNIT=3480,VOL=SER=TAPE04,
//     LABEL=(1,SL),DISP=(OLD,KEEP),DSN=USER3.BACKUP
//SYSPRINT DD SYSSUT=A
//SYSIN DD *
RESTORE INDD(TAPE) OUTDDYN(338000) -
   DS(INCL(PARTS.VSAM1)) -
   RENAME(*.VSAM1,*.VSAM2) -
   CATALOG
/*
```
RESTORE Command

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=nmmnK
//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) -
   TGTALLOC(SOURCE)
```

In "Example 6: using the RECATALOG keyword," 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)).

TGTALLOC(SOURCE) specifies that the data sets are to be restored with the same allocation type they had when they were dumped.
Chapter 18. 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:

```plaintext
--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:

```plaintext
--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. Do not use SET LASTCC before the first function command.

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.

SET command for DFSMSdss

The syntax of the SET command is:

```
SET PATCH offset=value
    MAXCC=number
    LASTCC=number
```
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:

```
IF  LASTCC  op  number  THEN  command
    REPLY  op  text

ELSE  command
    DO  command  END
```

**IF**
Specifies that a comparison is to be made. The outcome determines which logical path (in your command sequence) DFSMSdss takes.

**LASTCC**
Specifies that LASTCC (condition code of the last-requested function, such as COMPRESS, CONVERTV, COPY, COPYDUMP, DEFRAG, DUMP, PRINT, RELEASE, RESTORE) be compared to a specified number.

**MAXCC**
Specifies that MAXCC (maximum condition code of all completed operations) be compared to a specified number. MAXCC is initialized to zero upon entry to DFSMSdss.

**REPLY**
Specifies that DFSMSdss compare the operator response to the last SYSIN command stream to the text specified with the `text` variable of the REPLY keyword.

**op**
Specifies an operator that describes a comparison that DFSMSdss will make. For example, you may have specified a returned condition code with either the LASTCC or MAXCC keyword. You can compare this value to the value of the `number` variable that you specify after the specific operator. You can also compare the operator's response to a WTOR command to the text that you specify with the `text` variable of the REPLY keyword. The compare operator can perform any one of six possible comparisons:

- **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

**number**
Specifies the decimal integer that is to be compared with MAXCC or LASTCC. Values greater than 16 are reduced to 16.

**text**
Specifies the text that is to be compared with the operator's response to the WTOR command. You must enclose the text string within quotation marks.

**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 or IF ... THEN;
ELSE ...

To specify a null ELSE command, specify:

IF ... THEN ... or IF ... THEN ...
ELSE 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

3. ELSE must be the first word on a line and no continuation character should be used on the preceding line.
   Example:
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 506), 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 -
      ELSE 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," 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.
Syntax—Auxiliary Commands

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:

>>>EOJ--------------------------------------<<<
Chapter 19. 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 IBM 3494 and 3495 tape libraries and 3590 tape subsystems
- Supports IPLing from a DASD volume, in addition to tape and card readers
- Allows you to predefined 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 System/390® processor in either S/390 mode or S/370 mode. The Stand-Alone Services program can run on a processor that is in BASIC or LPAR mode, or you can run the Stand-Alone Services program in a virtual machine under VM.

The Stand-Alone Services program operates in extended control (EC) mode and requires 2MB of real storage.

**Virtual Machine (VM) Note:** To specify EC mode while the Stand-Alone Services program is running under VM/370, enter:

```
CP SET ECMODE ON
```

Running stand-alone services in 370 mode

The following conditions apply to Stand-Alone Services operations in 370 mode:
- Tape libraries are not supported by Stand-Alone Services in 370 mode.
- The initial program load (IPL) device and the console must be attached to the same processor you IPLed from (when there are two or more processors).
- For DASD and tape devices:
  - In 370 mode, Stand-Alone Services does not issue Assign or Unassign commands for tape devices, or Device Reserve or Device Release commands for DASD devices.
Stand-Alone Services

- The user must ensure that all devices to be used by Stand-Alone Services are not accessed by other systems during IPL and while Stand-Alone Services operations are in process.

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 and follow your installation's procedures to prevent the device from interrupting until the Stand-Alone Services operation is complete.

Running stand-alone services in XA or ESA mode

The following conditions apply to Stand-Alone Services operations in XA or ESA mode:

- For DASD and tape devices:
  - In XA or ESA mode, 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.

Running stand-alone services with a predefined console

Use the OPERCNSL keyword of the BUILDSA command to predefined 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 ES/9000 service 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 BUILDSA 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 BUILDSA 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 BUILDSA 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 the IBM 3494 and 3495 tape libraries to IPL Stand-Alone Services and restore your dump data set tapes, and how to use the tape library menu options.

Notes:
1. Stand-Alone Services supports IBM tape libraries in XA or ESA mode only.
2. Tape drives to be used for Stand-Alone Services must remain offline to other systems.
Stand-Alone Services

3. 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 IBM 3494 and 3495 tape libraries. Table 25 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 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.

Similarly, you cannot use the DFSMSdss BUILDSA 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 25 to either IPL the Stand-Alone Services program or to restore data from dump data set tapes.

Table 25. Stand-Alone Services Options when Using an IBM 3494 or 3495 Tape Library. Procedures referred to in this table appear later in this section.

<table>
<thead>
<tr>
<th>Task</th>
<th>Can You Perform the Task Using an IBM Tape Library with:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No “Setup Stand-Alone Device” Feature?</td>
</tr>
<tr>
<td>IPL the Core image</td>
<td>No.</td>
</tr>
<tr>
<td>Restore from Dump Tapes</td>
<td>Only for tapes inside the library. Use the TAPEVOLSER keyword of the RESTORE command.</td>
</tr>
</tbody>
</table>

**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 25. Use Procedure A for tapes that reside inside the library. Use Procedure B for tapes that reside outside the library.
Stand-Alone Services

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.
**Stand-Alone Services**

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 502.

**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 21 on page 515](#) shows an overview of the Stand-Alone Services data restoration process.
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](#) and creating a Stand-Alone Services IPL-able core image with the BUILDSA command. The BUILDSA 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 BUILDSA command is described in “BUILDSA command for DFSMSdss” on page 273.

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 518. An example of the Stand-Alone Services system IPL is included in "IPL example" on page 517.

To IPL from the Stand-Alone Services core image (created with the BUILDSA 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 OPERCNSL 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 OPERCNSL keyword specified, then Stand-Alone Services attempts to use the device address specified by OPERCNSL as the operator console, rather than waiting for the first interrupt. See "Running stand-alone services with a predefined console" on page 511 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.

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 prnt, and $ccuu$ is the unit address. For example, to select a 3800 print subsystem at address 510, enter:
   ```
   prnt,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.
Interpreting wait-state codes

The following programming status word (PSW) wait-state codes can occur during the Stand-Alone Services IPL:

<table>
<thead>
<tr>
<th>Code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>000033</td>
<td>A program check occurred while Stand-Alone Services was being loaded. Contact your software service representative.</td>
</tr>
<tr>
<td>000044</td>
<td>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.</td>
</tr>
<tr>
<td>000055</td>
<td>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.</td>
</tr>
<tr>
<td>000066</td>
<td>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.</td>
</tr>
<tr>
<td>000077</td>
<td>The IPL loader is unable to locate the SYS1.ADR.SAIPLD.V&lt;volser&gt; data set on the IPL volume.</td>
</tr>
</tbody>
</table>
Stand-Alone Services

000088 The device type being used to IPL is not a supported DASD device for IPLing the 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.

00BCBC An attempt was made to IPL a processor that is in BC mode. IPL on a processor that supports EC mode. When you IPL from a VM user ID, set it to EC mode. This PSW is only loaded if you are IPLing from card or tape.

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 511.

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.
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.

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 511.

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.

A condition code 3 (not operational) has been received when attempting to communicate with the service console. Contact your hardware service representative.

An error occurred while trying to communicate with the service console. This can result from a hardware or a software problem.

Waiting for an I/O interrupt while Stand-Alone Services is being loaded.

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 produced by DFSMSdss without the use of a system environment.

The RESTORE command can restore from tape volumes 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 created by a DFSMSdss logical dump, from a DFSMSdss tracks dump using the CPVOLUME keyword, or from dump tapes produced by other utilities.

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:

```
/SM590000/SM590000
  RESTORE
  FROMDEV(3400)
  FROMDEV(TAPE)
  FROMADDR(ccuu)
  TOADDR(ccuu)
/SM590000/SM590000
  VERIFY(volser)
  NOVERIFY
  FULL
  STARTTRK(cccccc, h)
  ENDTRK(cccccc, h)
  READCHECK
  NOREADCHECK
  TAPEVOLSER(tserial)
  TAPEVOLSER(tserial)
  FILE(0001)
  FILE(nnnn)
/SM590000/SM630000
```

See "Command syntax" on page 253 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
Stand-Alone Services

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: NOVFY 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 cccccc 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 cccccc 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.
Stand-Alone Services

- 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 511 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.

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 example, 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.
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 IBM 3494 and 3495 Tape Libraries.

**TAPECNTL command syntax**

The syntax of the TAPECNTL command is:

```
TAPECNTL DEVTYPE(3400) UNITADDR(ccuu) REWIND
```

See "Command syntax" on page 253 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.
Stand-Alone Services

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 BUILDSA 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.

BUILD function
The BUILDSA 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 BUILDSA command and examples are presented in "BUILD command for DFSMSdss" on page 273.

Understanding BUILDSA command authorization levels
Your ability to use the BUILDSA command is determined by your level of access authorization to source (input) data sets and target (output) data sets or volumes used by the BUILDSA operation. Storage administrators with any of the following access levels may use the BUILDSA command:

• Without special authorization
• DASDVOL-access authority to a volume
• DFSMSdss authorization

Using BUILDSA without special authorization
To use the BUILDSA 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.V*volser* data set, you are not authorized to update cylinder 0 head 0 of the DASD volume.

**Using BUILDSA with DASDVOL-access authorization**

To use the BUILDSA 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 BUILDSA 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 BUILDSA command by specifying the ADMINISTRATOR keyword.

The FACILITY-class profile STGADMIN.ADR.STGADMIN.BUILDSA for the BUILDSA 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 BUILDSA command keywords” on page 274 for more information.
Chapter 20. 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.

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:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Required RACF Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic profile handling</td>
<td>RACF 1.5 or later</td>
</tr>
<tr>
<td>DASDVOL access authority</td>
<td>RACF 1.6 or later</td>
</tr>
<tr>
<td>FACILITY class authority</td>
<td>RACF 1.7 or later</td>
</tr>
<tr>
<td>Erase-on-scratch</td>
<td>RACF 1.7 or later</td>
</tr>
<tr>
<td>Group data set creation</td>
<td>RACF 1.8.1 or later</td>
</tr>
<tr>
<td>Storage class and management class</td>
<td>RACF 1.8.1 or later</td>
</tr>
</tbody>
</table>

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 538). 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 541 and “DASDVOL limitations” on page 543. 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 538 for details.
Data Security

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”
- “Protecting the usage of DFSMSdss”
- “Password protection” on page 532
- “Protected user and group data sets” on page 532
- “Generic and discrete profile considerations” on page 533
- “Security-level, category, and label checking” on page 535
- “Protect-all and always-call” on page 535
- “Standard naming conventions” on page 535
- “DFSMSdss temporary data set names” on page 535
- “Discretely protected multivolume data set” on page 537
- “Erase-on-scratch” on page 537
- “SMS-managed data set protection” on page 537
- “Logging” on page 538

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.

Table 26 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 539.

Table 26. DFSMSdss FACILITY Class Profiles. This table maps the DFSMSdss keywords and functions to their associated FACILITY class profiles.

<table>
<thead>
<tr>
<th>Keyword or Function</th>
<th>Profile Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYPASSACS with COPY</td>
<td>STGADMIN.ADR.COPY.BYPASSACS</td>
</tr>
<tr>
<td>BYPASSACS with RESTORE</td>
<td>STGADMIN.ADR.RESTORE.BYPASSACS</td>
</tr>
<tr>
<td>CGCREATED</td>
<td>STGADMIN.ADR.CGCREATE</td>
</tr>
<tr>
<td>CONCURRENT with COPY</td>
<td>STGADMIN.ADR.COPY.CNCURRENT</td>
</tr>
<tr>
<td>CONCURRENT with DUMP</td>
<td>STGADMIN.ADR.DUMP.CNCURRENT</td>
</tr>
<tr>
<td>CONSOLIDATE</td>
<td>STGADMIN.ADR.CONSOLID</td>
</tr>
</tbody>
</table>
Data Security

Table 26. DFSMSdss FACILITY Class Profiles (continued). This table maps the DFSMSdss keywords and functions to their associated FACILITY class profiles.

<table>
<thead>
<tr>
<th>Keyword or Function</th>
<th>Profile Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONVERTV</td>
<td>STGADMIN.ADR.CONVERTV</td>
</tr>
<tr>
<td>DEFRAG</td>
<td>STGADMIN.ADR.DEFRAG</td>
</tr>
<tr>
<td>DELETECATALOGENTRY with RESTORE</td>
<td>STGADMIN.ADR.RESTORE.DELCATE</td>
</tr>
<tr>
<td>FCCGFREEZE with COPY</td>
<td>STGADMIN.ADR.COPY.FCFREEZE</td>
</tr>
<tr>
<td>FCSETGTOK with COPY</td>
<td>STGADMIN.ADR.COPY.FCSETGT</td>
</tr>
<tr>
<td>FCTOPPRCP with COPY</td>
<td>STGADMIN.ADR.COPY.FCTOPPRCP</td>
</tr>
<tr>
<td>FCTOPPRCP with DEFRAG</td>
<td>STGADMIN.ADR.DEFRAG.FCTOPPRCP</td>
</tr>
<tr>
<td>FlashCopy with COPY</td>
<td>STGADMIN.ADR.COPY.FLASHCPY</td>
</tr>
<tr>
<td>FlashCopy with CONSOLIDATE</td>
<td>STGADMIN.ADR.CONSOLID.FLASHCPY</td>
</tr>
<tr>
<td>FlashCopy with DEFRAg</td>
<td>STGADMIN.ADR.DEFRAG.FLASHCPY</td>
</tr>
<tr>
<td>IMPORT with RESTORE</td>
<td>STGADMIN.ADR.RESTORE.IMPORT</td>
</tr>
<tr>
<td>INCAT(catname) with COPY</td>
<td>STGADMIN.ADR.COPY.INCAT</td>
</tr>
<tr>
<td>INCAT(catname) with DUMP</td>
<td>STGADMIN.ADR.DUMP.INCAT</td>
</tr>
<tr>
<td>INCAT(catname) with RELEASE</td>
<td>STGADMIN.ADR.RELEASE.INCAT</td>
</tr>
<tr>
<td>NEWNAMEUNCONDITIONAL with DUMP</td>
<td>STGADMIN.ADR.DUMP.NEWNAMEU</td>
</tr>
<tr>
<td>PROCESS(SYS1) with COPY</td>
<td>STGADMIN.ADR.COPY.PROCESS.SYS</td>
</tr>
<tr>
<td>PROCESS(SYS1) with DUMP</td>
<td>STGADMIN.ADR.DUMP.PROCESS.SYS</td>
</tr>
<tr>
<td>PROCESS(SYS1) with RELEASE</td>
<td>STGADMIN.ADR.RELEASE.PROCESS.SYS</td>
</tr>
<tr>
<td>SET PATCH</td>
<td>STGADMIN.ADR.PATCH</td>
</tr>
<tr>
<td>TOLERATE(ENQF) with COPY</td>
<td>STGADMIN.ADR.COPY.TOLERATE.ENQF</td>
</tr>
<tr>
<td>TOLERATE(ENQF) with DUMP</td>
<td>STGADMIN.ADR.DUMP.TOLERATE.ENQF</td>
</tr>
<tr>
<td>TOLERATE(ENQF) with RESTORE</td>
<td>STGADMIN.ADR.RESTORE.TOLERATE.ENQF</td>
</tr>
</tbody>
</table>

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 535. For more information about ADSP, refer to “Automatic data set protection (ADSP) attribute” on page 534.

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."

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**

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. However, DFSMSdss and the system services it uses (such as allocation) call RACF to make authorization checks that can result in access being denied because of a mismatch between the security level, security category, or security label of the resource and that of your user ID.

**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.DFSS.DEFRAG.xxxxxxxxx.volser.DUMMY

where xxxxxxxxx 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:

<table>
<thead>
<tr>
<th>Number of qualifiers (n)</th>
<th>Temporary name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dsnhlq.Atidasid.chmmsstt</td>
</tr>
<tr>
<td>2</td>
<td>First 2 qualifiers.Atidasid.chmmsstt</td>
</tr>
<tr>
<td>&gt; 2</td>
<td>First 3 qualifiers.Atidasid.chmmsstt</td>
</tr>
</tbody>
</table>

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 hmmmstt 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.
Data Security

First, DFSMSdss allocates a temporary data set into which records are temporarily exported with the following name format:

CATHLQ.EXPORT.Thmmssstt

where,

CATHLQ   First three high-level qualifiers of the catalog being copied

thmmstt   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.Thmmssstt

where,

CATHLQ   First four high-level qualifiers of the catalog that is being copied

thmmstt   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
Data Security

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 215.

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 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 both physical and logical 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 215.

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.CONSOLID
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.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.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 volume, data set and catalog 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:

- NOPASS was specified on the PPT statement of the SCHEDxx parmlib member for the ADRDSSU or user program which calls ADRDSSU. ADRDSSU (or its caller) is coded in the PGM parameter of the EXEC statement and should be authorized.
- 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.
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 22 on page 542 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. Use the diagram to clarify the data security decisions under varying conditions.

Related reading: For more information about DFSMSdss use of utilities, see “Moving data sets with utilities” on page 105.

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.

Data Security
**Data Security**

**DASDVOL access authority**

When you have the required level of DASDVOL authority to a volume, as shown in Table 27 on page 543, you can perform maintenance operations such as dump, restore, scratch and rename of the data sets on that volume regardless of your DFSMSdss function requested.

- Authorized storage administrator?
  - Yes
  - No

- COPYDUMP function?
  - Yes
  - No

- Has installation exit requested that authorization checking be bypassed?
  - Yes
  - No

- Is this an authorized program call for which NOPASS = ON?
  - Yes
  - No

- Is this one of the following data sets?
  - Checkpoint / restart
  - Integrated catalog facility
  - VSAM without VVDS
  - VTOC index
  - VVDS
  - Yes
  - No

- Is RACF active in the system?
  - Yes
  - No

- RACF facility class for function?
  - Yes
  - No

- Does user have DASDVOL authority for doing a volume level function?
  - Yes
  - No

- Does user have RACF data set authority for a logical operation for source plus target data sets that are:
  - VSAM and non-VSAM
  - Cataloged and uncataloged
  - Yes
  - No

- Terminate request

- Does operation require a catalog update?
  - Yes
  - No

- Catalog is RACF-protected?
  - Yes
  - No

- Protection is to be:
  - Bypass catalog protection
  - Honored
  - Authorized
  - Not authorized

- Terminate request

- Does user have the authority allowed by the specific facility class?
  - Yes
  - No

- Prompt the operator for authority

- Authority granted

- Authority rejected

- Terminate request

- Obtain password that was passed to DFSMSdss or by a password operator prompt

- Valid password

- Invalid password

- Terminate request

- Perform requested function

---

**Figure 22. DFSMSdss Data Security Decisions**

---

**z/OS V1R13.0 DFSMSdss Storage Administration**

542
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 27. DASDVOL Access Authority. This table shows the DASDVOL-access authorities needed to perform volume and data-set-level functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Command</th>
<th>Volume</th>
<th>FULL</th>
<th>TRACKS</th>
<th>Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Volume</td>
<td>FULL</td>
<td>TRACKS</td>
<td>Data Set</td>
<td></td>
</tr>
<tr>
<td>COPY</td>
<td>Source</td>
<td>READ</td>
<td>READ</td>
<td>READ</td>
<td></td>
</tr>
<tr>
<td>COPY with DELETE</td>
<td>Source</td>
<td>N/A</td>
<td>N/A</td>
<td>ALTER</td>
<td></td>
</tr>
<tr>
<td>COPYDUMP</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>DEFRAIG</td>
<td>Source</td>
<td>N/A</td>
<td>N/A</td>
<td>UPDATE</td>
<td></td>
</tr>
<tr>
<td>DUMP</td>
<td>Source</td>
<td>READ</td>
<td>READ</td>
<td>READ</td>
<td></td>
</tr>
<tr>
<td>DUMP with DELETE</td>
<td>Source</td>
<td>N/A</td>
<td>N/A</td>
<td>ALTER</td>
<td></td>
</tr>
<tr>
<td>PRINT</td>
<td>Source</td>
<td>N/A</td>
<td>READ</td>
<td>READ</td>
<td></td>
</tr>
<tr>
<td>RELEASE</td>
<td>Source</td>
<td>N/A</td>
<td>N/A</td>
<td>UPDATE</td>
<td></td>
</tr>
<tr>
<td>RESTORE</td>
<td>Source</td>
<td>N/A</td>
<td>N/A</td>
<td>READ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Target</td>
<td>ALTER</td>
<td>ALTER</td>
<td>UPDATE</td>
<td></td>
</tr>
</tbody>
</table>

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 authority.

For more information about discrete profiles, see “Copy and data set profile considerations” on page 548 and “Restore and data set profile considerations” on page 554.
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 545.
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

This section covers the following commands:
- CGCREATED
- COMPRESS
- CONSOLIDATE
- CONVERTV
- COPY
- COPYDUMP
- DEFRAG
- DUMP
- PRINT
- RELEASE
- REPLACE

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.

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.
Data Security

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 28, 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 28 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 28. Access Authority for the DFSMSdss COPY Command. This table shows the minimum access levels required to perform a COPY command.

<table>
<thead>
<tr>
<th>SOURCE DATA SET</th>
<th>TARGET DATA SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-SMS</td>
<td>Non-SMS</td>
</tr>
<tr>
<td>With DELETE</td>
<td>Access Level</td>
</tr>
<tr>
<td>Data Set</td>
<td>Catalog</td>
</tr>
<tr>
<td>NO</td>
<td>READ</td>
</tr>
<tr>
<td>NO</td>
<td>READ</td>
</tr>
<tr>
<td>YES</td>
<td>ALTER</td>
</tr>
<tr>
<td>YES</td>
<td>ALTER</td>
</tr>
<tr>
<td>Non-SMS</td>
<td>SMS</td>
</tr>
<tr>
<td>With DELETE</td>
<td>Access Level</td>
</tr>
<tr>
<td>Data Set</td>
<td>Catalog</td>
</tr>
<tr>
<td>NO</td>
<td>READ</td>
</tr>
<tr>
<td>NO</td>
<td>READ</td>
</tr>
<tr>
<td>YES</td>
<td>ALTER</td>
</tr>
<tr>
<td>YES</td>
<td>ALTER</td>
</tr>
<tr>
<td>SMS</td>
<td>Non-SMS</td>
</tr>
<tr>
<td>With DELETE</td>
<td>Access Level</td>
</tr>
<tr>
<td>Data Set</td>
<td>Catalog</td>
</tr>
<tr>
<td>NO</td>
<td>READ</td>
</tr>
<tr>
<td>NO</td>
<td>READ</td>
</tr>
<tr>
<td>YES</td>
<td>ALTER</td>
</tr>
<tr>
<td>YES</td>
<td>ALTER</td>
</tr>
</tbody>
</table>
Table 28. Access Authority for the DFSMSdss COPY Command (continued). This table shows the minimum access levels required to perform a COPY command.

<table>
<thead>
<tr>
<th>SOURCE DATA SET</th>
<th>TARGET DATA SET</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>With DELETE</strong></td>
<td><strong>Access Level</strong></td>
</tr>
<tr>
<td><strong>Data Set</strong></td>
<td><strong>Catalog</strong></td>
</tr>
<tr>
<td>NO</td>
<td>READ</td>
</tr>
<tr>
<td>NO</td>
<td>READ</td>
</tr>
<tr>
<td>YES</td>
<td>ALTER</td>
</tr>
<tr>
<td>YES</td>
<td>ALTER</td>
</tr>
</tbody>
</table>

Notes:
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 [53](#) 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 29](#)). 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 29. DFSMSdss COPY Command and RACF Profiles. This table summarizes how DFSMSdss defines discrete profiles to protect the target set.

<table>
<thead>
<tr>
<th>COPY with:</th>
<th>Source Data Set Protected?</th>
<th>RACF Profile:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RENAMEN</strong></td>
<td><strong>DELETE</strong></td>
<td>Source Target</td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>None As predefined</td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>GENERIC No change As predefined</td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>DISCRETE No change DEFINE</td>
</tr>
<tr>
<td>NO</td>
<td>YES</td>
<td>NO None As predefined</td>
</tr>
<tr>
<td>NO</td>
<td>YES</td>
<td>GENERIC No change As predefined</td>
</tr>
<tr>
<td>NO</td>
<td>YES</td>
<td>DISCRETE DELETE DEFINE</td>
</tr>
<tr>
<td>YES</td>
<td>NO</td>
<td>NO None As predefined</td>
</tr>
<tr>
<td>YES</td>
<td>NO</td>
<td>GENERIC No change MENTITY</td>
</tr>
</tbody>
</table>
Table 29. DFSMSdss COPY Command and RACF Profiles (continued). This table summarizes how DFSMSdss defines discrete profiles to protect the target set.

<table>
<thead>
<tr>
<th>COPY with:</th>
<th>Source Data Set Protected?</th>
<th>RACF Profile:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENAME</td>
<td>Source</td>
<td>Target</td>
</tr>
<tr>
<td>YES</td>
<td>NO</td>
<td>DISCRETE</td>
</tr>
<tr>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>YES</td>
<td>YES</td>
<td>GENERIC</td>
</tr>
<tr>
<td>YES</td>
<td>YES</td>
<td>DISCRETE</td>
</tr>
</tbody>
</table>

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 30 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 30. DFSMSdss Copy and Define Discrete Profile Summary. This table summarizes what happens when DFSMSdss defines discrete profiles for a copy operation.

<table>
<thead>
<tr>
<th>Protection</th>
<th>MENTITY Specified?</th>
<th>Define Profile?</th>
<th>RACF Indicated?</th>
<th>Warning Message?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Target</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>None</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>None</td>
<td>Generic</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 30. DFSMSdss Copy and Define Discrete Profile Summary (continued). This table summarizes what happens when DFSMSdss defines discrete profiles for a copy operation.

<table>
<thead>
<tr>
<th>Protection</th>
<th>MENTITY Specified?</th>
<th>Define Profile?</th>
<th>RACF Indicated?</th>
<th>Warning Message?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Target</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>RACF Indicated</td>
<td>Yes</td>
<td>Yes</td>
<td>No (W)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Generic</td>
<td>None</td>
<td>Yes</td>
<td>Yes</td>
<td>No (W)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Generic</td>
<td>Generic</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>RACF Indicated</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RACF Indicated</td>
<td>None</td>
<td>Yes</td>
<td>Yes</td>
<td>No (W)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>Yes (C)</td>
<td>No (E)</td>
</tr>
<tr>
<td>RACF Indicated</td>
<td>Generic</td>
<td>Yes</td>
<td>Yes</td>
<td>No (W)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>Yes (C)</td>
<td>No (E)</td>
</tr>
<tr>
<td>RACF Indicated</td>
<td>RACF Indicated</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

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.

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.
Data Security

- 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:
Data Security

- 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 31.

Table 31 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.

Table 31. Access Authority for the DFSMSdss DUMP Command. This table shows the minimum access levels required to perform a DUMP command.

<table>
<thead>
<tr>
<th>SMS-Managed Data Set?</th>
<th>With DELETE?</th>
<th>Access Level:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Source Data Set</td>
<td>Catalog</td>
</tr>
<tr>
<td>NO</td>
<td>NO</td>
<td>READ</td>
<td>Automatic</td>
</tr>
<tr>
<td>NO</td>
<td>YES</td>
<td>ALTER</td>
<td>UPDATE</td>
</tr>
<tr>
<td>YES</td>
<td>NO</td>
<td>READ</td>
<td>Automatic</td>
</tr>
<tr>
<td>YES</td>
<td>YES</td>
<td>ALTER</td>
<td>Automatic</td>
</tr>
</tbody>
</table>

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 32.

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 32. Access Authority for the DFSMSdss RESTORE Command. This table shows the minimum access-levels required to perform a RESTORE command.
**Table 32. Access Authority for the DFSMSdss RESTORE Command (continued).** This table shows the minimum access-levels required to perform a RESTORE command.

<table>
<thead>
<tr>
<th>SOURCE DATA SET</th>
<th>TARGET DATA SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>No check</td>
<td>No check</td>
</tr>
<tr>
<td>No check</td>
<td>No check</td>
</tr>
<tr>
<td>READ</td>
<td>Automatic</td>
</tr>
<tr>
<td>READ</td>
<td>Automatic</td>
</tr>
</tbody>
</table>

**SMS**

<table>
<thead>
<tr>
<th>SOURCE DATA SET</th>
<th>TARGET DATA SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>No check</td>
</tr>
<tr>
<td>NO</td>
<td>No check</td>
</tr>
<tr>
<td>YES</td>
<td>READ</td>
</tr>
<tr>
<td>YES</td>
<td>READ</td>
</tr>
</tbody>
</table>

**Notes:**
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 32 on page 553 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 32 on page 553 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.
**Data Security**

**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 33 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.

**Table 33. DFSMSdss Copy and Define Profile Summary.** This table summarizes when DFSMSdss defines discrete profiles for a copy operation.

<table>
<thead>
<tr>
<th>Protection</th>
<th>Source</th>
<th>Target</th>
<th>MENTITY Specified?</th>
<th>Define Profile?</th>
<th>RACF Indicated?</th>
<th>Warning Message?</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>None</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>None</td>
<td>Generic</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>None</td>
<td>RACF Indicated</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Generic</td>
<td>None</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No (W)</td>
</tr>
<tr>
<td>Generic</td>
<td>Generic</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Generic</td>
<td>RACF Indicated</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>RACF Indicated</td>
<td>None</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No (W)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 33. DFSMSdss Copy and Define Profile Summary (continued). This table summarizes when DFSMSdss defines discrete profiles for a copy operation.

<table>
<thead>
<tr>
<th>Protection</th>
<th>Source</th>
<th>Target</th>
<th>MENTITY Specified?</th>
<th>Define Profile?</th>
<th>RACF Indicated?</th>
<th>Warning Message?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACF Indicated</td>
<td>Generic</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No (W)</td>
<td></td>
</tr>
<tr>
<td>RACF Indicated</td>
<td>RACF Indicated</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

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 215.

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 531.
Chapter 21. Data integrity—serialization

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.

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

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 DFSMStvs)” on page 568
- IMS backup-while-open support, see “Backup-while-open data sets (IMS)” on page 573
- DFSMStvs, see z/OS DFSMStvs Planning and Operating Guide and z/OS DFSMStvs 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)
Data Integrity—Serialization

- PRINT (except data set)
- DEFRAG

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.

Note that 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

Related reading: 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 ADRLOCK and a minor name of the volume serial number (padded with blanks to 8 bytes).
- 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 ADRLOCK and a minor name of the volume serial number (padded with blanks 8 bytes). 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 is requested.
Data Integrity—Serialization

- 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 (ADRLOCK, volser). 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 ADRLOCK/volser in a shared ENQ or ADRLOCK/NONSPEC in an exclusive ENQ.

Notes:
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 217.

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.

Enqueueing—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 non-VSAM data sets only.
- For the DUMP and COPY commands, SHARE applies to non-VSAM data sets and 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 enqueuing. 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. 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 34 shows the data set enqueue options.

TOLERATE(ENQFAILURE) can be used on VSAM and non-VSAM data sets.

<table>
<thead>
<tr>
<th>Options</th>
<th>None</th>
<th>SHARE</th>
<th>TOLERATE(ENQFAILURE)</th>
<th>SHARE and TOLERATE(ENQFAILURE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of enqueue attempted</td>
<td>Exclusive</td>
<td>Share</td>
<td>Exclusive</td>
<td>Share</td>
</tr>
<tr>
<td>If enqueue is successful</td>
<td>Process</td>
<td>Process</td>
<td>Process</td>
<td>Process</td>
</tr>
<tr>
<td>If enqueue is not successful</td>
<td>Do not process</td>
<td>Do not process</td>
<td>Process without enqueue</td>
<td>Process without enqueue</td>
</tr>
</tbody>
</table>

### 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.

### Dumping zFS 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 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 dump is not mounted.

If a zFS data set is mounted before you dump it, the data set can still be quiesced. The quiesce action prevents updates to the data set for the duration of the dump.

**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.
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.

DFSMsDs does not honor DYNALLOC for HFS source data sets for either logical data set copy or logical data set dump.

Enqueueing 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 36 on page 566 provides more information on data set serialization.

Read/Write serialization scheme

Table 35 shows the serialization scheme used by all operations, except COPYDUMP, for read/write access.

<table>
<thead>
<tr>
<th>ENQ Names</th>
<th>ENQ Control, Exclusive (E) or Share (S)</th>
<th>ENQ Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Name</td>
<td>Minor Name</td>
<td></td>
</tr>
<tr>
<td>For Source HFS Data Sets:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>followed by:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADRDSN</td>
<td>Data set name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>SYSZDSN</td>
<td>Data set name</td>
<td>S (see Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E (see Note 2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S (see Note 2)</td>
</tr>
<tr>
<td>For z/FS Data Sets:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>On the component during logical data set DUMP or COPY operations:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVSAM</td>
<td>See Note 3</td>
<td>E (see Note 5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S (see Note 5)</td>
</tr>
<tr>
<td>On the component during physical data set DUMP operations:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVSAM</td>
<td>See Note 3</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S (see Note 1)</td>
</tr>
<tr>
<td>On the component during other operations:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVSAM</td>
<td>See Note 3</td>
<td>E</td>
</tr>
</tbody>
</table>
## Data Integrity—Serialization

Table 35. Read/Write Access Serialization Scheme (continued)

<table>
<thead>
<tr>
<th>ENQ Names</th>
<th>ENQ Control, Exclusive (E) or Share (S)</th>
<th>ENQ Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Name</td>
<td>Minor Name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>ADRDSN</td>
<td>Cluster name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>SYSDSN</td>
<td>Cluster name</td>
<td>E (see Note 1)</td>
</tr>
</tbody>
</table>

### For Non-VSAM Data Sets:

<table>
<thead>
<tr>
<th>ENQ Names</th>
<th>ENQ Control, Exclusive (E) or Share (S)</th>
<th>ENQ Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Name</td>
<td>Minor Name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>ADRDSN</td>
<td>Data set name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>SYSDSN</td>
<td>Data set name</td>
<td>E (see Note 1)</td>
</tr>
</tbody>
</table>

### For GDG Data Sets:

<table>
<thead>
<tr>
<th>ENQ Names</th>
<th>ENQ Control, Exclusive (E) or Share (S)</th>
<th>ENQ Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Name</td>
<td>Minor Name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>ADRDSN</td>
<td>GDG base name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>SYSDSN</td>
<td>GDG base name</td>
<td>E (see Note 1)</td>
</tr>
</tbody>
</table>

### For VSAM Data Sets:

<table>
<thead>
<tr>
<th>ENQ Names</th>
<th>ENQ Control, Exclusive (E) or Share (S)</th>
<th>ENQ Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Name</td>
<td>Minor Name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>SYSVSAM</td>
<td>See Note 3</td>
<td>E (see Note 1)</td>
</tr>
</tbody>
</table>

### For Integrated Catalog Facility Catalogs:

<table>
<thead>
<tr>
<th>ENQ Names</th>
<th>ENQ Control, Exclusive (E) or Share (S)</th>
<th>ENQ Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Name</td>
<td>Minor Name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>SYSIGGV2</td>
<td>Catalog name</td>
<td>E (see Note 1)</td>
</tr>
</tbody>
</table>

### For VVDS Data Sets:

<table>
<thead>
<tr>
<th>ENQ Names</th>
<th>ENQ Control, Exclusive (E) or Share (S)</th>
<th>ENQ Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Name</td>
<td>Minor Name</td>
<td>E (see Note 1)</td>
</tr>
<tr>
<td>SYSZVVDS</td>
<td>SYS1.VVDS.Vvolser</td>
<td>E (see Note 1)</td>
</tr>
</tbody>
</table>
## Data Integrity—Serialization

### Table 35. Read/Write Access Serialization Scheme (continued)

<table>
<thead>
<tr>
<th>ENQ Names</th>
<th>ENQ Control, Exclusive (E) or Share (S)</th>
<th>ENQ Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Name</td>
<td>Minor Name</td>
<td></td>
</tr>
</tbody>
</table>

Notes:
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 enqueuing 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 36. Resource Serialization

<table>
<thead>
<tr>
<th>Function</th>
<th>Data Set Type</th>
<th>Volume Level Serialization</th>
<th>Data Set Level Serialization (ENQ or DYNALLOC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>VTOC</td>
<td>VVDS</td>
</tr>
<tr>
<td>COMPRESS</td>
<td>Non-VSAM</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>CONSOLIDATE</td>
<td>Non-VSAM</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>VSAM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CONVERTV</td>
<td>Non-VSAM</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Data Set COPY</td>
<td>Non-VSAM</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>VSAM</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
**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).
Data Integrity—Serialization

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
Data Integrity—Serialization

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, DFSMSdfp, 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, DFSMSdfp, and DFSMSdss.

Although the BWO(TYPECICS) parameter applies to both CICS and DFSMStvs, DFSMStvs enables you to back up a data set while they are open whether or not you are running CICS.

Figure 23 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 23. Block Diagram for Backup-While-Open Serialization

In Figure 23 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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning to DFSMSdss</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Normal serialization and processing techniques are used to dump the data set.</td>
</tr>
<tr>
<td>001</td>
<td>CICSVR forward-recover the data set.</td>
</tr>
<tr>
<td>010</td>
<td>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.</td>
</tr>
<tr>
<td>011</td>
<td>A control-interval split, control-area split, or an extend of the data set completed successfully. CICS or DFSMStvs closed the data set and there is no mismatch between the base cluster and any alternate index.</td>
</tr>
<tr>
<td>100</td>
<td>The data set may be dumped while it is open for update.</td>
</tr>
<tr>
<td>101</td>
<td>The data set was restored and is down-level; CICSVR or DFSMStvs must process it before a database application uses it.</td>
</tr>
<tr>
<td>110</td>
<td>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.</td>
</tr>
<tr>
<td>111</td>
<td>The data set is in an indeterminate state.</td>
</tr>
</tbody>
</table>

### 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.
- 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 DFSMStvs 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. 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.
**Backup-while-open data sets (IMS)**

DFSMStvs 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 22, “Application programming interface,” on page 575.

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 37.

**Note:** During a logical data set dump, you must specify VALIDATE (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.

DFSMStvs 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).

**Table 37. DFSMSdss Support for IMS Backup-While-Open Data Sets**

<table>
<thead>
<tr>
<th>Topic</th>
<th>For IMS data sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can non-VSAM data sets be copied or dumped as open data sets?</td>
<td>Yes</td>
</tr>
<tr>
<td>Can non-indexed VSAM data sets be copied or dumped as open data sets?</td>
<td>Yes</td>
</tr>
<tr>
<td>Must indexed VSAM data sets be specifically designated as being BWO-eligible to be copied or dumped as open data sets?</td>
<td>No</td>
</tr>
<tr>
<td>How are indexed VSAM data sets designated as being eligible for update activity detection through a BWO counter?</td>
<td>By defining the data set as BWO(TYPEIMS)</td>
</tr>
<tr>
<td>How is backup as an open data set triggered?</td>
<td>Through a UIM request</td>
</tr>
<tr>
<td>How is bypassing of enqueues on SYSDSN and SYSVSAM managed?</td>
<td>Automatically</td>
</tr>
</tbody>
</table>
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.
Chapter 22. 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 24 on page 578.

**ADRDSSU**
The name of the DFSMSdss module main entry point.

**OPTPTR**
The pointer to the option list and similar to OPTIONADDR described in z/OS DFSMSdfp 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 258. 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 581.

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 DFSMSdfp 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 DFSMSdfp Utilities](../index.html). 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 DFSMSdfp Utilities](../index.html). 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 582 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 DFSMSdfp Utilities](https://www.ibm.com/servers/dfs/dfsmsdfp/). 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 DFSMSdfp Utilities](https://www.ibm.com/servers/dfs/dfsmsdfp/). 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.
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=ADROSSU, PARAM=(OPTPTR, DDPTR, PAGEPTR, UIMPTR, UAPTR), VL=1
LINK EP=ADROSSU, 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=ADRMAIA, PARAM=(OPTPTR, DDPTR, PAGEPTR, UIMPTR, UAPTR), VL=1
LINK EP=ADRMAIA, 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=ADRMAIA, PARAM=(OPTPTR, DDPTR, PAGEPTR, UIMPTR, UAPTR, ASNPTR), VL=1
LINK EP=ADRMAIA, PARAM=(OPTPTR, DDPTR, PAGEPTR, UIMPTR, UAPTR, ASNPTR), VL=1
CALL (15), (OPTPTR, DDPTR, PAGEPTR, UIMPTR, UAPTR, ASNPTR), VL
```
As shown in Figure 24 on page 578, 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.

Notes:
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 SRVRTIME 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 23, “Examples of the application program with the user interaction module (UIM),” on page 619.

Cross-memory Application Interface overview

The DFSMSdsss 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 DFSMShsm. DFSMSdss also provides a JCL interface for this support.
DFSMsDSS Cross Memory support is designed to be invoked using the LINK, CALL, or ATTACH macros. The support may 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 may be created by issuing a START command or may 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 may 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 will be connecting to the server that is created.

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.

**Restrictions:** 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 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\textit{id}

Where \textit{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.

NOTE: 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)'

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, UFASYN and UFASYSPR specify this to DFSMSdss.

If the application allows DFSMSdss to handle SYSPRINT by not setting UFASYSPR, then the application can not 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 UFASYSPR, 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: UFAINV, UFUIMAL and UFUIMCH. Two additional bits indicate the existence or absence of
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 604 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" below 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 24 on page 578 can be viewed in block form as shown in Figure 25.

```
REG 1 ─── Interface Pointer ─── Length
      | Exit Identification Block |
```

*Figure 25. 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 604 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
Application Interface

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.

Notes:

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 (ADRUFO) 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 595 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.
Application Interface

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 SMSGCNT in place of ABEND and AMSGCNT when the Cross-Memory Application Interface is used. Although ABEND and AMSGCNT 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) 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, GA22-7800.

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 EIRECPTTR 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 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 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 EIRECPTTR 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 EIRECPTTR 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 582).

The valid return codes for printing SYSPRINT records are:
Application Interface

0  Continue normal processing
4  Record replaced
8  Insert record
12 Delete record
16 Record modified
20 Disconnect exit

Notes:
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 197.

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
Application Interface

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). 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 IECOEVSE 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.
Application Interface

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 IECOENTE 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](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 EIDDINFO. 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.

EIRECPTR 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 Table 38 on page 604. 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 Table 38 on page 604. 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 (EI22LINF) is set on by the UIM during a logical restore, then DFSMSdss uses the log parameter (EI22LPRM) and the log stream ID (EI22LSID) as the log information for the target data set, provided that the target data set is SMS-managed. If EI22LINF is not set, or the target data set is not SMS-managed, then the contents of EI22LPRM and EI22LSID 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 the Bypass Serialization indicator, 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. 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 UNCAT keywords are specified. If a preallocated data set is not large enough during a data set restore operation and the bypass serialization indicator 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.
**Application Interface**

- 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 cannot allocate the target data set on a device whose unit type matches that of the source, the data set is not processed.

- Serialization by enqueuing 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.

20 Disconnect exit
32 End function
36 End data set

Refer to EIRC36 in [“Data set verification (Eioption 21)” on page 595](#) for details on ending spheres.

**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 [Table 38 on page 604](#).

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)
Application Interface

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

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 595 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.
Application Interface

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" on page 584.

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 EIRECPTR 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. (DFMSdss sets EI26VOL with EI26DDN and EI26VSER.)
- When a dump output volume associated with a specific DDNAME receives a terminating error condition. (DFMSdss 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. (DFMSdss 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...
Application Interface

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 604.

- The data set name.
- The new name of the data set.
- 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.
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 604. 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)

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 parameter 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.

Avoiding lockout

Refer to "Avoiding lockout" on page 560 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. Table 38 on page 604 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 595 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.

### ADREID0 data area

Table 38 describes the data area corresponding to the DFSMSdss exit identification block.

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>42</td>
<td>ADREIB</td>
<td>LENGTH OF ADREIB - 2</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>2</td>
<td>EIDLLEN</td>
<td>BLOCK IDENTIFIER EBCDIC &quot;EIDB&quot;</td>
</tr>
<tr>
<td>2</td>
<td>(2)</td>
<td>4</td>
<td>EID</td>
<td>TASK ID NUMBER</td>
</tr>
<tr>
<td>6</td>
<td>(6)</td>
<td>4</td>
<td>EITSKID</td>
<td>ALLOWANCE OPTION BYTE 1</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>4</td>
<td>EIXALLOW</td>
<td>ALLOW REPLACE OF RECORD</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>1</td>
<td>EIXALLOW0</td>
<td>ALLOW INSERTION OF RECORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1..</td>
<td>EIXREP</td>
<td>ALLOW DELETION OF RECORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...1.</td>
<td>EIXINS</td>
<td>ALLOW MODIFICATION OF RECORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>....</td>
<td>EIXDEL</td>
<td>ALLOW DISCONNECT OF EXIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...1.</td>
<td>EIXMOD</td>
<td>ALLOW WTOR RESPONSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...1.</td>
<td>EIXDIS</td>
<td>ALLOW SELECTION OF USER STATS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...1.</td>
<td>EIXWTOR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...1.</td>
<td>EIXSTAT</td>
<td></td>
</tr>
</tbody>
</table>
### Table 38. ADREIDO Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 (B)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EIXTERM</td>
<td>ALLOW FUNCTION TERMINATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXALOW1</td>
<td>ALLOWANCE OPTION BYTE 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXDSET</td>
<td>ALLOW DATA SET TERMINATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>UNUSED</td>
</tr>
<tr>
<td>12 (C)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EIXALOW2</td>
<td>ALLOWANCE OPTION BYTE 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>13 (D)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EIXALOW3</td>
<td>ALLOWANCE OPTION BYTE 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>14 (E)</td>
<td>SIGNED</td>
<td>2</td>
<td>EIOPTION</td>
<td>PROCESSING OPTION</td>
</tr>
<tr>
<td>16 (10)</td>
<td>SIGNED</td>
<td>2</td>
<td>EIRETCOD</td>
<td>EXIT RETURN CODE</td>
</tr>
<tr>
<td>18 (12)</td>
<td>SIGNED</td>
<td>4</td>
<td>EIRECALN</td>
<td>RECORD AREA LENGTH</td>
</tr>
<tr>
<td>22 (16)</td>
<td>SIGNED</td>
<td>4</td>
<td>EIRECLEN</td>
<td>ORIGINAL RECORD LENGTH</td>
</tr>
<tr>
<td>26 (1A)</td>
<td>ADDRESS</td>
<td>4</td>
<td>EIRECPTR</td>
<td>ORIGINAL RECORD ADDRESS</td>
</tr>
<tr>
<td>30 (1E)</td>
<td>ADDRESS</td>
<td>4</td>
<td>EIUSEPTR</td>
<td>USER DATA AREA ADDRESS</td>
</tr>
<tr>
<td>34 (22)</td>
<td>ADDRESS</td>
<td>4</td>
<td>EIDDID</td>
<td>EIOp06 DDNAME/VOLID PTR</td>
</tr>
<tr>
<td>38 (26)</td>
<td>BITSTRING</td>
<td>4</td>
<td>EIXFLAGS</td>
<td>OTHER FLAGS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXABEND</td>
<td>FOR EIOp02 ONLY, 1=MESSAGE IS ADR013 INDICATING AN ABEND CONDITION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXNTERR</td>
<td>FOR EIOp02 ONLY, 1=MESSAGE IS TYPE 'E' AND IS NOT 324 OR 347</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXWNGOK</td>
<td>FOR EIOp14 ONLY, 1=WARNING MSGS WERE ISSUED AND NONE ARE FATAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXTRKER</td>
<td>TRACK IS IN ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>39 (27)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EIXFLAG1</td>
<td>FLAG BYTE 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXTDM5</td>
<td>DUMMY SOURCE TAPE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXTDMT</td>
<td>DUMMY TARGET TAPE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXTISXM</td>
<td>APL USING XMAPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EIXDAYS</td>
<td>XMAPI ALLOCATION ERR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>40 (28)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EIXFLAG2</td>
<td>FLAG BYTE 2</td>
</tr>
<tr>
<td>41 (29)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EIXFLAG3</td>
<td>FLAG BYTE 3</td>
</tr>
<tr>
<td>42 (2A)</td>
<td>CHARACTER</td>
<td>*</td>
<td></td>
<td>FORCE SIZE OF CONTROL BLOCK TO WORD BOUNDARY</td>
</tr>
</tbody>
</table>

**Note:** THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 6 AND POINTED TO BY EIDDID.
### Table 38. ADREID0 Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>16</td>
<td>EIREC00</td>
<td>SHARED SUBPOOL NUMBER</td>
</tr>
<tr>
<td>0 (0)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>EI00SBL</td>
<td>STARTUP FLAGS</td>
</tr>
<tr>
<td>1 (1)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EI00FLGS</td>
<td>SHORT VTOC ENQUEUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00SENQ</td>
<td>DO NOT ALLOW IGWNOTIF CALL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00NONF</td>
<td>DO NOT PERFORM ADRLOCK ENQUEUE OR DEQUEUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00NOLK</td>
<td>ENQUEUE OR DEQUEUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00NENQ</td>
<td>NO INPUT VOLUME SERIALIZATION. VALID DURING FUNCTION STARTUPS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00BSEC</td>
<td>BYPASS SECURITY VERIFICATION FOR DATA SETS ON THE VOLUME</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00MVOLRECOV</td>
<td>=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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00BSER</td>
<td>1=BYPASS Serialization of the data sets being restored during a physical data set restore operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00BSER</td>
<td>1=BYPASS SERIALIZATION OF THE DATA SETS BEING RESTORED DURING A PHYSICAL DATA SET RESTORE OPERATION</td>
</tr>
<tr>
<td>2 (2)</td>
<td>CHARACTER</td>
<td>6</td>
<td>EI00SVOLORECOV</td>
<td>SOURCE VOLSER. VALID DURING COPY FULL/TRACKS, DUMP FULL/TRACKS, AND RESTORE FULL/TRACKS FUNCTION STARTUPS. FOR RESTORE, THIS FIELD CONTAINS THE VOLSER OF THE INPUT VOLUME WHERE THE DUMP DATA SET RESIDES</td>
</tr>
<tr>
<td>8 (8)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EI00FLG2</td>
<td>SECOND FLAG AREA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00SWNCRYPT</td>
<td>SOFTWARE ENCRYPTION WAS REQUESTED (RSA OR KEYPASSWORD WAS SPECIFIED)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00CANSFE</td>
<td>CANCEL SOFTWARE ENCRYPTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI00MIXDEV</td>
<td>MIXTURE OF DEVICES; SOME ARE PERFORMING HARDWARE ENCRYPTION AND OTHERS ARE NOT.</td>
</tr>
<tr>
<td>9 (9)</td>
<td>CHARACTER</td>
<td>7</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>

**Note:** The following maps the record presented by Exit 00 (Function Startup).

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>16</td>
<td>EIREC14</td>
<td>DSS RETURN CODE</td>
</tr>
<tr>
<td>0 (0)</td>
<td>CHARACTER</td>
<td>4</td>
<td>EI14RC</td>
<td>DSS MESSAGE</td>
</tr>
<tr>
<td>4 (4)</td>
<td>CHARACTER</td>
<td>4</td>
<td>EI14MESS</td>
<td>DSS MESSAGE</td>
</tr>
<tr>
<td>4 (4)</td>
<td>CHARACTER</td>
<td>3</td>
<td>EI14MNUM</td>
<td>MESSAGE NUMBER</td>
</tr>
<tr>
<td>7 (7)</td>
<td>CHARACTER</td>
<td>1</td>
<td>EI14MTYP</td>
<td>MESSAGE TYPE</td>
</tr>
<tr>
<td>8 (8)</td>
<td>CHARACTER</td>
<td>8</td>
<td>EI14CPUT</td>
<td>DSS CPU TIME</td>
</tr>
</tbody>
</table>
### Table 38. ADREID0 Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>56</td>
<td>EIREC20</td>
<td>DATA SET NAME/CLUSTER NAME</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>44</td>
<td>EI20DSN</td>
<td>SOME FLAGS:</td>
</tr>
<tr>
<td>44</td>
<td>BITSTRING</td>
<td>1</td>
<td>EI20FLGS</td>
<td>1=DATA SET IS VSAM 0=DATA SET IS NONVSAM</td>
</tr>
<tr>
<td></td>
<td>..</td>
<td>...</td>
<td>EI20VSAM</td>
<td>1=DATA SET IS PROTECTED BY A DISCRETE RACF PROFILE</td>
</tr>
<tr>
<td></td>
<td>..</td>
<td>...</td>
<td>EI20RACF</td>
<td>1=LAST VOLUME INDICATOR ON FOR NON-VSAM DS</td>
</tr>
<tr>
<td>45</td>
<td>CHARACTER</td>
<td>1</td>
<td>EI20DA#</td>
<td>NUMBER OF DATA COMPONENTS</td>
</tr>
<tr>
<td>46</td>
<td>UNSIGNED</td>
<td>1</td>
<td>EI20IX#</td>
<td>NUMBER OF INDEX COMPONENTS (0 IF NONVSAM)</td>
</tr>
<tr>
<td>48</td>
<td>ADDRESS</td>
<td>4</td>
<td>EI20DA@</td>
<td>POINTER TO DATA COMPONENT INFO FOR VSAM/DATA SET INFO FOR NONVSAM</td>
</tr>
<tr>
<td>52</td>
<td>ADDRESS</td>
<td>4</td>
<td>EI20IX@</td>
<td>POINTER TO INDEX COMPONENT INFO</td>
</tr>
<tr>
<td>56</td>
<td>SIGNED</td>
<td>2</td>
<td>*</td>
<td>PADDING (UNUSED) THE EI20SCNT, EI20DSHA, AND EI20ISHA FIELDS ARE PROVIDED WHEN PROCESSING THE FIRST VOLUME OF A MULTI VOLUME DATA SET</td>
</tr>
<tr>
<td>58</td>
<td>SIGNED</td>
<td>2</td>
<td>EI20SCNT</td>
<td>TOTAL STRIPE COUNT FOR THIS VSAM DATA SET</td>
</tr>
<tr>
<td>60</td>
<td>UNSIGNED</td>
<td>4</td>
<td>EI20DSHA</td>
<td>VSAM DATA COMPONENT HIGH ALLOCATED RBA CNT</td>
</tr>
<tr>
<td>64</td>
<td>UNSIGNED</td>
<td>4</td>
<td>EI20ISHA</td>
<td>VSAM INDEX COMPONENT HIGH ALLOCATED RBA CNT</td>
</tr>
<tr>
<td>0</td>
<td>STRUCTURE</td>
<td>68</td>
<td>EI20DSI</td>
<td>DATA SET INFO</td>
</tr>
<tr>
<td>0</td>
<td>CHARACTER</td>
<td>44</td>
<td>EI20CON</td>
<td>COMPONENT NAME (BLANKS FOR NONVSAM)</td>
</tr>
<tr>
<td>44</td>
<td>UNSIGNED</td>
<td>2</td>
<td>EI20NVOL</td>
<td># OF VOLUMES IN DS</td>
</tr>
<tr>
<td>46</td>
<td>CHARACTER</td>
<td>2</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>48</td>
<td>CHARACTER</td>
<td>20</td>
<td>EI20VLI</td>
<td>VOLUME INFORMATION</td>
</tr>
<tr>
<td>48</td>
<td>CHARACTER</td>
<td>6</td>
<td>EI20VOL</td>
<td>VOLSER</td>
</tr>
<tr>
<td>54</td>
<td>SIGNED</td>
<td>2</td>
<td>EI20DSTP#</td>
<td>STRIPE NUMBER FOR THE DATA COMPONENT. 0 FOR NON-STRIPED DATA SETS</td>
</tr>
<tr>
<td>56</td>
<td>UNSIGNED</td>
<td>4</td>
<td>EI20RBA</td>
<td>RBA TOKEN. THE EI20DRBA FIELD IS CALCULATED BY SUBTRACTING THE HIGH RBA FIELDS FROM THE LOW RBA FLDS.</td>
</tr>
<tr>
<td>60</td>
<td>UNSIGNED</td>
<td>4</td>
<td>EI20DRBA</td>
<td>TOTAL NUMBER OF RBAS PROCESSED FOR THIS COMPONENT</td>
</tr>
<tr>
<td>62</td>
<td>SIGNED</td>
<td>2</td>
<td>EI20VLSQ</td>
<td>VOLUME SEQUENCE NUMBER FOR THE DATA SET</td>
</tr>
<tr>
<td>64</td>
<td>SIGNED</td>
<td>2</td>
<td>*</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>

**Note:** THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 20: VOLUME NOTIFICATION EXIT.
## Application Interface

### Table 38. ADREID0 Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (0)</td>
<td>STRUCTURE</td>
<td>44</td>
<td>EIREC21</td>
<td>DATA SET/CLUSTER NAME</td>
</tr>
<tr>
<td>0 (0)</td>
<td>CHARACTER</td>
<td>44</td>
<td>EI21DSN</td>
<td></td>
</tr>
</tbody>
</table>

Note: THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 22: BYPASS VERIFICATION EXIT.

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>44 (2C)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EI22FLGS</td>
<td>EXIT 22 FLAGS: 1=BYPASS SERIALIZATION, 1=BYPASS DATASET LEVEL, STORCLAS &amp; MGMTCLAS SECURITY CHECKS. ALSO BYPASS JES3 INTEGRITY CHECKS AND DO NOT CREATE DISCRETE DATASET PROFILES.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22SER</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22SEC</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>44 (2D)</td>
<td>CHARACTER</td>
<td>5</td>
<td>EI22SFLG</td>
<td>BYPASS SOURCE FOR DB2 RENAME</td>
</tr>
<tr>
<td>45 (2D)</td>
<td>CHARACTER</td>
<td>1</td>
<td>EI22SMSF</td>
<td>SOURCE DATA SET FLGS</td>
</tr>
<tr>
<td>46 (2E)</td>
<td>CHARACTER</td>
<td>2</td>
<td>EI22SDSG</td>
<td>SOURCE DATA SET ORG</td>
</tr>
<tr>
<td>48 (30)</td>
<td>BITSTRING</td>
<td>2</td>
<td>EI22SVFLG</td>
<td>SRCVE VSAM DSET FLGS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22SESRS</td>
<td>1=ESDS DATA SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22KSRS</td>
<td>1=KSDS DATA SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22KRDSS</td>
<td>1=KRDSD DATA SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22LSRS</td>
<td>1=LINEAR DATA SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22SRDSS</td>
<td>1=RRDS DATA SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22SPSSI</td>
<td>1=PAGE/SWAP/STGINDEX, ETC. UNSUPPORTED DATA SET</td>
</tr>
<tr>
<td>49 (31)</td>
<td></td>
<td>1</td>
<td>EI22SAIX</td>
<td>1=AIX DATA SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26</td>
<td>EI22SVD</td>
<td>VRRDS DATASET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22SSCS</td>
<td>1=BCS DATA SET</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 (32)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EI22FLG2</td>
<td>MORE EXIT 22 FLAGS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22RSET</td>
<td>RESET DS CHANGED FLAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22SYS</td>
<td>GET SHARED SYSDSN ENQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22RRB</td>
<td>MARK TGT RECOVERY REQ'D</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22LINF</td>
<td>LOG INFO PASSED, SEE EI22LPDM AND EI22LSD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22BWOP</td>
<td>BWO_ALLOWED PASSED, SEE EI22BWOA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22MS</td>
<td>CALLER IS IMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EI22BWOE</td>
<td>ENQ ON BWODSN ONLY, NO ADR OR SYS ENQ'S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>51 (33)</td>
<td>CHARACTER</td>
<td>1</td>
<td>EI22LPDM</td>
<td>LOG PARAMETER FOR TGT</td>
</tr>
<tr>
<td>52 (34)</td>
<td>CHARACTER</td>
<td>26</td>
<td>EI22LSD</td>
<td>LOG STREAM ID FOR TGT</td>
</tr>
<tr>
<td>78</td>
<td>BITSTRING</td>
<td>1</td>
<td>EI22FLG3</td>
<td>MORE EXIT 22 FLAGS</td>
</tr>
</tbody>
</table>
### Table 38. ADREID0 Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111....</td>
<td>1111..</td>
<td>1111....</td>
<td>EI22BWOA</td>
<td>BWO_ALLOWED</td>
</tr>
<tr>
<td>...1...</td>
<td>...1...</td>
<td>...1...</td>
<td>EI22BRLS</td>
<td>BYPASS RLS QUIESCE</td>
</tr>
<tr>
<td>...1...</td>
<td>...1...</td>
<td>...1...</td>
<td>EI22SFSM</td>
<td>SET RECONNECTABLE FLAG</td>
</tr>
<tr>
<td>....11..</td>
<td>....11..</td>
<td>....11..</td>
<td>*</td>
<td>UNUSED</td>
</tr>
<tr>
<td>79</td>
<td>(4F)</td>
<td>CHARACTER</td>
<td>23</td>
<td>AVAILABLE</td>
</tr>
</tbody>
</table>

**Note:** THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 23: DATA SET PROCESSED NOTIFICATION EXIT.

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0)</td>
<td>STRUCTURE</td>
<td>*</td>
<td>EIREC23</td>
<td>CONSTANT LENGTH PORTION OF CONTROL BLK</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>CHARACTER</td>
<td>141</td>
<td>EI23CNST</td>
<td>DATA SET/CLUSTER NAME</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>CHARACTER</td>
<td>44</td>
<td>EI23DSN</td>
<td>NEW DSET/CLUSTER NAME</td>
</tr>
<tr>
<td>44</td>
<td>(2C)</td>
<td>CHARACTER</td>
<td>44</td>
<td>EI23NEWN</td>
<td>RETURN CODE FOR DATA SET PROCESSING</td>
</tr>
<tr>
<td>88</td>
<td>(58)</td>
<td>SIGNED</td>
<td>2</td>
<td>EI23DSRC</td>
<td>SOURCE DATA SET FLAGS</td>
</tr>
<tr>
<td>90</td>
<td>(5A)</td>
<td>CHARACTER</td>
<td>5</td>
<td>EI23SFLG</td>
<td>SOURCE DATA SET ORG</td>
</tr>
<tr>
<td>90</td>
<td>(5A)</td>
<td>CHARACTER</td>
<td>1</td>
<td>EI23SMSF</td>
<td>DATA SET/CLUSTER NAME</td>
</tr>
<tr>
<td>91</td>
<td>(5B)</td>
<td>CHARACTER</td>
<td>2</td>
<td>EI23DSRG</td>
<td>SOURCE DATA SET ORG</td>
</tr>
<tr>
<td>93</td>
<td>(5D)</td>
<td>BITSTRING</td>
<td>2</td>
<td>EI23VFLG</td>
<td>SOURCE VSAM DSET FLAGS</td>
</tr>
<tr>
<td>94</td>
<td>(5E)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EI23VDS</td>
<td>SUPPORTED DATA SETS</td>
</tr>
<tr>
<td>95</td>
<td>(5F)</td>
<td>CHARACTER</td>
<td>5</td>
<td>EI23FLG</td>
<td>TARGET DATA SET ORG</td>
</tr>
<tr>
<td>95</td>
<td>(5F)</td>
<td>CHARACTER</td>
<td>1</td>
<td>EI23SMSF</td>
<td>TARGET DATA SET ORG</td>
</tr>
<tr>
<td>96</td>
<td>(60)</td>
<td>CHARACTER</td>
<td>2</td>
<td>EI23DSRG</td>
<td>TARGET DATA SET ORG</td>
</tr>
<tr>
<td>98</td>
<td>(62)</td>
<td>BITSTRING</td>
<td>2</td>
<td>EI23VFLG</td>
<td>TARGET VSAM DSET FLAGS</td>
</tr>
<tr>
<td>99</td>
<td>(63)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EI23TFLG</td>
<td>TARGET VSAM DSET FLAGS</td>
</tr>
<tr>
<td>100</td>
<td>(64)</td>
<td>CHARACTER</td>
<td>16</td>
<td>EI23RLST</td>
<td>RESERVED</td>
</tr>
<tr>
<td>100</td>
<td>(64)</td>
<td>CHARACTER</td>
<td>8</td>
<td>EI23GMT</td>
<td>RLS GMT TIME STAMP</td>
</tr>
<tr>
<td>108</td>
<td>(6C)</td>
<td>CHARACTER</td>
<td>8</td>
<td>EI23LOC</td>
<td>RLS LOCAL TIME STAMP</td>
</tr>
<tr>
<td>116</td>
<td>(74)</td>
<td>CHARACTER</td>
<td>8</td>
<td>EI23BYTES</td>
<td>D8 BYTE COUNT</td>
</tr>
<tr>
<td>124</td>
<td>(7C)</td>
<td>CHARACTER</td>
<td>1</td>
<td>EI23FLGS</td>
<td>MISC FLAGS</td>
</tr>
</tbody>
</table>

**Note:** THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 23: DATA SET PROCESSED NOTIFICATION EXIT.
### Table 38. ADREID0 Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>(7D)</td>
<td>CHARACTER</td>
<td>6</td>
<td>EI23BPSE</td>
<td>E23BYTES IS FOR PSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>131</td>
<td>(83)</td>
<td>CHARACTER</td>
<td>9</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>140</td>
<td>(8C)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>EI23VOL#</td>
<td>NUMBER OF VOLUMES</td>
</tr>
<tr>
<td>141</td>
<td>(8D)</td>
<td>CHARACTER</td>
<td>6</td>
<td>EI23VSER(*)</td>
<td>VOLSER ARRAY</td>
</tr>
</tbody>
</table>

**Note:** THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 24: CONCURRENT COPY INITIALIZATION COMPLETE.

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0 )</td>
<td>STRUCTURE</td>
<td>83</td>
<td>EIREC24</td>
<td>RETURN CODE</td>
</tr>
<tr>
<td>0</td>
<td>(0 )</td>
<td>UNSIGNED</td>
<td>2</td>
<td>EI24RTCD</td>
<td>REASON CODE</td>
</tr>
<tr>
<td>2</td>
<td>(2 )</td>
<td>UNSIGNED</td>
<td>2</td>
<td>EI24RSCD</td>
<td>REASON CODE</td>
</tr>
<tr>
<td>4</td>
<td>(4 )</td>
<td>UNSIGNED</td>
<td>4</td>
<td>*</td>
<td>RESERVED</td>
</tr>
<tr>
<td>8</td>
<td>(8 )</td>
<td>CHARACTER</td>
<td>6</td>
<td>EI24VOL</td>
<td>DATA SET NAME</td>
</tr>
<tr>
<td>14</td>
<td>(E )</td>
<td>CHARACTER</td>
<td>64</td>
<td>EI24DSN</td>
<td>DATA SET NAME</td>
</tr>
<tr>
<td>58</td>
<td>(3A)</td>
<td>BITSTRING</td>
<td>1</td>
<td>EI24FLGS</td>
<td>FLAGS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E24RSET</td>
<td>DS CHANGE FLAG RESET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E24SAIX</td>
<td>1=AIX DATA SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E24BCSEL</td>
<td>1=BASE CLUSTER SELECTED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>UNUSED</td>
</tr>
<tr>
<td>59</td>
<td>(3B)</td>
<td>CHARACTER</td>
<td>24</td>
<td>*</td>
<td>AVAILABLE</td>
</tr>
</tbody>
</table>

**Note:** THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 26: DUMP OUTPUT VOLUME MOUNT NOTIFICATION EXIT.

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0 )</td>
<td>STRUCTURE</td>
<td>128</td>
<td>EIREC26</td>
<td>EXIT TYPE</td>
</tr>
<tr>
<td>0</td>
<td>(0 )</td>
<td>BITSTRING</td>
<td>4</td>
<td>EI26TYPE</td>
<td>EXIT TYPE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EI26VOL</td>
<td>OUTPUT VOLUME NOTIFICATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EI26TERM</td>
<td>OUTPUT VOLUME TERMINATED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EI26ROCE</td>
<td>BWO RO COUNT ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EI26VCLO</td>
<td>OUTPUT VOLUME CLOSE - ONLY FOR DASD OUTPUTS DURING LOGICAL DUMP OPERATIONS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EI26TWHE</td>
<td>WHEN SET TO 1, TAPE DEVICE WILL BE ENCRYPTING DATA IN HARDWARE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>UNUSED</td>
</tr>
<tr>
<td>0</td>
<td>(0 )</td>
<td>BITSTRING</td>
<td>3</td>
<td>*</td>
<td>RESERVED FOR EXPANSION</td>
</tr>
<tr>
<td>4</td>
<td>(4 )</td>
<td>CHARACTER</td>
<td>64</td>
<td>EI26DSN</td>
<td>DSNAME IF EI26ROCE = '1'B</td>
</tr>
<tr>
<td>68</td>
<td>(44)</td>
<td>UNSIGNED</td>
<td>1</td>
<td>EI26DSNL</td>
<td>LENGTH OF DSNAME</td>
</tr>
<tr>
<td>69</td>
<td>(45)</td>
<td>UNSIGNED</td>
<td>3</td>
<td>*</td>
<td>RESERVED FOR ALIGNMENT</td>
</tr>
<tr>
<td>72</td>
<td>(48)</td>
<td>CHARACTER</td>
<td>8</td>
<td>EI26DDN</td>
<td>OUTPUT DDNAME IF EI26VOL, EI26TERM, OR EI26VCLO SET</td>
</tr>
<tr>
<td>80</td>
<td>(50)</td>
<td>CHARACTER</td>
<td>6</td>
<td>EI26VSR</td>
<td>VOLSER - PRESENT IF EI26VOL, EI26TERM, EI26VCLO SET</td>
</tr>
<tr>
<td>86</td>
<td>(56)</td>
<td>CHARACTER</td>
<td>2</td>
<td>*</td>
<td>RESERVED FOR ALIGNMENT</td>
</tr>
<tr>
<td>88</td>
<td>(58)</td>
<td>UNSIGNED</td>
<td>4</td>
<td>EI26VTRC</td>
<td>RETURN FOR VOLUME TERM AND VOLUME CLOSE</td>
</tr>
<tr>
<td>92</td>
<td>(5C)</td>
<td>ADDRESS</td>
<td>36</td>
<td>*</td>
<td>RESERVED FOR EXPANSION</td>
</tr>
</tbody>
</table>

**Note:** THE FOLLOWING MAPS THE RECORD PRESENTED IN EXIT 27: PHYSICAL DATA SET PROCESSED NOTIFICATION EXIT.

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0 )</td>
<td>STRUCTURE</td>
<td>168</td>
<td>EIREC27</td>
<td>DATA SET/CLUSTER NAME</td>
</tr>
<tr>
<td>0</td>
<td>(0 )</td>
<td>CHARACTER</td>
<td>44</td>
<td>EI27DSN</td>
<td>NEW DATA SET/CLUSTER NAME</td>
</tr>
<tr>
<td>44</td>
<td>(2C)</td>
<td>CHARACTER</td>
<td>44</td>
<td>EI27NEWN</td>
<td>NEW DATA SET/CLUSTER NAME</td>
</tr>
<tr>
<td>88</td>
<td>(58)</td>
<td>SIGNED</td>
<td>2</td>
<td>EI27DSRC</td>
<td>RETURN CODE FOR DATA SET PROCESSING</td>
</tr>
</tbody>
</table>
### Table 38. ADREID0 Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>(5A)</td>
<td>BITSTRING</td>
<td>5</td>
<td>EI27SFLG</td>
<td>SOURCE DATA SET FLAGS</td>
</tr>
<tr>
<td>90</td>
<td>(5A)</td>
<td>CHARACTER</td>
<td>1</td>
<td>E27SSMF</td>
<td>SOURCE SMS FLAGS</td>
</tr>
<tr>
<td>91</td>
<td>(5B)</td>
<td>CHARACTER</td>
<td>2</td>
<td>E27SDSRC</td>
<td>SOURCE DATA SET ORG</td>
</tr>
<tr>
<td>93</td>
<td>(5D)</td>
<td>CHARACTER</td>
<td>2</td>
<td>E27SVFLG</td>
<td>VSAM DATA SET FLAGS: 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</td>
</tr>
<tr>
<td>94</td>
<td>(5E)</td>
<td>BITSTRING</td>
<td>5</td>
<td>EI27TFLG</td>
<td>TARGET DATA SET FLAGS</td>
</tr>
<tr>
<td>95</td>
<td>(5F)</td>
<td>CHARACTER</td>
<td>1</td>
<td>E27TSMSSF</td>
<td>TARGET SMS FLAGS</td>
</tr>
<tr>
<td>96</td>
<td>(60)</td>
<td>CHARACTER</td>
<td>2</td>
<td>E27TDSRG</td>
<td>TARGET DATA SET ORG</td>
</tr>
<tr>
<td>98</td>
<td>(62)</td>
<td>CHARACTER</td>
<td>2</td>
<td>E27TVFLG</td>
<td>VSAM DATA SET FLAGS: 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</td>
</tr>
<tr>
<td>99</td>
<td>(63)</td>
<td>BITSTRING</td>
<td>5</td>
<td>EI27TFLG</td>
<td>TARGET DATA SET FLAGS</td>
</tr>
<tr>
<td>100</td>
<td>(64)</td>
<td>CHARACTER</td>
<td>8</td>
<td>EI27TRKS</td>
<td>TOTAL TRACKS COPIED</td>
</tr>
<tr>
<td>108</td>
<td>(6C)</td>
<td>CHARACTER</td>
<td>6</td>
<td>EI27TVOL</td>
<td>SOURCE VOLUME SERIAL</td>
</tr>
<tr>
<td>114</td>
<td>(72)</td>
<td>CHARACTER</td>
<td>6</td>
<td>EI27TVOL</td>
<td>TARGET VOLUME SERIAL</td>
</tr>
</tbody>
</table>
### 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

#### Constants

<table>
<thead>
<tr>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E27FLGS</td>
<td>MISC FLAGS</td>
</tr>
<tr>
<td>E27LVOL</td>
<td>LAST VOLUME INDICATOR FROM THE SOURCE DATA SET</td>
</tr>
<tr>
<td>E27VLSQ</td>
<td>VOLUME SEQUENCE NUMBER FOR SOURCE DATA SET</td>
</tr>
<tr>
<td>E27DSHA</td>
<td>VSAM DATA COMPONENT ALLOCATED RBA COUNT</td>
</tr>
<tr>
<td>E27ISHA</td>
<td>VSAM INDEX COMPONENT HIGH ALLOCATED RBA COUNT</td>
</tr>
<tr>
<td>E27SCNT</td>
<td>VSAM DATA SET STRIPE COUNT</td>
</tr>
<tr>
<td>E27DSTP#</td>
<td>DATA COMPONENT STRIPE COUNT</td>
</tr>
<tr>
<td>EI27DRBA</td>
<td>RBAS PROCESSED FOR THE DATA COMPONENT</td>
</tr>
<tr>
<td>EI27IRBA</td>
<td>RBAS PROCESSED FOR THE INDEX COMPONENT</td>
</tr>
<tr>
<td>EI28PARMS1</td>
<td>EXIT 28 PARMS FIELD 1. ANY FIELD SET ON IDICATES THAT FIELD HAS BEEN SPECIFIED TO BE USED FOR THIS EXIT</td>
</tr>
<tr>
<td>EI28PARMS2</td>
<td>EXIT 28 PARMS FIELD 2</td>
</tr>
<tr>
<td>EI28TGTDSN</td>
<td>TARGET DATA SET NAME</td>
</tr>
<tr>
<td>EI28SRCDSN</td>
<td>SOURCE DATA SET NAME</td>
</tr>
<tr>
<td>EI28F9DSCB</td>
<td>FORMAT 9 DSCB</td>
</tr>
<tr>
<td>EI28FIELDS1</td>
<td>EXIT 28 FIELDS 1. FLAG BYTE FOR BIT FIELDS.</td>
</tr>
<tr>
<td>EI28CREDT</td>
<td>CREATION DATE (DS1CREDT)</td>
</tr>
</tbody>
</table>

---

### Application Interface

Table 38. ADREID0 Mapping Macro (continued)

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>CHARACTER</td>
<td>2</td>
<td>E27FLGS</td>
<td>MISC FLAGS</td>
</tr>
<tr>
<td>121</td>
<td>CHARACTER</td>
<td>1</td>
<td>E127LVOL</td>
<td>LAST VOLUME INDICATOR FROM THE SOURCE DATA SET</td>
</tr>
<tr>
<td>122</td>
<td>SIGNED</td>
<td>2</td>
<td>E127VLSQ</td>
<td>VOLUME SEQUENCE NUMBER FOR SOURCE DATA SET</td>
</tr>
<tr>
<td>124</td>
<td>SIGNED</td>
<td>4</td>
<td>E127DSHA</td>
<td>VSAM DATA COMPONENT ALLOCATED RBA COUNT</td>
</tr>
<tr>
<td>128</td>
<td>SIGNED</td>
<td>4</td>
<td>E127ISHA</td>
<td>VSAM INDEX COMPONENT HIGH ALLOCATED RBA COUNT</td>
</tr>
<tr>
<td>132</td>
<td>SIGNED</td>
<td>2</td>
<td>E127SCNT</td>
<td>VSAM DATA SET STRIPE COUNT</td>
</tr>
<tr>
<td>134</td>
<td>SIGNED</td>
<td>2</td>
<td>E27DSTP#</td>
<td>DATA COMPONENT STRIPE COUNT</td>
</tr>
</tbody>
</table>

Note: THE EI27DRBA AND EI27IRBA FIELDS ARE CALCULATED BY SUBTRACTING THE STARTING RBA FROM THE ENDING RBA.

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>136</td>
<td>UNSIGNED</td>
<td>4</td>
<td>EI27DRBA</td>
<td>RBAS PROCESSED FOR THE DATA COMPONENT</td>
</tr>
<tr>
<td>140</td>
<td>UNSIGNED</td>
<td>4</td>
<td>EI27IRBA</td>
<td>RBAS PROCESSED FOR THE INDEX COMPONENT</td>
</tr>
<tr>
<td>144</td>
<td>CHARACTER</td>
<td>24</td>
<td>*</td>
<td>RESERVED</td>
</tr>
</tbody>
</table>

Note: THE FOLLOWING MAPS THE RECORD PRESENTED BY EXIT 28: TARGET DATA SET ALLOCATION NOTIFICATION EXIT.
Cross reference for ADREID0

ADREIB 0
EIDDID 22
EIDDINFO 0
EIDONAME 0
EIDLLEN 0
EEID 2
EEIOPTION E
EIRECALT 12
EIRECLEN 16
EIRECPTR 1A
EIRECOD 0
EIREC14 0
EIREC20 0
EIREC21 0
EIREC22 0
EIREC23 0
EIREC24 0
EIREC26 0
EIREC27 0
EIREC2B 0
EIRECE 0
EIRECCOD 10
EITSKID 6
EIUSEPTR 1E
EIUSOLID 8
EIXABEND 26
EIXALLOW A
EIXALLOW0 A
EIXALLOW1 B
EIXALLOW2 C
EIXALLOW3 D
EIUSDYSYS 27
EIUSEL A
EIUSIS A
Application Interface
EIXERR D
EIXFLAGS 26
EIXFLAG0 26
EIXFLAG1 27
EIXFLAG2 28
EIXFLAG3 29
EIXINS A
EIXMOD A
EIXMWPL 22
EIXNTERR 26
EIXREP A
EIXSTAT A
EIXTDSET B
EIXTDUMS 27
EIXTDUMT 27
EIXTERM A
EIXTISXM 27
EIXTRC32 27
EIXTRERR 27
EIXTRKER 26
EIXTWERR 27
EIXWNGOK 26
EIXWTOR A
EI00BSEC 1
EI00BSER 1
EI00CANSFE 8
EI00FLGS 1
EI00FLG2 8
EI00MIXDEV 8
EI00MVOLRECOV 1
EI00NENQ 1
EI00NOLK 1
EI00NONF 1
EI00SBPL 0
EI00SENQ 1
EI00SVOL 2
EI00SWNCRYPT 8
EI06DCB@ 10
EI06FLGS F
EI06FTLR F
EI06THWE F
EI14CPUT 8
EI14MESS 4
EI14MNUM 4
EI14MTYP 7
EI14RC 0
EI20CON 0
EI20DA# 2E
EI20DA@ 30
EI20DRBA 3C
EI20DSHA 3C
EI20DSI 0
EI20DSN 0
EI20FLGS 2C
EI20ISHA 40
EI20IX# 2F
EI20IX@ 34
EI20LVOL 2C
EI20NVOL 2C
EI20RACF 2C
EI20RBA 38
EI20SCNT 3A
EI20VLI 30
EI20VLSQ 40
EI20VOL 30
EI20VSAM 2C
EI21DSN 0
EI22BMIG 2C
EI22BRLS 4E
EI22BSEC 2C
EI22BSER 2C
EI22BWOA 4E
EI22BWOE 32
EI22BWOP 32
EI22DB2 2C
EI22DSN 0
EI22EXTR 2C
EI22FLGS 2C
EI22FLG2 32
EI22FLG3 4E

614

z/OS V1R13.0 DFSMSdss Storage Administration


Application Interface
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=nmmK
//STEP1   EXEC PGM=MYJOB
//STEP2   DD DSN=MY.LINKLIB,DISP=SHR
//SYSOUT   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)
//SYSPRINT DD SYSOUT=A
//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=ADRDSSU,PARAM=(OPTPTR),VL=1
. .
. .
CNOP 2,4
OPTPTR DC H(0)
```

Related reading: For additional information about the Application Interface, see Chapter 23, “Examples of the application program with the user interaction module (UIM),” on page 619.

How to determine DFSMSdss 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 in-line 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 DFSMSdss from the information returned in registers 1 and 14.

On return, register 1 contains information as follows:
Application Interface

- 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.  
  Bits 1–7 are reserved.

  **Bytes 1–3**  
  Reserved.
Chapter 23. 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 26 shows the process by which DFSMSdss can call user interaction module (UIM) functions.

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 (Figure 27 on page 620 and Figure 28 on page 623)
- An output listing (Figure 29 on page 634) resulting from the sample program in Figure 27 on page 620 and Figure 28 on page 623

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.
**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 *

**USRAIPGM CSECT**
**USRAIPGM AMODE 31**
**USRAIPGM RMODE ANY**

* Load and call DFSMSdss *

**LOAD EP=ADRDSSU 01-LOAD**
LOAD B,6,4
LA 8,**B**
SR 1,1
LR 15,0
LA 3,USERAREA
ST 3,UA0

Figure 27. Application Interface Program Example (Part 1 of 3)
CALL (15),(OPTPTR,DDNPTR,PAGEPTR,UIMPTR,UAPTR),VL
+ DS OH 01-CALL
+ CNOP 0,4 02-IHB0P
+ LA 1,1HB0003 LIST ADDRESS @L1C 02-IHB0P
+ B 1HB0003A BYPASS LIST 02MC3742 02-IHB0P
+1HB0003 EQU * 02-IHB0P
+ DC A(OPTPTR) PROB.PROG.PARAMETER 02-IHB0P
+ DC A(DDNPTR) PROB.PROG.PARAMETER 02-IHB0P
+ DC A(PAGEPTR) PROB.PROG.PARAMETER 02-IHB0P
+ DC A(UIMPTR) PROB.PROG.PARAMETER 02-IHB0P
+ DC A(UAPTR+X'80000000') 0GB60P40 02-IHB0P
+1HB0003A EQU * 02-IHB0P
+ 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,UA0
* CALL (15),(OPTPTR,DDNPTR,PAGEPTR,UIMPTR,UAPTR,ASNPTR),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
+ DONTR DC AL2(DONLEN)
+ DONAMES DC XL8'00'
+ DC XL8'00'
+ DC XL8'00'
+ DC CL8'MYDATA'
+ DC CL8'SYSPRINT'
+ DONLEN EQU +-DONAMES
**********************************************************************
* Page number area (first page will be 1) *
**********************************************************************
+ CNOP 2,4
+ PAGEPTR DC AL2(PAGELEN)
+ PAGENO DC CL4'0001'
+ PAGELEN EQU +-PAGEptr

Figure 27. Application Interface Program Example (Part 2 of 3)
Figure 27. Application Interface Program Example (Part 3 of 3)
<table>
<thead>
<tr>
<th>Exit Point</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Initialize counters for task</td>
</tr>
<tr>
<td>1</td>
<td>Insert SYSIN records from user area</td>
</tr>
<tr>
<td>2</td>
<td>Insert table of accounting data</td>
</tr>
<tr>
<td>3</td>
<td>Count Tape Blocks Read</td>
</tr>
<tr>
<td>4</td>
<td>Count Logical Tape Blocks Read</td>
</tr>
<tr>
<td>5</td>
<td>Count Logical Tape Blocks Written</td>
</tr>
<tr>
<td>6</td>
<td>Count Tape Blocks Written</td>
</tr>
<tr>
<td>7</td>
<td>Count DASD Tracks Read</td>
</tr>
<tr>
<td>8</td>
<td>Count DASD Tracks Written</td>
</tr>
<tr>
<td>9</td>
<td>Nothing</td>
</tr>
<tr>
<td>10</td>
<td>Nothing</td>
</tr>
<tr>
<td>11</td>
<td>Insert a WTO message</td>
</tr>
<tr>
<td>12</td>
<td>Never allow ADR369D</td>
</tr>
<tr>
<td>13</td>
<td>Force Reblocking</td>
</tr>
<tr>
<td>14</td>
<td>Convert counts to printable characters</td>
</tr>
<tr>
<td>15</td>
<td>Nothing</td>
</tr>
<tr>
<td>16</td>
<td>Always bypass password protection for tape</td>
</tr>
<tr>
<td>17</td>
<td>Supply TAPE01 as a tape volser</td>
</tr>
<tr>
<td>18</td>
<td>Nothing</td>
</tr>
<tr>
<td>19</td>
<td>Save Tape DDNAME and volser which had error</td>
</tr>
<tr>
<td>20</td>
<td>Save Data Set name</td>
</tr>
<tr>
<td>21</td>
<td>Do not process temporary data sets</td>
</tr>
<tr>
<td>22</td>
<td>Bypass serialization of SYS1 data sets</td>
</tr>
<tr>
<td>23</td>
<td>End function if processing errors</td>
</tr>
<tr>
<td>24</td>
<td>End function if initialization fails</td>
</tr>
<tr>
<td>25</td>
<td>Nothing</td>
</tr>
<tr>
<td>26</td>
<td>Nothing</td>
</tr>
</tbody>
</table>

**User Interaction Module Example (Part 1 of 11)**

---

**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 Set my SA pointer
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 RLTCNT,RLTCNT Init Log Tape Blocks Read
XC WLTCNT,WLTCNT 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,EIRECOD 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. *
********************************************************************

Figure 28. User Interaction Module Example (Part 2 of 11)
### AIOPT01

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM</td>
<td>SYSINL, SYSIFRST</td>
</tr>
<tr>
<td>BO</td>
<td>NXTSYS</td>
</tr>
<tr>
<td>OI</td>
<td>SYSINL, SYSIFRST</td>
</tr>
<tr>
<td>ICM</td>
<td>TEMPBASE, 15, EIPMTR</td>
</tr>
<tr>
<td>ST</td>
<td>TEMPBASE, CURCOMM</td>
</tr>
<tr>
<td>NXTSYS</td>
<td>L TEMPBASE, CURCOMM</td>
</tr>
<tr>
<td>LA</td>
<td>WORKREG, 1(TEMPBASE)</td>
</tr>
<tr>
<td>STCM</td>
<td>WORKREG, 15, EIRECPTR</td>
</tr>
<tr>
<td>IC</td>
<td>LENGTH, LENGTH</td>
</tr>
<tr>
<td>LTR</td>
<td>LENGTH, LENGTH</td>
</tr>
<tr>
<td>BZ</td>
<td>SYSDEL</td>
</tr>
<tr>
<td>STCM</td>
<td>LENGTH, 15, EIRECLEN</td>
</tr>
<tr>
<td>LA</td>
<td>TEMPBASE, 1(LENGTH, TEMPBASE)</td>
</tr>
<tr>
<td>ST</td>
<td>TEMPBASE, CURCOMM</td>
</tr>
<tr>
<td>LH</td>
<td>RC, EIRC08</td>
</tr>
<tr>
<td>B</td>
<td>SYSEXIT</td>
</tr>
</tbody>
</table>

### NXTSYS

- **ICM**: TEMPBASE, CURCOMM
- **LA**: WORKREG, 1(LENGTH, TEMPBASE)
- **STCM**: WORKREG, 15, EIRECPTR
- **IC**: LENGTH, LENGTH
- **LTR**: LENGTH, LENGTH
- **BZ**: SYSDEL
- **STCM**: LENGTH, 15, EIRECLEN
- **LA**: TEMPBASE, 1(LENGTH, TEMPBASE)
- **ST**: TEMPBASE, CURCOMM
- **LH**: RC, EIRC08
- **B**: SYSEXIT

### SYSDEL

- **ICM**: LENGTH, 15, EIRECLEN
- **LTR**: LENGTH, LENGTH
- **BZ**: SYSRC00
- **LH**: RC, EIRC12
- **B**: SYSEXIT

### SYSRC00

- **LH**: RC, EIRC00

### SYSEXIT

- **LH**: RC, EIRC00

---

### AIOPT02

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICM</td>
<td>SYSBASE, 15, EIRECPTR</td>
</tr>
<tr>
<td>CLC</td>
<td>MSGID(8), ADR012I</td>
</tr>
<tr>
<td>BNE</td>
<td>NOTLAST</td>
</tr>
<tr>
<td>TM</td>
<td>SYSPRFL, INSDONE</td>
</tr>
<tr>
<td>BO</td>
<td>NOTLAST</td>
</tr>
<tr>
<td>L</td>
<td>WORKREG, INSERTCT</td>
</tr>
<tr>
<td>LR</td>
<td>MSGOFF, WORKREG</td>
</tr>
<tr>
<td>LA</td>
<td>WORKREG, 1(WORKREG)</td>
</tr>
<tr>
<td>C</td>
<td>WORKREG, MSGCOUNT</td>
</tr>
<tr>
<td>BNE</td>
<td>INSERT</td>
</tr>
<tr>
<td>WI</td>
<td>SYSPRFL, INSDONE</td>
</tr>
<tr>
<td>LA</td>
<td>WORKREG, TRAILER</td>
</tr>
<tr>
<td>STCM</td>
<td>WORKREG, 15, EIRECPTR</td>
</tr>
<tr>
<td>LH</td>
<td>WORKREG, MSGLEN</td>
</tr>
<tr>
<td>STCM</td>
<td>WORKREG, 15, EIRECLEN</td>
</tr>
<tr>
<td>B</td>
<td>SYSEXIT</td>
</tr>
</tbody>
</table>

### INSERT

- **MH**: MSGOFF, MSGLEN
- **LA**: WORKREG, MSGTABLE
- **LA**: WORKREG, 0(MSGOFF, WORKREG)
- **STCM**: WORKREG, 15, EIRECPTR
- **LH**: WORKREG, MSGLEN
- **STCM**: WORKREG, 15, EIRECLEN
- **L**: WORKREG, INSERTCT
- **LA**: WORKREG, 1(WORKREG)
- **ST**: WORKREG, INSERTCT
- **LH**: RC, EIRC08
- **B**: SYSEXIT

### NOTLAST

- **LH**: RC, EIRC00

### SYSEXIT

- **LH**: RC, EIRECOD

---

Figure 28. User Interaction Module Example (Part 3 of 11)
Figure 28. User Interaction Module Example (Part 4 of 11)
Figure 28. User Interaction Module Example (Part 5 of 11)
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,UBFYOFF  Get offset for PARM list
AR WORKREG,TEMPDBUFF  Add to address of UF0
USING UFOFUNCT,WORKREG  Establish addressability
OI UXFAPLUE,UFXABUFF  Make sure IO buffers are >16M
OI UXFAPLUE,UFXA31B  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),BLANK  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: Function Ending

AIOPT14 LH WORKREG,RTBCNT  Get Tape Blocks Read
CVD WORKREG,CDVWORK  Convert to decimal
UNPK RTBINS,CDVWORK  Unpack into message insert
OI RTBINS+7,X'F0'  Make last character printable
LH WORKREG,WTBCNT  Get Tape Blocks Written
CVD WORKREG,CDVWORK  Convert to decimal
UNPK WTBINS,CDVWORK  Unpack into message insert
OI WTBINS+7,X'F0'  Make last character printable
LH WORKREG,RLTBCNT  Get Tape Logical Blocks Read
CVD WORKREG,CDVWORK  Convert to decimal
UNPK RLTTINS,CDVWORK  Unpack into message insert
OI RLTTINS+7,X'F0'  Make last character printable
LH WORKREG,LDTCNT  Get Tape Log Blocks Written
CVD WORKREG,CDVWORK  Convert to decimal
UNPK WTINS,CDVWORK  Unpack into message insert
OI WTINS+7,X'F0'  Make last character printable
LH WORKREG,WDTCNT  Get Disk Tracks Read
CVD WORKREG,CDVWORK  Convert to decimal
UNPK DTTINS,CDVWORK  Unpack into message insert
OI DTTINS+7,X'F0'  Make last character printable
LH WORKREG,DTATCNT  Get Disk Tracks Written
CVD WORKREG,CDVWORK  Convert to decimal
UNPK WTINS,CDVWORK  Unpack into message insert
OI WTINS+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)

Figure 28. User Interaction Module Example (Part 6 of 11)
Figure 28. User Interaction Module Example (Part 7 of 11)
**UIM**

---

---

**Figure 28. User Interaction Module Example (Part 8 of 11)**

---

---
AIOPT23 EQU *
ICM TEMPBASE,15,EIREC_PTR Get record pointer
USING EIREC23,TEMPBASE Get addressability
LH WORKREG,EIRC_SRC If data set processing RC^=0
DROP TEMPBASE Release addressability
LTR WORKREG,WORKREG Then
BZ LABL1 End function
B LABL2 Else
LABL1 LH RC,EIRC00 Set normal return code
LABL2 STH RC,EIRET_CODE Store retcode in EIDB
BR 14 Return to intercept processor

AIOPT24 EQU *
Start EIOP24 processing
LH RC,EIRC00 Assume goodness
ICM TEMPBASE,15,EIREC_PTR Get record pointer
USING EIREC24,TEMPBASE Get addressability
LH WORKREG,EIRC_RT_CODE Get the return code
DROP TEMPBASE Release addressability
LTR WORKREG,WORKREG If zero
BZ OPT24_END End function
LH RC,EIRC32 Else end the function
OPT24_END STH RC,EIRET_CODE Store retcode in EIDB
BR 14 Return to intercept processor

AIOPT25 EQU *
Start EIOP25 processing
LH RC,EIRC00 Set normal process retcode
STH RC,EIRET_CODE Store retcode in EIDB
BR 14 Return to intercept processor

AIOPT26 EQU *
Start EIOP26 processing
LH RC,EIRC00 Set normal process retcode
STH RC,EIRET_CODE Store retcode in EIDB
BR 14 Return to intercept processor

CURCOMM DC F'0' Address of current command
SYSINFL DC X'00' SYSIN Flag
SYSIFRST EQU X'80' First entry SYSIN Performed
SYSPRF DC X'00' SYSPRINT Flag
INSDONE EQU X'80' Done inserting SYSPRINT reccs
TMPL DC C'TEMP' TEMP high-level qualifier
SYSL DC C'SYS1.' SYS1 high-level qualifier

SAVEAREA DC 18F'0' My save area

Figure 28. User Interaction Module Example (Part 9 of 11)
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

* UIMWT0 DC C'>>> USRUIM is active <<<'

WTOLEN EQU *-UIMWT0 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
* RDTCNT DC H'0' Count Disk Tracks Read
WDTCNT 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
*************************************************************************

Figure 28. User Interaction Module Example (Part 10 of 11)
* 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'XXXXXXXXX'
DC C' Tape Blocks Read: ' RTBINS DC C'XXXXXXXXX'
DC C' Logical Tape Blocks Read: ' RLTBINS DC C'XXXXXXXXX'
DC CL(133-(READMSG))'
*
WRITEMSG DC C' Disk Tracks Written: ' WDTINS DC C'XXXXXXXXX'
DC C' Tape Blocks Written: ' WTBINS DC C'XXXXXXXXX'
DC C' Logical Tape Blocks Written: ' WLTBINS DC C'XXXXXXXXX'
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'XXXXXXXXX'
DC C' VOLSER: ' OPT19VS DC C'XXXXXXXXX'
DC C' Return Code: ' OPT19RC DC C'XX'
DC CL(133-(OPT19MSG))'
*
OPT20MSG DC C' EI20DSN = ''' OPT20DSN DC C'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
DC C''''''
DC CL(133-(OPT20MSG))'
*
CVDWORK DS D Workarea for decimal conversion
*
ADREID0 Macro to map EIDB
*
ADRUF0 Macro to map UFO
*
USRUIM CSECT
MSGTABLE DS CL13300 Room in table for 100 msgs
*
TABLEMAP DSECT DSECT to insert msgs into table
TABENTRY DS CL133
*
SYSPMAP DSECT DSECT to find MSGID in DFSMSdss
MSGID DS CL1

Figure 28. User Interaction Module Example (Part 11 of 11)
Figure 29. Output Resulting from Use of the UIM Exits (Part 1 of 2)
Figure 29. Output Resulting from Use of the UIM Exits (Part 2 of 2)
Chapter 24. 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 39. 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 39. Data Set Attributes and How They Are Determined.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Is the Attribute Determined by...</th>
<th>...the Source Data Set?</th>
<th>...the Pre-Allocated Target?</th>
<th>...a Keyword?</th>
<th>...another Factor?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set name</td>
<td></td>
<td>Yes, if no RENAME or RENAMEU</td>
<td>No</td>
<td>RENAME or RENAMEU</td>
<td>No</td>
</tr>
<tr>
<td>Data set size</td>
<td></td>
<td>Yes, unless the preallocated target data set is larger</td>
<td>Yes, if it is big enough</td>
<td>ALLDATA(x) or ALLEXCP</td>
<td>No</td>
</tr>
<tr>
<td>Volumes</td>
<td></td>
<td>Yes, if doing a RESTORE and no output volumes were specified, no target exists, and not SMS-managed</td>
<td>Yes</td>
<td>Yes, if not SMS, OUTDDNAME(x,...) or OUTDYNAM(x,...)</td>
<td>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</td>
</tr>
<tr>
<td>Data set location on volume</td>
<td></td>
<td>Yes, if no target and either ABSTR, PSU, POU, or DAU</td>
<td>Yes</td>
<td>FORCE can override ABSTR, PSU, POU, and DAU</td>
<td>DFSMSdss locates wherever space is available; DEFRAG may move extents</td>
</tr>
<tr>
<td>PDS directory size (blocks)</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PDSE directory size (blocks)</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SMS Storage Class or</td>
<td></td>
<td>Yes, if no target and BYPASSACS is specified</td>
<td>Yes</td>
<td>Yes, if STORCLAS(x) or MGMTCLAS(x), or both, are specified with BYPASSACS</td>
<td>ACS routines if no target and BYPASSACS is not specified</td>
</tr>
<tr>
<td>Management Class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMS Data Class</td>
<td></td>
<td>Yes, if no target</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>BLKSZ</td>
<td></td>
<td>Yes, if REBLOCK is not specified and data set is not system reblockable</td>
<td>No</td>
<td>If REBLOCK keyword is specified, DFSMSdss chooses a new optimal blocksize</td>
<td>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</td>
</tr>
<tr>
<td>LRECL</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RECFM</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DSORG</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Number of stripes</td>
<td></td>
<td>Yes, source must be striped</td>
<td>Yes for non-VSAM. No for VSAM</td>
<td>No</td>
<td>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</td>
</tr>
</tbody>
</table>
### Data Set Attributes

**Table 39. Data Set Attributes and How They Are Determined. (continued)**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Is the Attribute Determined by...</th>
<th>...the Source Data Set?</th>
<th>...the Pre-Allocated Target?</th>
<th>...a Keyword?</th>
<th>...another Factor?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of volumes (VOLCOUNT)</td>
<td>Yes</td>
<td>Yes</td>
<td>VOLCOUNT can make a single volume source into a multivolume target, or change the number of volumes for a multivolume data set</td>
<td>Yes (see the COPY and RESTORE command VOLCOUNT parameter descriptions for specific information)</td>
<td></td>
</tr>
<tr>
<td>Number of extents</td>
<td>Yes, for imbedded extended KSDSs during physical data set restore</td>
<td>Yes</td>
<td>No</td>
<td>Yes, if doing a COPY, CONVERT(PDS(x)) or CONVERT(PDSE(x))</td>
<td>DFSMSdss always tries to consolidate during COPY/RESTORE. RELEASE may reduce the number of extents</td>
</tr>
<tr>
<td>PDS/PDSE</td>
<td>Yes, if no target or if doing a RESTORE</td>
<td>Yes, if doing a COPY</td>
<td>No</td>
<td>Yes, if doing a COPY, CONVERT(PDS(x)) or CONVERT(PDSE(x))</td>
<td>If SMS or VSAM, cataloged by default; if physical data set restore, only single volume non-VSAM is cataloged (if CATALOG is specified)</td>
</tr>
<tr>
<td>Cataloged</td>
<td>Yes, if RECATALOG(*) is specified, no target, and the user is not doing a physical data set restore</td>
<td>Yes</td>
<td>RECATALOG(x), CATALOG, UNCATALOG (applies to source only)</td>
<td>Yes, if RECATALOG(*) is specified, no target, and the user is not doing a physical data set restore</td>
<td></td>
</tr>
<tr>
<td>Allocation unit</td>
<td>Yes, if no target and TGTALLOC (SOURCE) specified or defaulted</td>
<td>Yes</td>
<td>TGTALLOC(x)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Free space in VSAM</td>
<td>Yes, if going to like device, nonVALIDATE, no dummy blocks, and CI and CA sizes do not change</td>
<td>Yes, if doing VSAM I/O and uses values in target catalog entry</td>
<td>VALIDATE, NOVALIDATE, FREESPACE</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Security (RACF)</td>
<td>If no target and source was generic or discrete, and no applicable profile protecting new target, a new discrete will be defined</td>
<td>Yes</td>
<td>MENTITY, MVOLSER</td>
<td>Full RACF profile information (access lists) are not preserved</td>
<td></td>
</tr>
<tr>
<td>AIX data sets on VSAM clusters</td>
<td>If SPHERE is specified during COPY or DUMP and RESTORE, sphere and connections are preserved</td>
<td>No</td>
<td>SPHERE</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>GDS state</td>
<td>Yes, if no target and TGTGDS (source) specified</td>
<td>Yes</td>
<td>TGTGDS(x)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>RLS BWO field</td>
<td>Yes, if no preallocated target and no UIM input</td>
<td>Yes, if no UIM input</td>
<td>No</td>
<td>UIM can pass a value in Exit 22 during logical RESTORE</td>
<td></td>
</tr>
<tr>
<td>RLS timestamps</td>
<td>Yes, if not a logical restore</td>
<td>No</td>
<td>No</td>
<td>For logical restore, if dumped using RLS access, timestamps reflect the time of the dump; otherwise the timestamps are zero</td>
<td></td>
</tr>
<tr>
<td>RLS recovery required</td>
<td>Yes, if no UIM input</td>
<td>No</td>
<td>No</td>
<td>UIM can pass a value in Exit 22 during logical RESTORE</td>
<td></td>
</tr>
<tr>
<td>RLS log parameter</td>
<td>Yes, if no preallocated target and no UIM input</td>
<td>Yes, if no UIM input</td>
<td>No</td>
<td>UIM can pass a value in Exit 22 during logical RESTORE</td>
<td></td>
</tr>
<tr>
<td>RLS log stream ID</td>
<td>Yes, if no preallocated target and no UIM input</td>
<td>Yes, if no UIM input</td>
<td>No</td>
<td>UIM can pass a value in Exit 22 during logical RESTORE</td>
<td></td>
</tr>
</tbody>
</table>
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 DFDS 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

Publications for this product are offered in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when using PDF files, you may view the information through the z/OS Internet Library website or the z/OS Information Center. If you continue to experience problems, send an email to mhvrdfs@us.ibm.com or write to:

IBM Corporation
Attention: MHVRDFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to z/OS TSO/E Primer, z/OS TSO/E User’s Guide, and z/OS ISPF User’s Guide Vol I for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer or Library Server versions of z/OS books in the Internet library at:
http://www.ibm.com/systems/z/os/zos/bkserv/

One exception is command syntax that is published in railroad track format, which is accessible using screen readers with the Information Center, as described in “Dotted decimal syntax diagrams.”

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always
present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only
one of the syntax elements that share the same dotted decimal number can
specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and
2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword.
In this example, if you include the FILE keyword but do not specify an option,
default option KEEP will be applied. A default option also applies to the next
higher dotted decimal number. In this example, if the FILE keyword is omitted,
default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1!
(KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next
higher dotted decimal number, 2.1 (which does not have an associated
keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE
is omitted.

- * means a syntax element that can be repeated 0 or more times. A dotted
decimal number followed by the * symbol indicates that this syntax element can
be used zero or more times; that is, it is optional and can be repeated. For
example, if you hear the line 5.1* data area, you know that you can include one
data area, more than one data area, or no data area. If you hear the lines 3*, 3
HOST, and 3 STATE, you know that you can include HOST, STATE, both
together, or nothing.

Notes:
1. If a dotted decimal number has an asterisk (*) next to it and there is only one
   item with that dotted decimal number, you can repeat that same item more
   than once.
2. If a dotted decimal number has an asterisk next to it and several items have
   that dotted decimal number, you can use more than one item from the list,
   but you cannot use the items more than once each. In the previous example,
you could write HOST STATE, but you could not write HOST HOST.
3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

- + means a syntax element that must be included one or more times. A dotted
decimal number followed by the + symbol indicates that this syntax element
must be included one or more times; that is, it must be included at least once
and can be repeated. For example, if you hear the line 6.1+ data area, you must
include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE,
you know that you must include HOST, STATE, or both. Similar to the * symbol,
the + symbol can only repeat a particular item if it is the only item with that
dotted decimal number. The + symbol, like the * symbol, is equivalent to a
loop-back line in a railroad syntax diagram.
Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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

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

This publication primarily documents information that is NOT intended to be used as a Programming Interface of DFSMSdss.

This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of DFSMSdss. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Programming Interface information

End of Programming Interface information

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, 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.

Trademarks

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

This glossary defines technical terms and abbreviations used in DFSMSdss documentation. If you do not find the term you are looking for, refer to the index of the appropriate DFSMSdss manual or view IBM Dictionary of Computing, located at http://www.ibm.com/networking/nsg/nsgmain.htm

A

ABARS. Aggregate backup and recovery support.

ABEND. Abnormal end of task. End of a task, a job, or a subsystem because of an error condition that cannot be resolved by recovery facilities while the task is performed.

ABENDxxx. The keyword that identifies the abnormal end of DFSMSdss because of a system-detected error.

ABSTR. A subparameter of the SPACE parameter in a DD statement. It indicates that specified tracks be assigned to a data set.

ACCEPT processing. An SMP/E process necessary for installing the FMIDs. SMP/E ACCEPT processing uses JCL to accept the modules and macros necessary to run the FMIDs. The FMIDs are accepted into the DLIBs from the temporary data sets.

ACCESS method services. A multifunction service program that is used to manage both VSAM and non-VSAM data sets and integrated catalog facility or the ICF catalog. It is used to define data sets and allocate space for them; convert indexed-sequential data sets to key-sequenced data sets; modify data set attributes in the catalog; reorganize data sets; facilitate data portability between operating systems; create backup copies of data sets, data set records, and catalog entries; help make inaccessible data sets accessible; list the records of data sets and catalogs; define and build alternate indexes; and convert OS CVOLs and the ICF catalog to integrated catalog facility catalogs.

ACDS. Active control data set.

ACS. Automatic class selection.

ADSP. Automatic data set protection.

alias. An alternate name for a member of a partitioned data set.

ALLOC. A space allocation parameter that indicates type, such as cylinders or tracks.

alternate index. In systems with VSAM, a key-sequenced data set containing index entries organized by the alternate keys of its associated base data records. It provides an alternate means of locating records in the data component of a cluster on which the alternate index is based.

alternate index cluster. In VSAM, the data and index components of an alternate index.

APAR. Authorized program analysis report.

APF. Authorized program facility.

application interface. An interface used to invoke DFSMSdss from another program.

apply processing. In SMP and SMP/E, the process, initiated by the APPLY command, that places system modifications (SYSMODS) into the target system libraries.

attach. In programming, to create a task that can be performed asynchronously with the performance of the mainline code.

authorization. (1) The right granted to a user to communicate with or make use of a computer system. (2) The process of giving a user either complete or restricted access to an object, resource, or function.

authorized program analysis report (APAR). A request for correction of a problem caused by a suspected defect in a current unaltered release of a program.

automatic class selection (ACS). A mechanism for assigning SMS classes and storage groups.

automatic data set protection (ADSP). A system function, enabled by the SETROPTS ADSP specification and the assignment of the ADSP attribute to a user with ADDUSER or ALTUSER, that causes all permanent data sets created by the user to be automatically defined to RACF with a discrete RACF profile.

backout. The CICSVR function that you can use if CICS fails in the attempt to back out uncommitted changes on a VSAM sphere. Using information from the RCDS, CICSVR constructs a job to back out uncommitted changes on a VSAM data set as indicated on the log.
**backup.** The process of creating a copy of a data set to ensure against accidental loss.

**backup while open.** DFSMSdss can perform backup of data sets that are open for update for a long period of time (like CICS). DFSMSdss can perform a logical data set dump of these data sets even if another application has them serialized.

**base cluster.** In systems with VSAM, a key-sequenced or entry-sequenced data set over which one or more alternate indexes are built.

**basic catalog structure (BCS).** The name of the catalog structure in the integrated catalog facility environment. An integrated catalog facility catalog consists of a BCS and its related VSAM volume data sets (VVDSs).

**basic direct access method (BDAM).** An access method used to directly retrieve or update particular blocks of a data set on a direct access device.

**basic partitioned access method (BPAM).** An access method that can be applied to create program libraries in direct access storage for convenient storage and retrieval of programs.

**basic sequential access method (BSAM).** An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or a direct access device.

**BCS.** Basic catalog structure.

**BDAM.** Basic direct access method.

**BLK.** A subparameter of the SPACE parameter in a DD statement. It specifies that space is allocated by blocks.

**block length.** Synonym for block size.

**block size.** (1) The number of data elements in a block. (2) A measure of the size of a block, usually specified in units such as records, words, computer words, or characters. (3) Synonymous with block length. (4) Synonymous with physical record size.

**BPAM.** Basic partitioned access method.

**bpi.** Bits per inch.

**Broken data set.** Data sets which do not conform to IBM data set standards are referred to as broken. Broken data sets are either missing catalog entries, VTOC entries, or VVDS entries; or, have invalid catalog entries, VTOC entries, or VVDS entries.

**BSAM.** Basic sequential access method.

**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 (COMPACTON) parameter of data class. VSAM can compress individual records in a compressed-format
concatenation. An operation that joins two characters or strings in the order specified, forming one string whose length is equal to the sum of the lengths of the two characters or strings.

concurrent copy. A function to increase the accessibility of data by letting you make a consistent backup or copy of data concurrent with normal application program processing.

concurrent copy-compatible (CC-compatible) SnapShot. See virtual concurrent copy.

conditioned volume. The target volume from a previous FULL volume COPY operation which specified DUMPCONDITIONING.

control area (CA). A group of control intervals used as a unit for formatting a data set before adding records to it. Also, in a key-sequenced data set, the set of control intervals, pointed to by a sequence-set index record, that is used by VSAM for distributing freespace and for placing a sequence-set index record adjacent to its data.

control interval (CI). A fixed-length area of auxiliary storage space in which VSAM stores records. It is the unit of information transmitted to or from auxiliary storage by VSAM.

control volume (CVOL). A volume that contains one or more indexes of the catalog.

constructs. A collective name for data class, storage class, management class, and storage group.

CONVERTV command. The DFSMSdss function that converts volumes to and from Storage Management Subsystem management without data movement.

COPY command. The DFSMSdss function that performs data set, volume, and track movement.

CP. Control program.

CREDT. Creation date.

CSW. Channel status word.

CVAF. Common VTOC access facility.

CVOL. Control volume.

CVT. Communication vector table.

CYL. A subparameter of the SPACE parameter in a DD statement. It specifies that space is allocated by cylinders.

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 DFSMSdfp, DFSMSdss, DFSMShsm, and DFSMSrmm which, together with RACF, provide a system-managed, administrator-controlled storage environment.

data set backup. Backup to protect against the loss of individual data sets.

data set 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.

DFSMSshsm. 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.

dsnname. 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 since the last backup. CICSVR gets the information it needs to construct the recovery job from the RCDS. The contents of the logs are applied to the VSAM sphere to return it to its exact state before the data was lost. With CICSVR forward recovery, CICSVR restores a DFSMSShsm or DFSMSdss backup for you.

fragmentation index. The qualitative measure of the scattered free space on a volume.

fully-qualified data set name. A data set in which all the qualifiers are completely spelled out.

function modification identifier (FMID). A code that identifies the release levels of a program product.

FVL. Function vector list.

G

GDG. Generation data group.

GDS. Generation data set.

generalized trace facility (GTF). An optional OS/VS service program that records significant systems events, such as supervisor calls and start I/O operations, for the purpose of problem determination.

generation data group (GDG). A collection of historically related non-VSAM data sets that are arranged in chronological order; each data set is a generation data set.

generation data set. One generation of a generation data group.

global resource serialization (GRS). A component of z/OS used for serializing use of system resources and for converting hardware reserves on DASD volumes to data set enqueues.

GT. Greater than.

GTF. Generalized trace facility.

H

HFS. Hierarchical file system.

I

ICKDSF. Device Support Facilities.

IDCAMS. Access Method Services.

IDRC. Improved data recording capability.

IMS/VS. Information Management System/Virtual Storage.

INCORROUT. In diagnosing program failures, the keyword that identifies incorrect or missing program output.

incremental backup. A process in which data sets are backed up only if they have changed since their last backup.

installation exit. The means specifically described in an IBM software product's documentation by which an IBM software product may be modified by a customer's system programmers to change or extend the functions of the IBM software product. Such modifications consist of exit routines written to replace one or more existing modules of an IBM software product, or to add one or more modules or subroutines to an IBM software product, for the purpose of modifying (including extending) the functions of the IBM software.
**integrated catalog facility.** A facility by which VSAM data set volume-related fields are separated from the catalog and maintained in the VVDS on the volume on which the data set resides.

**integrated catalog facility catalog.** A catalog that is composed of a basic catalog structure (BCS) and its related volume table of contents (VTOC) and VSAM volume data sets (VVDSs).

**Interactive Problem Control System (IPCS).** A component of 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.

**ISAM.** Indexed sequential access method.

**ISMF.** Interactive Storage Management Facility.

**ISPF.** Interactive Systems Productivity Facility.

**ISPF/PDF.** Interactive Systems Productivity Facility/Program Development Facility.

**J**

**JCL.** Job control language.

**JES.** Job entry subsystem.

**JES2.** An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for operation, processes their output, and purges them from the system. In an installation site with more than one processor, each JES2 processor independently controls its job input, scheduling, and output processing.

**JES3.** An MVS subsystem that receives jobs into the system, converts them to internal format, selects them for operation, processes their output, and purges them from the system. In complexes that have several loosely coupled processing units, the JES3 program manages processors so that the global processor exercises centralized control over the local processors and distributes jobs to them via a common job queue.

**JFCB.** Job file control block.

**job control language (JCL).** A problem-oriented language used to identify the job or describe its requirements to an operating system.

**job entry subsystem (JES).** A system facility for spooling, job queuing, and managing I/O.

**JSCB.** Job step control block.

**K**

**K.** Kilobyte: 1 024 bytes.

**key-sequenced data set.** A VSAM file or data set whose records are loaded in ascending key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in key sequence by means of distributed free space. Relative byte addresses can change because of control interval or control area splits.

**keyword.** A symptom that describes one aspect of a program failure.

**KRDS.** Keyrange data set. Also known as a key-sequenced data set with key ranges.

**KSDS.** Key-sequenced data set.

**L**

**LASTCC.** Last condition code.

**LDS.** Linear data set.

**like devices.** Devices that have the same track capacity and number of tracks per cylinder (for example, 3380 Model D, Model E, and Model K).

**LINK.** An assembler language macro instruction that causes control to be passed to a specified entry point. The linkage relationship established is the same as that created by a basic assembler language (BAL) instruction.

**link-pack area (LPA).** An area of virtual storage that contains reenterable routines that are loaded at IPL (initial program load) time and can be used concurrently by all tasks in the system.

**load module.** A computer program in a form suitable for loading into main storage for operation.

**load module library.** A partitioned data set used to store and retrieve load modules.

**logical DUMP operation (data set).** A DUMP operation in which logical processing is performed.
**Logical Processing (Data Set).** Processing that treats each data set and its associated information as a logical entity. As an example, DFSMSdss processes an entire data set before beginning with the next one.

**Logical Storage Subsystem (LSS).** Used internally by ESS to manage a set of logical volumes which are associated with an individual device adapter, e.g., a physical ESS subsystem may be partitioned into multiple logical storage subsystems.

**Logical Restore Operation (Data Set).** A restore operation that uses as input a data set produced by a logical DUMP operation.

**Logical Volume.** The output produced from a physical DUMP operation, for which all data is derived from a single DASD volume.

**Loop.** In diagnosing program failures, the keyword that identifies a program failure in which some part of the program repeats endlessly.

**LPA.** Link-pack area.

**LSS.** Logical storage subsystem.

**LT.** Less than.

**LRECL.** Logical record length.

**LVOL.** Logical volume.

**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. Minivolmes are supported only by the system version of DFSMSdss.

**MSGADRnnnt.** In diagnosing program failures, the DFSMSdss message keyword that tells of an error, or seems itself to be in error.

**MVS.** Multiple Virtual Storage.

**N.**

**NVR.** Non-VSAM volume record.

**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.

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 DFSMSdsdss 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 DFSMSdsdss function that releases the unused space in sequential and partitioned data sets for use by other data sets.

release command. The DFSMSdsdss function that releases the unused space in sequential and partitioned data sets for use by other data sets.

reserve command. The DFSMSdsdss function that releases the unused space in sequential and partitioned data sets for use by other data sets.

reserve command. The DFSMSdsdss function that releases the unused space in sequential and partitioned data sets for use by other data sets.

RESTORE command. The DFSMSdsdss 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 DFSMSdsdss 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 zSeries File System.

zSeries File System (zFS). A z/OS UNIX file system that can be used in addition to the hierarchical file system (HFS). zFS stores files in VSAM linear data sets. z/OS provides support for zFS in its Distributed File Service element.
Index

Special characters
* (single asterisk) used in partially qualified data set names 25, 264
** (double asterisk) used in partially qualified data set names 25, 264
% (percent sign) used in partially qualified data set names 25, 264

Numerics
16 MB virtual storage, storage requirements per command 15, 16
16 megabytes virtual storage buffers above 261
31-bit addressing mode 586
3380 Direct Access Storage 93, 130
3390 Direct Access Storage 93, 130
3494 Tape Library 511
3495 Tape Library 511
64-bit addressing mode 261
9345 DASD module 93, 130

A
ABEND keyword 258
ABOVE16 keyword 261
access authorization for DFSMSdss commands 545
accessibility 641
ACCESSVOL keyword
FlashCopy relationship, volume 135
thawing the frozen, volume 135
ACL (automatic cartridge loader) 514
ACS information 179
ACS variables
CONVERTV 181
COPY 179
name/description 179
passed in COPY command 179
passed in RESTORE/CONVERTV 181
RESTORE 181
ACTIVE state 90, 122
ACTIVE subkeyword of TGTGDS COPY command 357
RESTORE command 484
ADMINISTRATOR keyword
COMPRESS command 281
CONSOLIDATE command 287
COPY command 311
DEFRAG command 376
DUMP command 398
FACILITY class profiles for 539
PRINT command 433
RELEASE command 443
RESTORE command 459
with FlashCopy 131
with native SnapShot 137
ADMINISTRATOR parameter
(Stand-Alone BUILDSA command) 274

ADRDSU 575
ADRDSU program
using for Linux partition 187
ADREID0 data area 604
ADRMCLVL macro 617
ADRTAPB 198
ADRUF0, presenting record – eioption 13 593
ADRUM module 578
ADRXMAIA 579
ADSP (Automatic Data Set Protection) 533, 534
AIX (alternate index), recataloging 316
aliases of non-VSAM data sets 49, 81, 116
ALLLDATA keyword
COPY command 312
copying data set to like device 370
copying data set to unlike device 371
DUMP command 399
dumping records past last-used-block pointer 50, 72
PRINT command 433
specified during a dump 72
to control what DFSMSdss copies 98
with preallocated target 128
ALLEXCP keyword
COPY command 313
copying data set to like device 370
copying data set to unlike device 371
DUMP command 399
dumping records past last-used-block pointer 50, 72, 188
Linux dumps 188
specified during a DUMP 72, 188
to control what DFSMSdss copies 98, 188
with preallocated target 128, 188
ALLOC keyword 26, 266
allocation space 356, 483
ALTER LOCK, IDCAMS command 80
alternate index, recataloging 316
always-call 533, 535
AMDSADMP (SADMP) service aid 519
AMSGCNT keyword 259
application interface
function 30
invoking 29
module names 32
summary
data set processing 604
record processing 603
application program process 619
archive 35, 38
ASIDPTR 577
assignment of class names using the
RESTORE and COPY commands 492
asterisks, in a data set qualifier 264
auditing information 30
authorization checking for
EXPORT/IMPORT (IDCAMS commands) 106
authorization installation exit routine 9
authorization levels for BUILDSA command 526
authorization structure 31
automatic cartridge loader 514
Automatic Data Set Protection
See ADSP
AUTORELADDRESS keyword 82, 121
COPY command 313
RESTORE command 493
auxiliary commands
continuation errors 507
EOJ 508
IF-THEN-ELSE 505
PARALLEL 502
SERIAL 502
SET 503
WETO 501
WTO 501
availability 35
availability management 9
availability strategy, planning 35
avoiding lockout 560

B
backing up (dumping) data 391
backing up HFS data sets 46
backing up volumes
with incremental FlashCopy 62
backspace physical tape record – eioption 25 601
backup
concurrent copy 11, 42, 107
data set 9, 35, 39, 46
DFSMShsm and DFSMSdss 10
disaster recovery 10, 23
incremental 9
integrated catalog facility user catalog 48
migrated data sets 10
reducing time
using volume dump and volume copy 53
scenario 45
SMS-managed data sets 51
SMS-managed environment 36
special requirements 46
system volumes 52
vital records 10, 22, 35
volume 9, 10, 35, 52
backup copies, multiple 372
backup-while-open serialization
CICS 568
concurrent copy 572
DFSMStvs 568
general processing 570
INDEX

CONSOLIDATE command (continued)
  CANCELERROR keyword 288
  DATASET keyword 288
  DEBUG keyword
  FRMSG sub-keyword 288
  TRACE sub-keyword 288
  description 285
diagnosing errors 167
  DYNALLOC keyword 289
diagnosis example 295
  EXCLUDE keyword 290
  FASTREPLICATION keyword 290
  FCTOPPRCPRIORITY keyword 291
  FILTERDD keyword 292
  FOR DFSMSdss 285
diagnosing what DFSMSdss copies 98
  for DFSMSds 285
  FORCECP keyword 292
  INCLUDE keyword 292
  MAXTIME keyword 293
  PASSWORD keyword 293
  performance 155
  PHYSINDDNAME keyword 294
  PHYSINDYNAM keyword 294
  PROCESS keyword 294
  syntax diagram 285
  using 153
  WAIT keyword 294
  WRITECHECK keyword 295

CONSOLIDATE option 155
  CANCELERROR keyword 288
  DATASET keyword 288
  DEBUG keyword
  FRMSG sub-keyword 288
  TRACE sub-keyword 288
  description 285
diagnosing errors 167
  DYNALLOC keyword 289
diagnosis example 295
  EXCLUDE keyword 290
  FASTREPLICATION keyword 290
  FCTOPPRCPRIORITY keyword 291
  FILTERDD keyword 292
  FOR DFSMSdss 285
diagnosing what DFSMSdss copies 98
  for DFSMSds 285
  FORCECP keyword 292
  INCLUDE keyword 292
  MAXTIME keyword 293
  PASSWORD keyword 293
  performance 155
  PHYSINDDNAME keyword 294
  PHYSINDYNAM keyword 294
  PROCESS keyword 294
  syntax diagram 285
  using 153
  WAIT keyword 294
  WRITECHECK keyword 295

CONVERTV command (continued)
  FORCECP keyword 298
  INCAT keyword 297
  NONSMS keyword 298
  PREPARE keyword 298
  RDETERMINE keyword 298
  sample operations 300
  SELECTMULTI keyword 299
  serialization 566
  SMS keyword 299
  syntax diagram 296
  TEST keyword 299
  CONVERTV processing, variables passed to ACS routines 181

COPY command
  changing management class 124
  storage class 124
  CICSVRBACKUP keyword 44
  CONCURRENT keyword 108
  CONVERT keyword
    partitioned data set 319
    partitioned data set extended 319
    data mover selection matrix 105
    for DFSMSds 301
    like devices 12
    logical processing 21
    logical volume 129
    LOGINDYNAM keyword 342
    module name 32
    moving data 12, 98, 116
    physical volume 130
    restriction
      copying an extended-format VSAM data set 118
    SOURCE subkeyword of TGTALLOC 321, 378
    sphere restrictions 118
    unlike devices 12
    used to make CICSVR backups 45
    user catalog 115
    using volume copy
      for reducing backup time 53, 62
      with volume dump 53, 62
    variables passed to ACS routines 179
    VSAM data sets 118
    COPY DATASET, data mover selection matrix 106
    COPY FULL
      freeing the sources, volume 135
      copy operation for logical data set 301
    COPY TRACKS CPVOLUME
      freeing the sources, volume 135
      COPY, access authorization for 547, 551
    COPY command
      explanation 372
      for DFSMSds 372
      INDNNNAME keyword 372, 408
      LOGICALVOLUME keyword 372
      OUTFILENAME keyword 373
      sample operations 373
      syntax diagram 372
    COPYDUMP command, Linux 195
    COPYDUMP, access authorization for 551
    copying multivolume data sets 117
    copying the Stand-Alone Services core image (example) 277
    COPYVOLID keyword 94
    COPY command 320
    CPVOLUME keyword 53, 94, 139
    COPY command 321
    DUMP command 403
    PRINT command 433
    REDETERMINE command 462
    RESTORE command 483
    CREDT keyword 25, 266
    criteria for filtering 24, 263
    critical data sets 37
    Cross-Memory Application Interface
      controlling DFSMSdss
        ASPACE parameter 582
        SNAPX parameter 582
        SRVRTIME parameter 581
    customer program, invocation 44, 62
    CYL subkeyword of TGTALLOC
      COPY command 356
      RESTORE command 483
    Damaged PDS, restoring 85
    DASD (direct access storage device)
      devices supported 17
      initialized 17
      reclaiming space 151
      space fragmentation 155
      space utilization 155
      DASD (direct access storage device) space fragmentation 374
dasfmt utility
      formatting a Linux for zSeries volume 186
    DASDVOL
      access authority 533, 538, 551
      authority 530, 542
      data class 7
      data compaction
        hardware 56
        software 56
      data integrity 23
      considerations
        COPY command 363
        DUMP command 424
        RESTORE command 452
        during data processing 23
        shared DASD considerations 62
      data mover selection matrix for Data Set Copy 106
      data security 164
      data set
        absolute track 79, 81
        backup 9, 35, 39, 46
        broken 24
        changed flag
          DUMP command 417
          characteristics (BY criteria) 24

Index 659
data set (continued)
converting to multivolume in an
SMS-managed environment 86
critical 37
DEFRAG, special 19
direct access 81, 82, 121
dummy 20
erase table for DEFRAG 164
expiration date handling 104
extents, combining 154
filtering 21, 24, 25
GDG 90, 121
HFS 46
indexed sequential 81
last-used-block pointer 50
line operator module names 32
logical dump 37, 40
logical restore 73
message 19
moving 98
multivolume 47, 116
name
fully qualified 24, 264
partially qualified 24, 264
non-SMS-managed 90
organizations 18
partitioned (PDS) 18
physical Dump 40
physical restore 78
profile 74, 125
profiles
discrete 533, 534
generic 533
renaming 102
restoring 72
in an SMS-managed
environment 85
multivolume 79
to multiple target volumes 79
sequential 18
SMS-managed 51, 86, 89
SMS-managed data sets
non-VSAM 490
physical restore actions 490
VSAM 490
special requirements 46, 114
SYS1 system 50
system 115
temporary copied data set 19
temporary names 18, 148
uncataloged 27
undefined DSORG 83
unmovable 81, 82, 120
VSAM 50
with phantom catalog entries 91
data set movement
concurrent copy 107
data set-changed indicator
DFSMsDss handling for a restored
data set 77, 92
event of use 45
using for data set backup 10, 25, 27, 39
using for volume backup 36
DATABASE 2
See DB2 (DATABASE 2) data sets
DATACLAS keyword 25, 267
DATABASE 2
See DB2 (DATABASE 2) data sets
DATALENGTH keyword (PRINT
command) 433
DATASET keyword
CONSOLIDATE command 288
COPY command 321
data mover selection matrix for
copy 106
DUMP command 403
PRINT command 433
RESTORE command 463
with logical processing 21
with physical data set restore 78
DB2 (DATABASE 2) data sets 18
DCB keywords LRECL and BLKSIZE,
specifying in a print operation 435
DDNAME keyword
COMPRESS command 282
CONVERTV command 297
DEFRAG command 377, 380
RELEASE command 443
DDNAME list 575
DDNAME/VOLID Pointer 586
DDPTR parameter 575
DEBUG keyword 259
CONSOLIDATE command
FRMSG sub-keyword 288
TRACE sub-keyword 288
DEBUG/FRMSG
FlashCopy, data sets 111
FlashCopy, volume 132
SnapShot 114, 158
SnapShot, volume 137
decrypting data
hardware considerations 69
software considerations 70
default block size when dumping to tape
or DASD 56
DEFERRED state 90, 122
DEFERRED subkeyword of TGTGDS
COPY command 357
RESTORE command 484
DEFINE function of SAF (system
authorization facility) 529
DEFRAG command 24
access authorization for 551
ADMINISTRATOR keyword 376
BY keyword 377
CONSOLIDATE keyword 377
data set erase table 164
data sets excluded 158
DDNAME keyword 378
DDNAME subkeyword 377
definition 14
DYNALLOC keyword 379
DYNAM keyword 380
EXCLUDE keyword 380
explanation 374
FASTREPLICATION keyword 381
for DFSMsDss 374
FORCECP keyword 383
FRAGMENTATIONINDEX
keyword 383
general hints 160
LIST subkeyword 377
MAXMOVE keyword 383
MAXTIME keyword 384
MMOVPT keyword 385
DEFRAG command (continued)
operation interrupted 19
options 159
PASSDELAY subkeyword 383
PASSWORD keyword 386
performance 155
sample operations 388
serialization 161
syntax diagram 375
VERSION1 keyword 387
WAIT keyword 387
when to run 155
with SnapShot 155
WRITECHECK keyword 388
DEFRAG data set, special 19
DEFRAG/RACF database 159
DELETE keyword
COPY command 325
DUMP command 404
DELETECATALOGENTRY keyword 91
DELETETCATALOGENTRY keyword
(RESTORE command) 463
delleting unwanted data sets 152
description 133
determining version, release, modification
level using the ADRMLVL macro 617
Device Support Facilities utility 17
devices supported 17, 93
DEVTYPE parameter (Stand-Alone
TAPECTL command) 525
DFM 325
DFP segment 537
DFSMsDss
access authorization for
commands 545
backing up Linux for z/Series 185
backup and DFSMShsm 10
BUILDSA command 512
CICSVR 44
commands
BUILDSA 273
CGCREATED 278
COMPRESS 280
CONSOLIDATE 285
CONVERTV 296
COPY 301
COPYDUMP 372
DEFRAG 374
DUMP 391
EOJ 508
IF-THEN-ELSE 504
PARALLEL 502
PRINT 431
RELEASE 440
RESTORE 451
SET 502
Writing to the Operator 501
WTO 501
compatibility with older versions 639
current 29
COPYDUMP command 72
devices supported 17, 93
double encryption 72
filtering 24
function protection 31
DFSMSdss (continued)

interactive 29
invoking 29
invoking with application interface 29
line operators 32
module names 32
module, main entry point 575
overview 7
pointer to address-space-identifier list 577
processing option 584
protecting modules 33
Stand Alone Restore program 512
storage requirements 15
system requirements 15
volume formats supported 17

DFSMSdss commands

BUILDSA 273
CGCREATED 278
COMPRESS 280
CONSOLIDATE 285
CONVERTV 296
COPY 301
COPYDUMP 372
DEFRAG 374
DUMP 391
EOJ 508
IF-THEN-ELSE 504
PARALLEL 502
PRINT 431
RELEASE 440
RESTORE 451
SERIAL 502
SET 503
Writing to the Operator 501
WTOR 501

DFSMSdss DUMP
NEWNAMEUNCONDITIONAL description 41

DFSMSshm and DFSMSdss, backup 10

DFSMSstvs
backup-while-open data sets 568
backup-while-open status definitions 570
direct access
data set moving 121
restoring 82
supported 18
storage device
See DASD (direct access storage device)
direct access data set copying 351
restoring 476
disability 641
disaster recovery 10, 23, 35, 36
disk drives
See DASD (direct access storage device)
disk track
reading – eioption 07 591
writing – eioption 08 591

DO-END group of commands
condition for processing 505
syntax 505
double encryption
processing requests for 71
DSCHA keyword 25, 267
DSORG keyword 26, 267
dummy data set 20, 537
DUMP command 14, 24, 57
access authorization for 551
ADMINISTRATOR keyword 398
ALLDATA keyword 399
ALLEXCPUT keyword 399
BY keyword 400
CANCELERROR keyword 400
CHECKVTOC keyword 400
CICSVRBACKUP keyword 44, 401
COMPRESS command 401, 407
CONCURRENT keyword 43, 402
CPVOLUME keyword 403
data set changed flag 417
data set example 262
DATASET keyword 403
DELETE keyword 404
disaster recovery of logical data sets 37
DYNALLOC keyword 404
ENQFAILURE subkeyword 420
exceptions to hardware data compaction 56
software data compaction 56
EXCLUDE keyword 405
explanation 391
FCWITHDRAW keyword 405
FILTERDD keyword 406
for DFSMSdss 391
FORCECP keyword 407
FULL keyword 407
general description 10
INCAT keyword 407
INCLUDE keyword 408
INDDNAME keyword 408
INDYNAM keyword 409
IOERROR subkeyword 420
logical data set 40
logical data set disaster recovery 37
logical volume 52
LOGINDDNAME keyword 411
LOGININDYNAM keyword 411
module name 32
NEWNAMEUNCONDITIONAL keyword 413
non-VSAM data sets 49
NOVALIDATE keyword 414, 423
ONLYINCAT keyword 407, 414
OPTIMIZE keyword 57, 414
OUTDDNAME keyword 414
PASSWORD keyword 415
performance considerations 57
physical data set 40
physical volume 52
printed output produced by integrated catalog facility user catalog 49
PROCESS keyword 416
PURGE keyword 416
READIOPACING keyword 417
RESET keyword 417
sample operations 424
SELECTMULTI keyword 342, 411
dump conditioned volume 326
dump data, multiple copies 372
dumping
data sets 50
efficiently 53
HFS data sets 46
indexed VSAM data sets 50
integrated catalog facility user catalog 48
multivolume data sets 47
non-VSAM data sets 49
SYS1 system data sets 50
VSAM spheres 49
DYNALLOC keyword
COMPRESS command 282
CONSOLIDATE command 289
COPY command 327
DEFRAG command 379
DUMP command 404
PRINT command 434
RELEASE command 444
RESTORE command 465
DYNALLOC option 159
DYNM keyword
COMPRESS command 282
CONSOLIDATE command 289
COPY command 327
DEFRAG command 379
DUMP command 404
PRINT command 434
RELEASE command 444
RESTORE command 465
dynamic allocation 564
eioption 00 – function startup 588
eioption 01 – reading SYSIN record 589
eioption 02 – printing SYSPRINT record 589
eioption 03 – reading physical tape record 590
eioption 04 – reading logical tape record 590
eioption 05 – writing logical tape record 591
eioption 06 – writing physical tape record 591
eioption 07 – reading disk track 591
eioption 08 – writing disk track 591
eioption 09 – reading utility SYSPRINT 591
eioption 10 – writing SYSPRINT record 592

E
integrated catalog facility user catalog (continued)
  See JCL (job control language)

job control language (continued)
  See JCL (job control language)

K
  key sequenced data sets, supported 18
  keyboard 641

L
  LASTCC
    definition 503
    in IF-THEN-ELSE command sequence 505
    in SET command 504

  LDS, supported 18
  LE operator 26, 267
  like devices, moving volumes to 138
  line operators, DFSMSdss/ISMF 32

  Linux copy
    COPYDUMP command 195
    full volume 195

  Linux dumps
    ALLEXCP keyword 189
    DATASET keyword 189
    FULL keyword 189
    using concurrent copy 189

  Linux for zSeries
    authorization 187
    backing up with partitions 185
    DFSMSdss command BUILD STAND-ALONE 185
    COPY FULL 185
    COPY FULL COPYVOLID ALLEXCP 185
    COPYDUMP 185
    DUMP DATASET 185
    DUMP FULL 185
    DUMP FULL with CONCURRENT COPY 185
    RESTORE DATASET 185
    RESTORE FULL 185
disk utility
    dasdfmt 186
    fdasd 187

  hardware environment 185
  restoring partitions 185
  submitting JCL batch jobs to z/OS
  using FTP 195
  volume serial rules 186
  LIST subkeyword
    DEFRAG command 377, 380

  location-dependent data 73, 81
  locking a user catalog 80
  lockout, avoiding 560, 561

LOGDDNAME keyword (RELEASE command) 445
  logging on to ISMF 29

  tape record, reading – eioption 04 590
  tape record, writing – eioption 05 591

  VSAM record, insert during restore – eioption 18 594

  logical COPY 97, 129
  logical data set dump of indexed VSAM data sets 50
  logical data set restore 73
  logical dump volume 52
  logical processing 21
  logical restore 85

  cataloging data sets 75
  data sets with phantom catalog entries 91
  preformatted empty VSAM data set 91
  renaming data sets 76
  user catalog aliases 80
  without preallocated targets 81

  LOGICALVOLUME keyword 78
  COPYDUMP command 372
  RESTORE command 470

  LOGINDDNAME keyword
    COPY command 342
    DUMP command 411
    logical processing 21

  LOGINDYNAM keyword
    COPY command 342
    DUMP command 411
    logical processing 21

  LRECL and BLKSIZE keywords, specifying in a print operation 435
  LT operator 26, 267

M
  magnetic tape devices supported 17

  mainframe
    education xvi
    maintenance 579

  MAKEMULTI keyword
    COPY command 344
    RESTORE command 470
    management class 7

    changing with copy 124
    changing with RESTORE 88

    mapping macro, ADREID0 604
    MAXCC keyword
      definition 503
      in IF-THEN-ELSE command sequence 505
      in SET command 504
    MAXMOVE keyword
      changing track utilization by
      reblocking 165
      MAXMOVE keyword 159
      MAXMOVE keyword (DEFRAG command) 383
      MAXSIZE variable 182
    MAXTIME keyword
      CONSOLIDATE command 293
MAXTIME keyword (DEFRAG command) 384
MENTITY keyword
COPY command 344
RESTORE command 470
message data set 19, 535
MGMTCLAS keyword 26, 87
COPY command 345
RESTORE command 471
MGMTCLAS profile 267, 537
migrated data sets, backup 10
minivolumes 17
MINSECQTY keyword 151
COPY command 345
RESTORE command 471
MGMTCLAS profile 267, 537
Migrated data sets, backup 10
MINSECQTY keyword (RELEASE command) 447
MINTRACKSUNUSED keyword 151
MINTRACKSUNUSED keyword (RELEASE command) 447
MMOVPCT keyword (DEFRAG command) 385
MMOVPCT option 160
mode switching 502
module names for data set application commands 32
moving catalogs 115
damaged PDS 119
data in an SMS-managed environment 12
data sets 98
preformatted empty VSAM 129
special requirements 114
utilities 105
data sets to unlike devices 120
direct access data set 121
empty non-VSAM data sets 114
generation data sets to non-SMS-managed volumes 122
to SMS-managed volumes 122
multivolume data set 116
non-VSAM data sets that have aliases 49, 81, 116
PDSE 119
SMS-managed data sets 123
system data sets 115
to preallocated data set 125
undefined DSORG data sets 114
unmovable data sets 120
volumes
logical volumes 129
physical volumes 130
to devices of equal capacity 138
to devices of greater capacity 138
to unlike devices 139
VM-format volumes 139
VTOC considerations 129
moving data
with concurrent copy 13
with FlashCopy 13
with SnapShot 13
moving data sets 261, 325
moving volumes
with FlashCopy 131
with native SnapShot 137
MULTI keyword 26, 267
multiple backup copies 372, 415
multiple subkeywords 254
multiple target volumes
restoring 79
specifying 127
multiple tracks read, on dump 414
multivolume data set 116
conversion 146, 148
copying 118
discrete profile 537
dumping 47
restoring 118
multivolume VSAM data sets 86, 118
MVOLSER keyword
COPY command 344
RESTORE command 470
MVS environments supported 15

N
names, data set 24
NE operator 26, 267
nested IF commands 502
ONLYINCAT keyword
COPY command 340
DUMP command 414
RELEASE command 445
OPEN/EOV
nonspecific tape volume mount – eioption 17 593
tape volume security and verification exit – eioption 16 593
operating environment 15
operating in serial or parallel mode 502
OPERATIONS attribute 529, 530
OPERCNSL parameter (Stand-Alone BUILDSA command)
limited validation of 511
purpose of 511
specifying 275
OPTIMIZE keyword 57
OPTIMIZE keyword (DUMP command) 414
option list 575
options for non-VSAM data sets, enqueue exit routine 562
options installation exit routine 9
OPTPRT parameter 575
organizations, data set 18
original record length 586
original record pointer 586
OUTDDNAME keyword 73, 101
COPY command 346
COPYDUMP command 373
DUMP command 414
PRINT command 435
RESTORE command 472
OUTDYNAM command 346
COPY command 347
RESTORE command 472
OUTDYNAM parameter (Stand-Alone BUILDSA command) 276
OUTDDNAME keyword 73, 101
output from DFSMSdss (in DD statement) 257
output tape I/O error – eioption 19 594
output volume selecting 101
specifying 73, 78, 93
OUTTRACKS keyword
COPY command 347
RESTORE command 473
overview of DFSMSdss 7
overview of the Stand-Alone Services process 514
null command 506
NULLMGMTCLAS keyword 87
COPY command 345
RESTORE command 471
NULLSTORCLAS keyword 87
COPY command 355
RESTORE command 483

O
ONLYINCAT keyword
COPY command 340
DUMP command 414
RELEASE command 445
OPEN/EOV
nonspecific tape volume mount – eioption 17 593
tape volume security and verification exit – eioption 16 593
operating environment 15
operating in serial or parallel mode 502
OPERATIONS attribute 529, 530
OPERCNSL parameter (Stand-Alone BUILDSA command)
limited validation of 511
purpose of 511
specifying 275
OPTIMIZE keyword 57
OPTIMIZE keyword (DUMP command) 414
option list 575
options for non-VSAM data sets, enqueue exit routine 562
options installation exit routine 9
OPTPRT parameter 575
organizations, data set 18
original record length 586
original record pointer 586
OUTDDNAME keyword 73, 101
COPY command 346
COPYDUMP command 373
DUMP command 414
PRINT command 435
RESTORE command 472
OUTDYNAM command 346
COPY command 347
RESTORE command 472
OUTDYNAM parameter (Stand-Alone BUILDSA command) 276
OUTDDNAME keyword 73, 101
output from DFSMSdss (in DD statement) 257
output tape I/O error – eioption 19 594
output volume selecting 101
specifying 73, 78, 93
OUTTRACKS keyword
COPY command 347
RESTORE command 473
overview of DFSMSdss 7
overview of the Stand-Alone Services process 514
page ejection, suppressing 259
page-number list 576
PAGENO keyword 259
PAGEPTR parameter 576
PARALLEL command 15, 502
PARALLEL keyword 57
PARAM keyword 577
PARM information in the EXEC statement 258
partially qualified data set names 24, 264
partitioned data set
See PDS (partitioned data set)
partitioned data set extended
See PDSE (partitioned data set extended)
PASSDELAY option 160
PASSDELAY subkeyword (DEFRAG command) 383
password DD, JCL statement 258
PASSWORD keyword
COMPRESS command 284
CONSOLIDATE command 293
COPY command 347
DEFRAG command 386
DUMP command 415
PRINT command 435
RELEASE command 447
RESTORE command 473
password protection 532
PATCH parameter definition 503
PDS (partitioned data set) 18
abnormal conditions 119
compressing 152
monitoring PDSs during compression 119
moving damaged 119
recovery by DFSMSdss during compression 119
restoring damaged 85
storage requirements 17
translation during compression 120
PDS subkeyword (COPY command) 319
PDSE (partitioned data set extended) 18
abnormal conditions 85
compressing 152
monitoring PDSEs during compression 119
moving damaged 119
recovery by DFSMSdss during compression 120
restoring damaged 85
storage requirements 17
supported 18
translation during compression 120
PDS subkeyword (COPY command) 319
PDSE (partitioned data set extended) 18
abnormal conditions 85
compressing 152
monitoring PDSEs during compression 119
moving damaged 119
recovery by DFSMSdss during compression 120
restoring damaged 85
storage requirements 17
supported 18
translation during compression 120
PERMIT command 33
phantom catalog entries 91
physical copy 97, 130
physical data set copy
performing 100
using a conditioned volume 100
physical full volume copy
FCINCREMENTAL keyword 62
FCINCREMENTALLAST keyword 63
FCINCRVERIFY keyword 63
FCWAIT keyword 63
physical processing 21, 22
physical restore 78, 80, 85
INITIAL status 89
output volume selection 78
physical restore of SMS-managed data sets 89
physical tape record
backspace – eioption 25 601
reading – eioption 03 590
writing – eioption 06 591
physical volume dump 52
PHYSINDD 415
PHYSINDDNAME keyword
CONSOLIDATE command 294
PHYSINDYNAM 416
PHYSINDYNAM keyword
CONSOLIDATE command 294
planning an availability strategy 35
preallocated data set
COPY command 125
moving to 125
REPLACE keyword 74
REPLACEUNCONDITIONAL keyword 74
restoring to 74
preallocated targets
restoring 82
restoring without 81
preallocation
non-VSAM 126
VSAM 125
PREPARE keyword 14, 144
PREPARE keyword (CONVERTV command) 298
preparing for Stand-Alone Services 271, 526
presenting
ADRUFO record – eioption 13 593
WTOR response – eioption 15 593
PRINT command
ADMINISTRATOR keyword 433
ALDDATA keyword 433
CPVOLUME keyword 433
data-length keyword 433
DATASET keyword 433
dynalloc keyword 434
ENQFAILURE subkeyword 436
ERRORTRACKS keyword 434
explanation 431
for DFSMSdss 431
INDDNAME keyword 434
INDYNAM keyword 434
IOERROR subkeyword 436
KEYLENGTH keyword 435
OUTDDNAME keyword 435
PRINT command (continued)
PASSWORD keyword 435
sample operations 438
SHARE keyword 436
syntax diagram 431
tolerate keyword 436
TRACKS keyword 436
VTOC keyword 437
WAIT keyword 438
PRINT, access authorization for 552
printers supported 17
printing SYSPRINT record – eioption 02 589
PROCESS keyword 114, 125
CONSOLIDATE command 294
COPY command 349
DUMP command 416
RELEASE command 448
RESTORE command 474
processing
logical 21, 22
physical 21, 22, 23
protect-all 535
protected catalogs 544
group data set 533
user data set 533
protecting data sets 531
passwords 532
resources 531
SMS-managed data sets 537
usage of DFSMSdss 531
protection
DFSMSdss function 31
DFSMSdss/ISMF modules 33
functions with RACF 31
ISMF access 31
keywords with RACF 33
PURGE keyword
COPY command 350
DUMP command 416
RESTORE command 475
Q
qualified data set names 264
R
RACF (Resource Access Control Facility)
authorization checking 344, 471
data set erase table for DEFRAG 164
database 159
DEFINE function 529
keyword profiles 33
logging 538
protecting keyword modules 33
protecting keywords 33
RACF-indicated, meaning of 534
RACF-protected, meaning of 529
RAMAC Virtual Array (RVA) 13, 43, 59,
113
RDEFINE command 33
read I/O pacing, performance considerations 61

z/OS V1R13.0 DFSMSdss Storage Administration
RESTORE command (continued)
ROLLEDOFF subkeyword of TGTGDS 484
SAM compressed data set 90 sample operations 493
SHARE keyword 482
SOURCE subkeyword of TGTALLOC 483
SOURCE subkeyword of TGTGDS 484
special considerations 452
SPHERE keyword 482
STORCLAS keyword 483
syntax diagram 453
TGTALLOC keyword 483
TGTGDS keyword 484
TOLERATE keyword 485
TRACKS keyword 485
TRK subkeyword of TGTALLOC 483
TTRADDRESS keyword 486
UNDEFINEDSORG subkeyword 474
unmovable data sets 81
variables passed to ACS routines 181
VOLCOUNT keyword 486
WAIT keyword 488
WRITECHECK keyword 489

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
restoring backups created with an earlier release 639
damaged PDSE 85
data sets 72
direct access data sets 82
extended-format VSAM data set stripe count of one 83
GDG data sets 90
in an SMS-managed environment 86
multivolume data sets 118
non-SMS-managed data set 90
PDSE 85
preallocated targets 74, 82
preformatted empty VSAM data set 91
SMS-managed data sets 86, 89
SMS-managed GDG data sets 90
undefined DSORG data set 83
volumes 92
VSAM cluster 84
VSAM sphere 83
without preallocated targets 81
restoring volumes with incremental FlashCopy 62
restriction copying an extended-format VSAM data set stripe count of one 118

REWIND parameter (Stand-Alone TAPECNTL command) 525
RIOP keyword 304, 307, 311
RLS quiesce processing 51, 128
ROLLED-OFF state 90, 122
ROLLEDOFF subkeyword of TGTGDS COPY command 357
RESTORE command 484
RRDS data sets, supported 18
running Stand-Alone Services in 370 mode 509
inXA or ESS mode 510
program 514
under VM 510
with a predefined console 511
RVA 318
RVA (RAMAC Virtual Array) 13, 43, 59, 113

SHARE keyword
COPY command 354
DUMP command 418
PRINT command 436
RUNNING command 482
SHARE keyword, with backup function for HFS 46
shortcut keys 641
simulating conversion 143
SIZE keyword 259
SIZE variable 182
SMS authorization 545
keyword (CONVERTV command) 299
managed data sets 537
SMS conversion eligibility 142
ineligibility 141, 142
SMS management conversion by data movement 142
collection from, by data movement 143
collection to and from 13
collection without data movement 145
converting from, without data movement 148
converting to and from 141
SMS Report 146
SMS-managed data sets backup 51
moving 123
physical RESTORE 89
restoring 86
restoring, GDG data sets 90
SMS-managed environment backup 36
moving data 12
Non-SMS-managed data 36
recovery 36
restoring data sets 85
SMS-managed data 36
SMS-managed volumes, moving data sets to 122
MSGCNT keyword 259
SnapShot 13
authorization checking 137
DEBUG(FRMS) keyword 114, 158
DEBUG(FRMSG) keyword, volume 137
in conjunction with physical full volume copy 53
moving data sets with SnapShot 113
moving volumes with native SnapShot 137
native mode 13, 113
problem solving 114, 158
problem solving, volume 137
RAMAC Virtual Array (RVA) 43, 108
reducing backup time 53, 87
virtual concurrent copy mode 43, 87
virtual concurrent copy mode 43, 107, 108, 113
SOURCE subkeyword of TGTALLOC COPY command 357
RESTORE command 483, 484
Stand-Alone Services feature (continued)
core image (continued)
specifying the output DASD volume 276
DASD devices in 370 mode 509
devices in XA or ESA mode 510
target device address, specifying 521
dump data set device address, specifying 521
device type, specifying 521
examples
BUILDSA command 276
copying the Stand-Alone Services core image 277
IPL 517
RESTORE command 523
TAPECNTL command 526
operator console potential problems 511
predefining 509, 510, 511
optional parameters
BUILDSA command 274
RESTORE command 521
predefined console potential problems with 519
purpose of 511
required parameters
BUILDSA command 274
RESTORE command 521
TAPECNTL command 525
RESTORE command 522
description 520
ENDTRK parameter 522
example 523
FILE parameter 523
FROMADDR parameter 521
FULL parameter 522
NOREADCHECK parameter 523
NOVERIFY parameter 522
optional parameters
READCHECK parameter 523
required parameters 521
STARTTRK parameter 522
syntax 521
TAPEVOLSER parameter 523
TOADDR parameter 521
VERIFY parameter 522
specifying information for the Stand-Alone Services core image 274
the DASD target device address 521
the dump data set device address 521
the dump data set device type 521
the input device 516
the JCL output location for IPL-able core image 276
the message output device 516
the output DASD volume for IPL-able core image 276
where Stand-Alone Services is IPLed from 273
stand-alone restore program overview 12
physical processing 23
system volumes, backing up 23
system volumes, recovering 94
used to restore Linux volumes 195
VM, running on 94
volume dump, physical 52
Stand-Alone Services feature
BUILDSA command
ADMINISTRATOR parameter 274, 527
determining authorization level 526
examples 276
INDDNAME parameter 274
IPL parameter 274
OPERCNSL parameter 275
optional parameters 274
OUTDDNAME parameter 276
OUTDYNAM parameter 276
required parameters 274
syntax 274
commands
BUILDSA 273
RESTORE 520
TAPECNTL 525
core image definition of 271, 526
specifying information for 274
specifying the JCL output location for 276
subkeywords
  maximum number 254
  multiple 254
  specifying 254
  supported
  platforms for Stand-Alone
  Services 509
  supported devices 18, 93
  suppressing page ejection 259
  switching modes 502
syntax diagram
  CONSOLIDATE command 285
  CONVERTV command 296
  COPY command 302
  COPYDUMP command 372
  DEFRAG command 374
  DUMP command 392
  PRINT command 431
  reading a 255
  RELEASE command 441
  RESTORE command 453
SYS1 subkeyword
  COPY command 349
  DUMP command 416
  RELEASE command 448
SYS1.ANMTMAIN.SNAPnnnn data
  sets 59
SYSALLDA 98
SYSDA 98
SYST
  DD statement, JCL 257
  record, reading – eioption 01 589
SYSPRINT
  DD statement, JCL 257
  reading utility – eioption 09 591
  record printing – eioption 02 589
  writing record – eioption 10 592
system
  authorization facility
    See SAF
  programming information 582
system consoles supported 17
system data mover (SDM) 59
system data set
  dumping 50
  moving 115
system environment 15
system requirements 15
system volumes
  backing up 52
  recovering 94
T
  tape
    libraries not supported in 370
    mode 509
    mount and demount procedures 512
    output I/O error – eioption 19 594
    physical record backspace – eioption 25 601
    reading logical record – eioption 04 590
    reading physical record – eioption 03 590
    volume mount OPEN/EVOC
      nonspecific – eioption 17 593
tape (continued)
  volume security OPEN/EVOC –
    eioption 16 593
  writing logical record – eioption
    05 591
  writing physical record – eioption
    06 591
  tape devices (with Stand-Alone Services)
    in 370 mode 509
    in XA or ESA mode 510
  tape devices supported 17
  TAEVOLSER parameter (Stand-Alone
    RESTORE command) 523
  target allocation 483
  target class selection 492
  target definition 257
  target volume 12, 123
  task processing, controlling 502
  TASK-ID 583
  temporary
    data set 536
    data set names 535
    message data set 535
    temporary copied catalogs 20
    temporary data set 19, 148
    temporary data set names 18
    temporary work space 98
  TEST keyword 144
  TEST keyword (CONVERTV
    command) 299
  TGTALLOC keyword
    COPY command 356
    RESTORE command 483
  TGTGDS keyword
    COPY command 357
    RESTORE command 484
  THEN command 505
  TMPMSGDS keyword 260
  TOADDR parameter (Stand-Alone
    RESTORE command) 521
  TOL(ENQF) keyword, with backup
    function for HFS 46
  TOLERATE keyword
    COPY command 358
    DUMP command 420
    PRINT command 436
    RESTORE command 485
    TRACKS keyword
    COPY command 358
    DUMP command 421
    physical processing 22
    PRINT command 436
    RESTORE command 485
    tracks read, on dump 414
    TRK subkeyword of TGTALLOC
    COPY command 356
    RESTORE command 483
  TSO FCWITHDRAW (continued)
    withdrawing FlashCopy relationship,
    volume 132
  TTRADDRESS keyword 83, 121
  U
  UAPTR parameter 577
  UIMPTR parameter 576
  UNCATEGORIZED keyword
    COPY command 359
    DUMP command 422
    uncataloged data set 27
    undefined DSORG data sets
      moving 114
      restoring 83
  UNDEFINEDSORG subkeyword
    COPY command 349
    RESTORE command 474
  UNITADDR parameter (Stand-Alone
    TAPECNTL command) 525
  universal access authority (UACC) 31
  unlike devices 130
    moving volumes 139
  UNLOAD parameter (Stand-Alone
    TAPECNTL command) 526
  unloading a tape with the TAPECNTL
    command 525
  unmovable data set
    moving 120
  REPLACE keyword 82
  REPLACEUNCONDITIONAL
    keyword 82
  restoring to preallocated targets 82
  unused space, releasing 151
  unwanted data sets, deleting 152
  USEEXCP keyword 260
  User
    area list 577
    Area Pointer 586
    Exit Allowance 583
    Return Code 584
  user catalog, moving 115
  User Interaction Module (UIM)
    example 633
    list 576
    sample output 633
  user interaction module exit
    functions 30
  utilities
    in data set copy operation 105
    moving data sets 105
    utilization, percent
      COPY command 348
      RESTORE command 474
    UTILMSG keyword 260
  V
  VALIDATE keyword 50
  record counting 8
VALIDATE keyword (DUMP command) 423
variable length (VL) parameter 577
verification
  bypass exit – eioption 22 595
  data set – eioption 21 595
  exit, OPEN/EQU – eioption 16 593
VERIFY parameter (Stand-Alone RESTORE command) 522
VERSION1 keyword 160
DEFRAG command 387
versions of DFSMSdss, compatibility 639
virtual concurrent copy 318
declared 11
  Snapshot 11
virtual concurrent copy mode 13
virtual concurrent copy working space 59
virtual input/output (VIO) device supported 17
virtual storage access method
  See VSAM (virtual storage access method)
vital records 10, 22, 35, 38
VL parameter 577
VM minivolumes 43
VM-format volumes
  backing up 53
  moving 139
  recovering 94
  with DFSMSdss 18
VM, running Stand-Alone Services under 510
VOLCOUNT keyword
  COPY command 360
  RESTORE command 486
volume
  backup 9, 35, 52
  broken 23
  conversion without data movement 14
  copy and dump combining 53
  DUMP with concurrent copy 425
  formats supported by DFSMSdss 17
  input, specifying 99, 100
  line operator module names 32
  logical copy operation 129
  logical dump 52
  multiple target 127
  multiple target, restore 79
  notification – eioption 20 594
  output, selecting 101
  output, selection 73, 78
  output, specifying 93
  physical copy 130
  physical dump 52
  preparing for conversion 144
  selecting target 123
  serialization 559
  system, recovering 94
  VM-formatted 18
  volume consideration 235
  volume copy
  reducing backup with volume dump 53
volume dump
  reducing backup with volume copy 53
volume list, ISMF 97
volume serial rules
  for Linux volumes 186
volumes
  using incremental FlashCopy 62
VSAM (virtual storage access method)
  cluster, restoring 84
  data set
    converting 86, 118
    copying 118
    dumping 50
    moving 118
    moving preformatted empty 129
    restoring preformatted empty 91
    storage requirements 17
  preallocation 125
  sphere
    dumping 49
    eligibility 146
    restoring 83
VSAM (Virtual Storage Access Method)
  data set
    filtering 263
    restore actions on 490
VTOC
  indexed 17
  keyword (PRINT command) 417, 437
  nonindexed 17
  VTOC Index 18

W
WAIT keyword 160
  COMPRESS command 284
  CONSOLIDATE command 294
  COPY command 362
  DEFRAG command 387
  DUMP command 423
  PRINT command 438
  RELEASE command 449
  RESTORE command 488
WAIT option 561, 567
wait-state codes for Stand-Alone program 518
when to use
  logical processing 22
  physical processing 23
  work space 98
  working space data set 59
WORKUNIT keyword 260
WORKVOL keyword 260
WRITECHECK keyword
  CONSOLIDATE command 295
  COPY command 363
  DEFRAG command 388
  RESTORE command 489
writing
  disk track – eioption 08 591
  logical tape record – eioption 05 591
  physical tape record – eioption 06 591
  SYSPRINT record – eioption 10 592
to the operator 501
WTO message – eioption 11 592
WTO message writing – eioption 11 592
WTOR
  command 501
  message writing – eioption 12 592, 593
  command for DFSMSdss 501

X
XABUFF keyword 261

Z
z/OS Basic Skills information center xvi
ZBUFF64R keyword 261
zFS data sets
  DUMP command considerations 418
  logical data set COPY 301
  logical dump and serialization 563
  physical dump and serialization 563
  read/write serialization scheme 564
  RELEASE command consideration 441