

z/VM  
7.2

*CMS File Pool Planning, Administration,  
and Operation*



**Note:**

Before you use this information and the product it supports, read the information in [“Notices” on page 719.](#)

This edition applies to version 7, release 2 of IBM z/VM (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2023-12-05

© **Copyright International Business Machines Corporation 1990, 2023.**

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

---

# Contents

<b>Figures.....</b>	<b>xiii</b>
<b>Tables.....</b>	<b>xvii</b>
<b>About This Document.....</b>	<b>xxi</b>
Intended Audience.....	xxi
Syntax, Message, and Response Conventions.....	xxi
Where to Find More Information.....	xxiv
Links to Other Documents and Websites.....	xxiv
<b>How to provide feedback to IBM.....</b>	<b>xxv</b>
<b>Summary of Changes for z/VM: CMS File Pool Planning, Administration, and Operation.....</b>	<b>xxvii</b>
SC24-6261-02, z/VM 7.2 (December 2023).....	xxvii
SC24-6261-02, z/VM 7.2 (December 2021).....	xxvii
SC24-6261-02, z/VM 7.2 (September 2021).....	xxvii
SC24-6261-02, z/VM 7.2 (July 2021).....	xxvii
SC24-6261-02, z/VM 7.2 (March 2021).....	xxvii
SC24-6261-01, z/VM 7.2 (September 2020).....	xxvii
SC24-6261-00, z/VM 7.1 (September 2018).....	xxvii
<b>Part 1. Administering Repository File Pools.....</b>	<b>1</b>
Chapter 1. File Pool Administration Overview.....	3
File Pool Servers and Possible Combinations.....	3
Types of File Pool Data Repositories.....	5
What Is a File Pool?.....	6
How Users Access Their Files.....	7
Data Repository Services.....	9
Coordinated Resource Recovery (CRR) Services.....	12
FIFO Services.....	12
Servers and File Pools.....	13
File Pool Contents.....	13
File Pool Control Data.....	13
File Pool Log Data.....	14
File Pool Repository Data.....	17
File Pool Server Machines.....	18
Multiple Production File Pools.....	23
Step 1: Estimate Maximum File Pool Size.....	23
Step 2: Decide on Additional Production File Pools.....	24
Step 3: Enroll Users in File Pools.....	24
Step 4: Implement Multiple File Pools.....	24
File Pool Administration Machines.....	25
A Note about File Pool Maximums.....	28
Other File Pool Facilities.....	28
Accounting Facility.....	29
Security Audit Trace.....	29
External Security Manager.....	29

Service Aids.....	29
What to Do Next.....	30
Chapter 2. Installation Planning for File Pools.....	31
VMSYS and VMPSFS.....	31
BFS Root File Space in the VMSYS File Pool (Default).....	31
BFS Root File Space in a Non-default File Pool.....	34
VMSYSU.....	34
VMSYSU Extensions to the BFS Root.....	35
VMSYSR.....	35
Chapter 3. Post-installation Activities.....	37
VMSYS, VMPSFS, VMSYSU, and VMSYSR.....	37
Characteristics of VMSYS.....	37
Characteristics of VMSYSU.....	39
Characteristics of VMPSFS.....	40
Characteristics of VMSYSR.....	40
Tailoring VMSYS, VMPSFS, VMSYSU, and VMSYSR.....	40
Tailoring CMS.....	49
Chapter 4. Migration Considerations.....	51
File Pool Startup Parameters.....	51
Backing Up the File Pool.....	51
Use of the Byte File System (BFS).....	51
LU Name of CRR Recovery Server (CRR Only).....	51
Data Space Exploitation (SFS Only).....	52
File Space Storage Use Exits.....	52
Catalog Reorganization.....	52
DFSMS/VM Exploitation.....	52
What to Read Next.....	53
Chapter 5. Operation.....	55
Starting Multiple User Mode Processing.....	55
Monitoring Server Operation.....	56
Automatic Backups.....	56
Forcing a Server User.....	57
Forcing Prepared Work.....	57
Periodic Catalog Maintenance.....	58
Limit Monitoring.....	58
Performance Monitoring.....	65
Stopping Server Processing.....	65
Control Data Backups at Shutdown.....	66
Other Ways to Stop Multiple User Mode Processing.....	67
Changing the File Pool ID.....	67
Restricting User Access in Multiple User Mode.....	69
Chapter 6. Managing Users and File Spaces.....	71
Enrolling SFS Users and Creating BFS File Spaces.....	71
Step 1: Determine Amount of Space to Assign Each File Space.....	72
Step 2: Determine Storage Groups to Assign File Spaces.....	73
Step 3: Decide What Will Be Accessed as File Mode A (SFS Only).....	73
Step 4: Update the User's z/VM System Directory Entry.....	74
Step 5: Log On an Administration Machine.....	76
Step 6: Determine Whether There Is Enough Physical Space.....	76
Step 7: Enter an ENROLL USER Command.....	76
Step 8: Do Optional Activities for Existing Users (SFS only).....	78
Step 9: Back Up the Storage Group.....	80
Step 10: Consider Increasing USERS and MAXCONN for Server.....	80



Establishing Clients for BFS File Spaces.....	80
Example of Mounting a File System.....	84
External Links of Type Mount (MEL).....	89
Changing the Amount of Space Assigned to a File Space.....	91
Changing a User's Threshold Value (SFS Only).....	92
Checking to See Who Is Enrolled (SFS) or Which File Spaces Exist.....	92
Determining Space Assigned to a File Space.....	92
Enrolling PUBLIC.....	92
Deleting Enrolled Users and File Spaces.....	93
Deleting Everyone Enrolled by ENROLL PUBLIC.....	94
Overview of Moving Users and File Spaces.....	95
Required Tasks Before Moving Users or File Spaces.....	95
Moving an SFS User or BFS File Space in Multiple User Mode.....	95
Moving an SFS User in Dedicated Maintenance Mode.....	96
Renaming File Spaces.....	99
Chapter 7. Recovery Procedures.....	101
Backing Up a File Pool - Overview.....	101
Backing Up The Control Data.....	104
Enabling Control Data Backup Processing.....	104
Control Data Backups at Shutdown.....	104
Control Data Backups Using FILESERV BACKUP.....	104
Control Data Backups Using the BACKUP Operator Command.....	105
Control Data Backups Using FILEPOOL CONTROL BACKUP.....	105
Control Data Backups Started Automatically.....	105
DASD Space Needed for Control Data Backup.....	105
Special Considerations for Control Data Backups to File Pool.....	105
Special Considerations for Control Data Backups To Tape.....	106
Special Considerations for Control Data Backups to CMS File.....	106
Frequency of Control Data Backup.....	106
Disabling Control Data Backup Processing.....	107
Defining or Changing the Default Control Data Backup File.....	107
Recovery from Control Minidisk Verification Errors.....	109
Restoring Control Data.....	110
What If a Control Data Restore Is Unsuccessful?.....	111
Restoring Repository Log Data.....	111
Backing Up User Data.....	111
Using FILEPOOL UNLOAD.....	112
Using FILEPOOL BACKUP.....	113
Back Up Considerations if Storage Group Managed by DFSMS/VM.....	115
DASD Needed for Backing Up User Data.....	115
Concurrent User Data Backups.....	115
Staging User Data Backups.....	115
Restoring User Data.....	115
From FILEPOOL UNLOAD for User Storage Group.....	116
From FILEPOOL UNLOAD for File Space.....	117
From FILEPOOL BACKUP.....	118
Concurrent User Data Restores.....	120
SFS Alias Consideration After Restoring User Storage Group (SFS only).....	120
Incremental Availability During Multiple FILEPOOL RESTORES.....	122
Using SFS Unresolved Aliases (SFS only).....	122
Finding Corrupted Files When a Minidisk is Corrupted.....	123
Considerations if Storage Group is Managed by DFSMS/VM.....	123
Restoring Individual Files.....	123
Restoring Individual Files in DFSMS/VM Migrated Status.....	127
Considerations for SFS Aliases After FILEPOOL FILELOAD.....	127
Considerations for SFS Aliases After FILEPOOL RELOAD FILES.....	127
Restoring a File Pool by Generating It Again.....	127

Replacing File Pool Minidisks.....	129
Replacing One File Pool Log Minidisk.....	131
Replacing a Control Minidisk or Storage Group 1 Minidisk.....	132
Replacing Storage Group 2 through n Minidisks.....	132
Replacing the Work Minidisk Containing POOLDEF File.....	132
Replacing Both File Pool Log Minidisks.....	133
Moving File Pool Minidisk Data Using DFSMS/VM.....	134
Using Non-File Pool Facilities for Backing Up Your File Pool.....	134
Chapter 8. Security.....	137
File Pool Authorizations and Permissions.....	137
Controlling Access to a File Pool Server Machine.....	139
Administration Authority.....	139
Enrolling an Administrator.....	141
Deleting an Administrator.....	142
Deleting a User or File Space.....	143
Auditing Security.....	143
Using an External Security Manager.....	161
What Is a Security Manager?.....	162
What an ESM Does Not Protect.....	162
How an ESM Affects Users.....	163
DMSESM PROFILE.....	164
Getting an External Security Manager to Interface with SFS or BFS.....	169
General Requirements for an ESM.....	169
File Pool Server Interface to an ESM.....	171
Writing Your Own ESM Exit Routines.....	176
ESM User Data (SFS ESM).....	189
Chapter 9. Managing Storage.....	195
Rearranging Existing Storage Groups.....	195
User Storage Groups.....	195
Catalog Storage Groups.....	195
Overview of Adding DASD Space to a File Pool.....	195
Required Tasks Before Adding Minidisks.....	196
Adding Minidisks in Multiple User Mode.....	199
Adding Minidisks in Dedicated Maintenance Mode.....	201
Creating a Minidisk Definition Control Statement File.....	202
Getting Information about Space in a File Pool.....	204
Moving File Pool Minidisks to Different Physical Devices.....	204
Moving a File Pool.....	204
Removing Space from a File Pool.....	206
Setting the Warning Threshold.....	208
Using DFSMS/VM for Storage Management.....	208
Activating the Storage Use Exits (SFS and BFS).....	209
SFS General Data Buffer.....	209
Activating the Sample Exit Functions.....	210
Modifying or Replacing the Exit Routine.....	210
Making the New or Modified Exit Routine Available.....	211
Chapter 10. Reorganizing the File Pool Repository Catalogs.....	213
Chapter 11. Regenerating a Repository File Pool.....	217
What to Do If Something Goes Wrong.....	221
Chapter 12. Accounting.....	223
Starting the Accounting Facility.....	223
Stopping the Accounting Facility.....	224
Accounting Record Formats.....	224

Initialization Record.....	225
Operator and Checkpoint Records.....	226
System Record.....	227
Termination Record.....	227
User Record.....	228
Customizing User Accounting Records.....	229
Writing Your Own Accounting Exit Routine.....	229
Making Your Accounting Exit Routine Available to CMS.....	230
Multiple User ID Support Considerations.....	231
Chapter 13. Setting Up a File Pool for Remote Use.....	233
Making Your File Pool Available to Other Processors.....	233
Restricting Access from Other Processors.....	236
Enrolling Remote Users.....	236
Using Nicknames and Local IDs for Remote Users.....	237
Coding a Program to Supply Local IDs for Remote Users.....	238
Nickname Resolution for SFS Commands.....	239
Coding the DMSJNE Routine.....	239
Installing the DMSJNE Routine.....	240
Chapter 14. Using Data Spaces (SFS Repository Servers Only).....	241
Data Spaces.....	241
Finding Data Space Candidates.....	241
Selecting a File Pool.....	242
Configuring Your Server for Data Spaces.....	242
Restrictions for Using Data Spaces on Extended Address Volume Minidisks.....	244
Restrictions for Using Data Spaces on FBA Minidisks.....	244
Moving System Minidisks into Data Spaces.....	244
Making Existing Directories Data Space Eligible.....	245
Monitoring Data Space Usage.....	245
Chapter 15. Generating a File Pool and Server.....	247
Step 1: Estimate the Maximum Number of Users (MAXUSERS).....	249
Step 2: Estimate the Maximum Number of Minidisks (MAXDISKS).....	249
Step 3: Determine Initial DASD Allocations.....	250
Step 3A: Determine the Control Minidisk Allocation.....	250
Step 3B: Determine the File Pool Repository & CRR Log Minidisk Allocations.....	251
Step 3C: Determine the Work Minidisk Allocation.....	254
Step 3D: Determine the Catalog Minidisk Allocation.....	254
Step 3E: Determine the User Data Minidisk Allocation.....	255
Step 4: Define a Server Machine.....	256
Step 5: Define an Administration Machine.....	261
Step 6: Log On the New Server Machine.....	262
Step 7: Create a Startup Parameter File.....	263
Step 8: Generate the File Pool.....	267
Step 9: Back Up the Control Data.....	270
Step 10: Start Multiple User Mode Processing.....	271
Step 11: Disconnect from the Server Machine.....	271
Step 12: Log On the Administration Machine.....	271
Step 13: Enroll the Administration Machine in the File Pool.....	271
Step 14: Create a PROFILE EXEC.....	272
Step 15: Consider Enrolling PUBLIC in the File Pool.....	272
Step 16: Enroll the Initial Set of File Pool Users.....	272
Step 17: Set Up the Server for Automatic Starting.....	272
Step 18: Copy the Sample Files to the New File Pool (SFS only).....	273
Step 19: Adjust Cache Size (SFS only).....	275
Chapter 16. Deleting a File Pool.....	277

Chapter 17. Participation in CRR (SFS only).....	279
Overview.....	279
Participation Commands.....	279
Administrator and Operator Intervention.....	279
Step 1: Determine Work That Is Prepared-and-Not-Connected.....	280
Step 2: Identify the Appropriate CRR Recovery Server.....	280
Step 3: Contact the CRR Recovery Server Operator.....	281
Step 4: Determine If Heuristic Action is Necessary.....	282
Step 5: If Action Is Required, Commit or Roll Back.....	282
Step 6: Determine If Any Further Action is Required.....	282
SFS Log Name Table Changes.....	283

## **Part 2. Administering Coordinated Resource Recovery (SFS only)..... 285**

Chapter 18. CRR Overview.....	287
What Is CRR?.....	287
Banking Example.....	287
SFS Example.....	288
What Are the CRR Tasks?.....	288
Administration and Operation.....	288
Application Development.....	288
Product Participation.....	288
Introduction to CRR Terminology.....	289
What Are Distributed Resources and Protected Resources?.....	289
What Are Protected Conversations?.....	289
What Are Distributed Applications?.....	292
CRR Functions.....	294
CRR Recovery Server.....	296
Coordination Function.....	296
Registration and Exits.....	297
Sync Point Processing.....	298
Sync Point Tree.....	299
Allocation and Sync Point Trees.....	302
Last Agent Optimization.....	304
Resynchronization Function.....	305
Logging Function.....	306
CRR Logs.....	306
Log Name Table.....	306
Log Ring.....	307
Product Participation in CRR.....	307
Data Restoration Considerations.....	308
CRR Management in the Distributed Environment.....	308
What Is a Transaction Program?.....	309
What Is an LUWID?.....	310
What About LUWIDs and CMS Work Units?.....	311
CRR Functional Flow.....	311
Chapter 19. CRR Administration.....	315
Generate a CRR Recovery Server.....	317
Designate an Alternate CRR Recovery Server.....	318
Remove Designation of Alternate CRR Recovery Server.....	320
CRR Log Name Table Management.....	321
Increase Size of CRR Log Minidisks.....	323
Free Space in the CRR Log Name Table.....	323
Stop and Restart Automatic Periodic Retry of Resynchronization.....	324
Stop Automatic Periodic Retry of Resynchronization.....	324

Restart Automatic Periodic Retry of Resynchronization.....	324
Bypass and Change Timed Wait Interval.....	325
Bypass Timed Wait Interval.....	325
Change Timed Wait Interval.....	325
Set Up Remote User Machine Administration of CRR Recovery Server.....	326
Managing CRR Performance.....	327
Replace Minidisks in CRR File Pool.....	327
Replace One CRR Log Minidisk.....	328
Replace Both CRR Log Minidisks.....	328
Problem Management.....	330
Step 1: Notification of a Problem.....	332
Step 2: Obtain Token for LUWID at VM1.....	333
Step 3: Determine Status of Resources at VM1.....	333
Step 4: Obtain Token for LUWID at VM3.....	334
Step 5: Determine Status of Resources at VM3.....	335
Step 6: Determine Heuristic Action.....	336
Step 7: Take Heuristic Action.....	336

### **Part 3. Administration Reference..... 337**

Chapter 20. File Pool Server Startup Parameters.....	339
How to Specify Startup Parameters.....	339
Startup Parameter Descriptions.....	342
Chapter 21. File Pool Administration and Server Commands.....	357
Administration and Server Command Types.....	357
Administration Commands.....	357
Server Machine Commands.....	358
Operator Commands.....	359
Using the Online HELP Facility.....	360
AUDIT.....	362
BACKUP.....	366
CRR ERASE LU.....	368
CRR ERASE LUWID.....	370
CRR QUERY LOG.....	371
CRR QUERY LOGTABLE.....	373
CRR QUERY LU.....	376
CRR QUERY LUWID.....	382
CRR RESUME.....	388
CRR RESYNC.....	390
CRR SUSPEND.....	394
DATASPACE.....	396
DEFBACKUP.....	398
DELETE ADMINISTRATOR.....	400
DELETE LOCK.....	403
DELETE PUBLIC.....	406
DELETE USER.....	408
DISABLE.....	412
ENABLE.....	415
ENROLL ADMINISTRATOR.....	417
ENROLL PUBLIC.....	420
ENROLL USER.....	422
ERASE LUNAME.....	427
ETTRACE.....	429
FILEPOOL BACKUP.....	432
FILEPOOL CLEANUP.....	437
FILEPOOL CONTROL BACKUP.....	439

FILEPOOL DISABLE.....	443
FILEPOOL ENABLE.....	447
FILEPOOL FILELOAD.....	451
FILEPOOL FORMAT AUDIT.....	455
FILEPOOL LIST BACKUP.....	459
FILEPOOL LIST MINIDISK.....	467
FILEPOOL MINIDISK.....	470
FILEPOOL RELOAD.....	476
FILEPOOL RENAME.....	484
FILEPOOL RESTORE.....	488
FILEPOOL UNLOAD.....	495
FILESERV BACKUP.....	500
FILESERV CRRLOG.....	502
FILESERV DEFAUDIT.....	504
FILESERV DEFBACKUP.....	506
FILESERV DEFCRRLOG.....	509
FILESERV FIXCENT.....	511
FILESERV GENERATE.....	513
FILESERV LIST.....	521
FILESERV LOG.....	523
FILESERV MINIDISK.....	525
FILESERV MOVEUSER.....	528
FILESERV REGENERATE.....	531
FILESERV REORG.....	535
FILESERV START.....	537
FORCE.....	539
GRANT ADMIN.....	541
ITRACE.....	542
MODIFY USER.....	544
QUERY ACCESSORS (SFS only).....	547
QUERY DATASPACE (SFS only).....	550
QUERY DEFBACKUP.....	553
QUERY DISABLE.....	555
QUERY FILEPOOL AGENT.....	557
QUERY FILEPOOL CATALOG.....	560
QUERY FILEPOOL CONFLICT.....	563
QUERY FILEPOOL COUNTER.....	569
QUERY FILEPOOL CRR (CRR only).....	574
QUERY FILEPOOL DISABLE.....	577
QUERY FILEPOOL LOG.....	583
QUERY FILEPOOL MINIDISK.....	586
QUERY FILEPOOL OVERVIEW.....	589
QUERY FILEPOOL REPORT.....	592
QUERY FILEPOOL STATUS.....	623
QUERY FILEPOOL STORGRP.....	652
QUERY LIMITS.....	655
QUERY LOGTABLE.....	658
QUERY PREPARED.....	660
REVOKE ADMIN.....	663
SET THRESHOLD.....	664
STOP.....	666
<b>Appendix A. Sample Execs for SFS Administration.....</b>	<b>669</b>
SFSTRANS EXEC.....	670
REGRANT EXEC.....	674
TALLY EXEC.....	677
WHO EXEC.....	681

<b>Appendix B. File Pool Server Maximums.....</b>	<b>685</b>
<b>Appendix C. File Pool Server Exit Considerations.....</b>	<b>687</b>
<b>Appendix D. Mapping SFS Authorization Calls to RACROUTE Requests.....</b>	<b>689</b>
Authorization Checking Routine Processing.....	689
SFS Operator Command Authorization Checking Routine.....	689
SFS Administrator Command Authorization Checking Routine.....	692
SFS Object Authorization Routine.....	694
User Data Structure.....	695
RACROUTE Request Formats.....	696
Mapping Tables.....	697
ESM Initialization and Termination.....	697
ESM Program Check.....	698
SFS Operator Commands.....	698
SFS Non-Operator Commands and CSL Routines.....	698
SFS Objects.....	699
<b>Appendix E. Sample NetView CLISTs for Distributed CRR Management.....</b>	<b>707</b>
Process Explanation.....	707
Using the Sample CLIST Files.....	708
File Descriptions.....	708
Operation and Testing.....	709
Potential Problems and Solutions.....	709
Use of Available Automation Functions.....	710
Listings of Sample Files.....	710
AOPMCRRM FPCLST.....	710
CRRMSGR DNCLST.....	711
CRRMSGR EXEC.....	713
CRRM001 FPPANEL.....	714
CRRM002 FPPANEL.....	714
CRRM003 FPPANEL.....	715
CRRM004 FPPANEL.....	715
CRROPF APPEND.....	716
MATDNCRR APPEND.....	716
MATFPCRR APPEND.....	717
<b>Notices.....</b>	<b>719</b>
Programming Interface Information.....	720
Trademarks.....	720
Terms and Conditions for Product Documentation.....	721
IBM Online Privacy Statement.....	721
<b>Bibliography.....</b>	<b>723</b>
Where to Get z/VM Information.....	723
z/VM Base Library.....	723
z/VM Facilities and Features.....	725
Prerequisite Products.....	726
<b>Index.....</b>	<b>727</b>





---

# Figures

- 1. File Pool Structure..... 7
- 2. File Pool Structure..... 8
- 3. A File Pool Server Machine Operator Console..... 19
- 4. Multiple User Mode Processing..... 20
- 5. Disconnected Server Operation..... 21
- 6. Startup Parameters for VMSEVS (for System Data)..... 41
- 7. Startup Parameters for VMSEVP (for Product Service Data)..... 42
- 8. Startup Parameters for VMSEVU (for User Data)..... 42
- 9. Startup Parameters for VMSEVR (for CRR Recovery Server)..... 42
- 10. A File Space..... 81
- 11. A File Space For Department 16..... 81
- 12. A File Space For Department 16 with Subdirectory Common..... 82
- 13. Example with SALES and INVENTORY..... 83
- 14. Example Department 16 with the Inventory and Sales File Spaces..... 84
- 15. Example /../VMFBFS:VMSYS:ROOT..... 85
- 16. Example - the result of the Ed's commands..... 86
- 17. Example - the result of the Brad's commands..... 88
- 18. Example - External Link..... 90
- 19. File Pool Control Data and User Data..... 102
- 20. IBM-supplied DMSESM PROFILE..... 164
- 21. File Pool Server Interface to an ESM..... 171
- 22. DMSJBTP TEMPLATE file..... 176
- 23. DMSJBATP TEMPLATE File..... 178

24. DMSJBPTP TEMPLATE File.....	183
25. Example MDISK Statements for Additional File Pool Minidisks.....	198
26. Example Control Statements for FILESERV MINIDISK or FILEPOOL MINIDISK Processing.....	202
27. Format of the Control Statements for FILESERV MINIDISK or FILEPOOL MINIDISK Processing.....	203
28. DMS5XXTP TEMPLATE File.....	210
29. Initialization Accounting Record Format.....	225
30. Operator/Checkpoint Accounting Record Format.....	226
31. System Accounting Record Format.....	227
32. Termination Accounting Record Format.....	228
33. User Accounting Record Format.....	228
34. DMS2AB TEMPLATE File.....	230
35. Example z/VM System Directory Control Statements for Server FPSERV1.....	257
36. Protected Resources in Same Processor.....	290
37. Protected Resources in a TSAF or CS Collection.....	291
38. Protected Resources in SNA Network.....	292
39. Protected Resources Using Protected Conversations in a SNA Network.....	294
40. Coordinated Resource Recovery (CRR) Functions.....	295
41. Registration and Exits.....	298
42. Sync Point Tree (Detailed Format).....	301
43. Sync Point Tree (General Format).....	302
44. Allocation and Sync Point Trees.....	303
45. Last Agent Optimization.....	305
46. Distributed Management of CRR.....	309
47. Simplified Structure for a Distributed Application Program (TPN A).....	310
48. Simplified Structure for a Distributed Application Program (TPN B).....	310

49. CRR Problem Management Scenario.....	331
50. Example Content of SERVER DMSPARMS File.....	340
51. Parental Direction Flow.....	391
52. Child's Response Flow.....	392
53. Contents of the DMSIBM POOLDEF File.....	514



---

# Tables

1. Examples of Syntax Diagram Conventions.....	xxi
2. Typical Activities.....	4
3. As-Needed Activities.....	4
4. Optional Activities.....	5
5. Dedicated Maintenance Mode Commands.....	19
6. File Pool Repository and CRR Operator Commands.....	22
7. File Pool Administration Commands.....	26
8. Differences between User Data Backup and Control Data Backup.....	103
9. Differences between User Data Restore and Control Data Restore.....	103
10. Forms of Backups Needed to Replace File Pool Minidisks.....	129
11. Authorization or Permission Depending on Connection Type.....	138
12. File Pool Request Codes (SFS).....	151
13. File Pool Request Codes (BFS).....	156
14. Special File Pool Request Codes for SFS Operator Commands.....	168
15. SFS ESM Exits.....	171
16. BRM requests.....	183
17. CRED - DMSPERM parm 10, length is 64 bytes.....	185
18. FSP- DMSPERM parm 11, length is 208 bytes.....	186
19. CLIENT Structure - DMSPERM parm 12, length is 88 bytes.....	188
20. CSL Routines That Allow ESM User Data.....	190
21. General Data Buffer Out Flags Settings.....	211
22. DMSJNEPL Macro Mapping.....	239
23. Estimates of Control Minidisk Sizes.....	250

24. Uses of MINIOPT NOMDC and MINIOPT NOMDC NOCACHE.....	261
25. CRR Administration and Operation Activities.....	315
26. CRR Dedicated Maintenance Mode Command.....	316
27. CRR Operator Command.....	316
28. CRR Administration Commands.....	316
29. CRR File Pool Minidisk Replacement Tasks.....	327
30. File Pool Administration Commands.....	357
31. File Pool FILESERV Commands.....	359
32. File Pool Operator Commands.....	359
33. QUERY FILEPOOL Command Information Cross Reference.....	593
34. QUERY FILEPOOL Command Information Cross Reference.....	624
35. File Pool Server Maximums.....	685
36. Rebuilt SFS Operator Commands.....	689
37. Rebuilt SFS Command and CSL Routine Strings.....	692
38. Format of the ESM User Data Structure.....	695
39. Mapping for ESM Initialization.....	697
40. Mapping for ESM Termination.....	697
41. Mapping for ESM Program Check.....	698
42. RACROUTE Mapping for SFS Operator Commands.....	698
43. RACROUTE Mapping for SFS Non-Operator Commands and CSL Routines.....	698
44. Mapping for COPYFILE Command.....	699
45. Mapping for CREATE ALIAS Command.....	699
46. Mapping for CREATE DIRECTORY Command.....	700
47. Mapping for DMSCROB CSL Routine.....	700
48. Mapping for CREATE LOCK Command.....	701

49. Mapping for DISCARD and ERASE Commands.....	701
50. Mapping for DISCARD and ERASE Commands (Commit or Rollback).....	702
51. Mapping for DMSOPEN CSL Routine.....	702
52. Mapping for 'Do You Know?' Query.....	703
53. Mapping for Generic Profile Query.....	703
54. Mapping for Miscellaneous Authority Check.....	704
55. Mapping for RELOCATE (File, Alias, External Object, or Directory) Command.....	704
56. Mapping for RELOCATE (Commit or Rollback).....	705
57. Mapping for RENAME(File, Alias, External Object, or Directory) Command.....	705
58. Mapping for RENAME (Commit or Rollback).....	705
59. Mapping for 'What Authority?' Query.....	706





# About This Document

---

This document provides the information you will need to plan, administer, and operate IBM® z/VM® CMS file pools. It provides information about using file pools as repositories for CMS shared file system (SFS) and OpenExtensions™ byte file system (BFS) data. It also provides information about using file pool server machines for coordinated resource recovery (CRR) and BFS FIFO functions. It includes a reference section that contains descriptions of file pool startup parameters and file pool administration and server commands.

## Intended Audience

---

This information is for CMS system administrators or anyone managing the tasks of planning, administering, and operating CMS file pool servers. A system administrator's role may vary depending on the size of an installation and the organization's requirements. This information provides guidelines that will apply to most environments.

You should be familiar with DASD management and z/VM user directory control statements on your system. You should also be familiar with the shared file system and the byte file system as described in *z/VM: CMS User's Guide* and *z/VM: OpenExtensions User's Guide*.

## Syntax, Message, and Response Conventions

---

The following topics provide information on the conventions used in syntax diagrams and in examples of messages and responses.

### How to Read Syntax Diagrams

Special diagrams (often called *railroad tracks*) are used to show the syntax of external interfaces.

To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

- The **▶▶—** symbol indicates the beginning of the syntax diagram.
- The **—▶** symbol, at the end of a line, indicates that the syntax diagram is continued on the next line.
- The **▶—** symbol, at the beginning of a line, indicates that the syntax diagram is continued from the previous line.
- The **—▶◀** symbol indicates the end of the syntax diagram.

Within the syntax diagram, items on the line are required, items below the line are optional, and items above the line are defaults. See the examples in [Table 1 on page xxi](#).

<i>Table 1. Examples of Syntax Diagram Conventions</i>	
<b>Syntax Diagram Convention</b>	<b>Example</b>
<b>Keywords and Constants</b> A keyword or constant appears in uppercase letters. In this example, you must specify the item KEYWORD as shown.  In most cases, you can specify a keyword or constant in uppercase letters, lowercase letters, or any combination. However, some applications may have additional conventions for using all-uppercase or all-lowercase.	<b>▶▶ KEYWORD ▶▶</b>

Table 1. Examples of Syntax Diagram Conventions (continued)

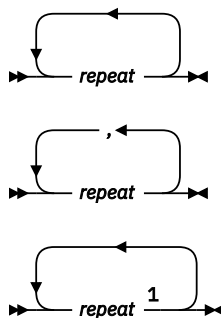
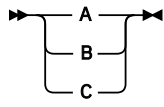
Syntax Diagram Convention	Example
<p><b>Abbreviations</b></p> <p>Uppercase letters denote the shortest acceptable abbreviation of an item, and lowercase letters denote the part that can be omitted. If an item appears entirely in uppercase letters, it cannot be abbreviated.</p> <p>In this example, you can specify KEYWO, KEYWOR, or KEYWORD.</p>	<p>▶▶ KEYWOrd ◀◀</p>
<p><b>Symbols</b></p> <p>You must specify these symbols exactly as they appear in the syntax diagram.</p>	<p>* Asterisk</p> <p>:</p> <p>Comma</p> <p>= Equal Sign</p> <p>- Hyphen</p> <p>() Parentheses</p> <p>.</p> <p>Period</p>
<p><b>Variables</b></p> <p>A variable appears in highlighted lowercase, usually italics.</p> <p>In this example, <i>var_name</i> represents a variable that you must specify following KEYWORD.</p>	<p>▶▶ KEYWOrd — <i>var_name</i> ◀◀</p>
<p><b>Repetitions</b></p> <p>An arrow returning to the left means that the item can be repeated.</p> <p>A character within the arrow means that you must separate each repetition of the item with that character.</p> <p>A number (1) by the arrow references a syntax note at the bottom of the diagram. The syntax note tells you how many times the item can be repeated.</p> <p>Syntax notes may also be used to explain other special aspects of the syntax.</p>	 <p>Notes:</p> <p><sup>1</sup> Specify <i>repeat</i> up to 5 times.</p>
<p><b>Required Item or Choice</b></p> <p>When an item is on the line, it is required. In this example, you must specify A.</p> <p>When two or more items are in a stack and one of them is on the line, you must specify one item. In this example, you must choose A, B, or C.</p>	<p>▶▶ A ◀◀</p> 

Table 1. Examples of Syntax Diagram Conventions (continued)

Syntax Diagram Convention	Example
<p><b>Optional Item or Choice</b></p> <p>When an item is below the line, it is optional. In this example, you can choose A or nothing at all.</p> <p>When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.</p>	
<p><b>Defaults</b></p> <p>When an item is above the line, it is the default. The system will use the default unless you override it. You can override the default by specifying an option from the stack below the line.</p> <p>In this example, A is the default. You can override A by choosing B or C.</p>	
<p><b>Repeatable Choice</b></p> <p>A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item.</p> <p>In this example, you can choose any combination of A, B, or C.</p>	
<p><b>Syntax Fragment</b></p> <p>Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name.</p> <p>In this example, the fragment is named "A Fragment."</p>	

## Examples of Messages and Responses

Although most examples of messages and responses are shown exactly as they would appear, some content might depend on the specific situation. The following notation is used to show variable, optional, or alternative content:

**xxx**

Highlighted text (usually italics) indicates a variable that represents the data that will be displayed.

[ ]

Brackets enclose optional text that might be displayed.

{ }

Braces enclose alternative versions of text, one of which will be displayed.

|

The vertical bar separates items within brackets or braces.

...

The ellipsis indicates that the preceding item might be repeated. A vertical ellipsis indicates that the preceding line, or a variation of that line, might be repeated.

## Where to Find More Information

---

For more information about CMS and other z/VM topics, see [“Bibliography” on page 723](#).

### Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

## How to provide feedback to IBM

---

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



# Summary of Changes for z/VM: CMS File Pool Planning, Administration, and Operation

---

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

## SC24-6261-02, z/VM 7.2 (December 2023)

---

This edition includes terminology, maintenance, and editorial changes.

## SC24-6261-02, z/VM 7.2 (December 2021)

---

This edition includes terminology, maintenance, and editorial changes.

## SC24-6261-02, z/VM 7.2 (September 2021)

---

This edition includes terminology, maintenance, and editorial changes.

## SC24-6261-02, z/VM 7.2 (July 2021)

---

This edition includes terminology, maintenance, and editorial changes.

## SC24-6261-02, z/VM 7.2 (March 2021)

---

This edition includes changes to support product changes provided or announced after the general availability of z/VM 7.2.

### **[VM66201, VM66425] z/Architecture Extended Configuration (z/XC) support**

With the PTFs for APARs VM66201 (CP) and VM66425 (CMS), z/Architecture® Extended Configuration (z/XC) support is provided. CMS applications that run in z/Architecture can use multiple address spaces. A z/XC guest can use VM data spaces with z/Architecture in the same way that an ESA/XC guest can use VM data spaces with Enterprise Systems Architecture. z/Architecture CMS (z/CMS) can use VM data spaces to access Shared File System (SFS) Directory Control (DIRCONTROL) directories. Programs can use z/Architecture instructions and registers (within the limits of z/CMS support) and can use VM data spaces in the same CMS session.

Information in the following topics is updated:

- [“Data Space Exploitation \(SFS Only\)” on page 52](#)
- [Chapter 14, “Using Data Spaces \(SFS Repository Servers Only\),” on page 241](#)
- [“Configuring Your Server for Data Spaces” on page 242](#)
- [Generating a File Pool and Server, “Step 4: Define a Server Machine” on page 256](#)

## SC24-6261-01, z/VM 7.2 (September 2020)

---

This edition supports the general availability of z/VM 7.2.

## SC24-6261-00, z/VM 7.1 (September 2018)

---

This edition supports the general availability of z/VM 7.1.





---

# Part 1. Administering Repository File Pools

This part describes how to administer one or more file pools. Read [Chapter 1, “File Pool Administration Overview,”](#) on page 3 first because it defines terms and concepts used throughout this part. The remaining chapters of this part do not need to be read sequentially. These chapters are:

- [Chapter 2, “Installation Planning for File Pools,”](#) on page 31
- [Chapter 3, “Post-installation Activities,”](#) on page 37
- [Chapter 4, “Migration Considerations,”](#) on page 51
- [Chapter 5, “Operation,”](#) on page 55
- [Chapter 6, “Managing Users and File Spaces,”](#) on page 71
- [Chapter 7, “Recovery Procedures,”](#) on page 101
- [Chapter 8, “Security,”](#) on page 137
- [Chapter 9, “Managing Storage,”](#) on page 195
- [Chapter 10, “Reorganizing the File Pool Repository Catalogs,”](#) on page 213
- [Chapter 11, “Regenerating a Repository File Pool,”](#) on page 217
- [Chapter 12, “Accounting,”](#) on page 223
- [Chapter 13, “Setting Up a File Pool for Remote Use,”](#) on page 233
- [Chapter 14, “Using Data Spaces \(SFS Repository Servers Only\),”](#) on page 241
- [Chapter 15, “Generating a File Pool and Server,”](#) on page 247
- [Chapter 16, “Deleting a File Pool,”](#) on page 277
- [Chapter 17, “Participation in CRR \(SFS only\),”](#) on page 279



# Chapter 1. File Pool Administration Overview

This chapter defines the terms and concepts used to administer one or more file pool servers. It also provides an overview that a system administrator needs to know to administer one or more file pools. File pool servers provide repository services for Shared File Systems (SFS) and OpenExtensions Byte File System (BFS). Additional file pool services are provided for Coordinated Resource Recovery (CRR). An additional file pool service for optimizing the performance of BFS FIFO objects can also involve the administration of some specialized FIFO file pool services.

## File Pool Servers and Possible Combinations

SFS, BFS, CRR, and FIFO services can be performed by one or more file pool servers in your system. The following rules explain the combinations allowed:

- Any file pool server has the potential for performing any of these services. That potential is realized when you:
  - add users, file spaces (or both) to the file pool
  - create CRR or FIFO (or both) selections with the startup parameters
- Administrators enable SFS activity in a file pool by enrolling users with SFS file spaces.
- Administrators enable BFS activity in a file pool by creating (enrolling) Byte File Systems (BFSs).
- SFS file spaces and BFS file spaces can coexist in the same file pool.
- Administrators select which file pool server(s) are to perform CRR and FIFO services based on startup parameters.
- Servers may be dedicated to CRR, FIFO or both services or these services also may be performed in connection with ordinary SFS or BFS repository services.
- There is at most one file pool in a system designated as the CRR Recovery Server. It may perform other file pool services as well (although this is not recommended.)
- Each file pool server that has BFS file spaces either defaults to doing its own FIFO services, or, through startup parameters, designates another file pool server to do its FIFO services. (See [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.) A BFS participating file pool can select only one other file pool to do its FIFO services, and that file pool must be on the local system. However, any file pool can do FIFO file pool services for any number of other local file pools.

When a server is dedicated to a particular service, only some specific administrative functions apply to that server. When a topic applies to particular aspects of file pool services, it is identified as such so you can bypass sections or commands that do not pertain to your current activities. Following are some of the filters used to help you to bypass unnecessary reading:

- CRR only - the function or section applies only to CRR services.
- FIFO only - the function or section applies only to FIFO service.
- Repository only - the function or section does not apply to CRR or FIFO dedicated file pool servers, but does apply to the basic SFS and Byte File System (BFS) repository services.
- SFS only - the command, function, topic, or section applies only to SFS and SFS file spaces.
- BFS only - the command, function, topic, or section applies only to BFS file spaces.

When a file pool is dedicated to CRR or FIFO work, by definition it is not directly involved with repository functions. On the other hand, when a file pool server performs some CRR or FIFO services along with some (SFS or BFS) repository services (possible but not generally recommended) fewer such filters apply.

When a filter is omitted or when the words *file pool server*, or *file pool* or *server* are used, you should understand the function, option, command, topic or section pertains to any or all of the file pool services.

A guide to the job of file pool administration is contained in [Table 2 on page 4](#), [Table 3 on page 4](#), and [Table 4 on page 5](#). These tables identify the pages on which instructions for various often-used procedures begin. Related procedures are grouped into chapters bearing titles such as "Operation", "Recovery", "Security", and "Managing Users". Unrelated or lengthy procedures, such as file pool generation, are in separate chapters.

The tables divide the job of file pool administration into three categories. [Table 2 on page 4](#) shows the typical day-to-day tasks a person administering a file pool does. That is, he or she starts something called a *file pool server machine*, monitors it, shuts it off periodically, and makes an occasional backup of the data in a *file pool*. A file pool administrator also enrolls users in the file pool and, as needed, changes the amount of space a user is allowed to consume.

[Table 3 on page 4](#) shows activities you may or may not need to do, depending on events that are out of your control. For instance, you may need to add DASD space to the file pool if your users consume all the space in it. Or, if there is a DASD error, you will need to restore data that was lost.

[Table 4 on page 5](#) shows other tasks you might do that are completely optional. You can decide at some later time whether you want to do them.

If you are the person responsible for administering the file pools supplied with z/VM, which are named VMSYS, VMPSFS, VMSYSU, and VMSYSR after reading this section you must see [Chapter 3, "Post-installation Activities," on page 37](#).

Part 3 of this document contains a command reference for experienced file pool administrators who no longer find it necessary to follow the step-by-step procedures in this part.

*Table 2. Typical Activities*

<b>Task</b>	<b>Location</b>
Start a file pool server machine	<a href="#">Starting Multiple User Mode Processing</a>
Monitor file pool server processing	<a href="#">Monitoring Server Operation</a>
Stop a file pool server machine	<a href="#">Stopping Server Processing</a>
Back up (make a copy) of the data in a file pool	<a href="#">"Backing Up a File Pool - Overview" on page 101</a>
Enroll users or add file spaces and give them space in a file pool	<a href="#">"Enrolling SFS Users and Creating BFS File Spaces" on page 71</a>
Change the amount of space a user or file space has	<a href="#">"Changing the Amount of Space Assigned to a File Space" on page 91</a>
Delete users or file spaces in a file pool	<a href="#">"Deleting Enrolled Users and File Spaces" on page 93</a>

*Table 3. As-Needed Activities*

<b>Task</b>	<b>Location</b>
Add disk space to the file pool	<a href="#">Rearranging Existing Storage Groups</a>
Move the file pool	<a href="#">Moving a File Pool</a>
Move a user or file space to a different storage group	<a href="#">"Moving an SFS User in Dedicated Maintenance Mode" on page 96</a>
Move a user or file space to a different file pool	<a href="#">"Overview of Moving Users and File Spaces" on page 95</a>

Table 3. As-Needed Activities (continued)

Task	Location
Restore file pool data that was lost	<a href="#">“Restoring Control Data” on page 110</a>
Change the size of the logs	<a href="#">“Replacing Both File Pool Log Minidisks” on page 133</a>
Reorganize the file pool catalogs	<a href="#">Reorganizing the File Pool Repository Catalogs</a>
Regenerate the file pool	<a href="#">Regenerating a Repository File Pool</a>
Trace the execution of the file pool server machine	<a href="#">Tracing facilities</a>

Table 4. Optional Activities

Task	Location
Enroll another administrator	<a href="#">Enrolling an Administrator</a>
Delete an administrator	<a href="#">Deleting an Administrator</a>
Delete everyone enrolled by ENROLL PUBLIC (SFS only)	<a href="#">“Deleting Everyone Enrolled by ENROLL PUBLIC” on page 94</a>
Audit access to the file pool	<a href="#">Auditing Security</a>
Generate accounting information	<a href="#">Starting the Accounting Facility</a>
Improve performance by exploiting data spaces (SFS only)	<a href="#">Using Data Spaces (SFS Repository Servers Only)</a>
Make a file pool available to other processors	<a href="#">“Making Your File Pool Available to Other Processors” on page 233</a>
Define another file pool and file pool server machine	<a href="#">Generating a File Pool and Server</a>
Monitor performance	<a href="#">See <i>z/VM: Performance</i></a>
Delete a file pool	<a href="#">Deleting a File Pool</a>
Modify DMSSFSEX file pool server exits	<a href="#">Activating the Storage Use Exits (SFS and BFS)</a>

## Types of File Pool Data Repositories

With CMS file pools, you have the capability of providing data repository services for two types of data:

- Shared File System (SFS) data that is stored and accessed as records.
- Byte File System (BFS) data that is stored and accessed as streams of data bytes, without reference to record boundaries

In addition, there are many other distinguishing characteristics between these two types of data. They involve the characteristics of data sharing, update and change consistency, recoverability, object types, object naming, and access authorization as a few examples.

Despite these differences, most file pool administration services are common to both of these types of data.

## What is a File Pool?

The distinguishing characteristics of the two data types are associated with the type of file space created to hold the data. That is, SFS data is stored in SFS file spaces and BFS data is stored in BFS file space.

**Note:** BFS file spaces are also known as Byte File Systems (BFSs).

Although you can mix SFS and BFS file spaces in a file pool, you cannot mix SFS and BFS data types in the same file space.

## What Is a File Pool?

---

SFS and BFS data is stored in one or more *file pools*. Each file pool is a collection of minidisks owned by a particular file pool virtual machine known as a file pool server. The minidisks are used for storing file pool repository data, along with control data (for example, catalogs, logs, and parameter files) necessary for keeping the data definitions and recovery information.

In addition to managing repository data, some file pools may be designated for off-loading specific services that do not necessarily require reference to file pool repository data. The reasons for off-loading work to these specialized file pools is to improve performance or to provide a more centralized point of control and administration. Such off-load capabilities is currently allowed for Coordinated Resource Recovery (CRR) services and BFS FIFO services. When a file pool server performs one of these specialized services, it may be referred to as a server of that purpose (CRR server or FIFO server). It may be dedicated to that service or it may work in conjunction with other services. For example, a file pool server may be a CRR server and a repository server, or it may be a CRR server and a FIFO server. See [“File Pool Servers and Possible Combinations”](#) on page 3 for definitions of the permitted combinations.

Your system may include servers other than file pool servers. Examples include inter-processor communications servers and relational data base servers.

Before a file pool can hold users' files, you must prepare its minidisks using a process known as *file pool generation*. During file pool generation, the minidisks are formatted and internal information is placed on them. The only limit to the number of file pools you can generate is the capacity of your system.

All file pools have a name. This *file pool identifier* or *file pool ID*, for short, is the name by which users know the file pool. When users are allowed to use a file pool, they must be told the ID of the file pool. If, for instance, user Bob is enrolled in a file pool named FINANCE, he needs to specify FINANCE on some CMS commands that manipulate his files. Without knowing the file pool ID, he is unable to use the file pool.

The file pool ID is established when the file pool is generated. You can change a file pool's name, but it is not practical to change it often. Every time you change the file pool ID, you must tell all the file pool users what the new name is. They, in turn, may have to update execs or programs to refer to the new file pool ID. The following figure shows an example of the minidisks that constitute a file pool:

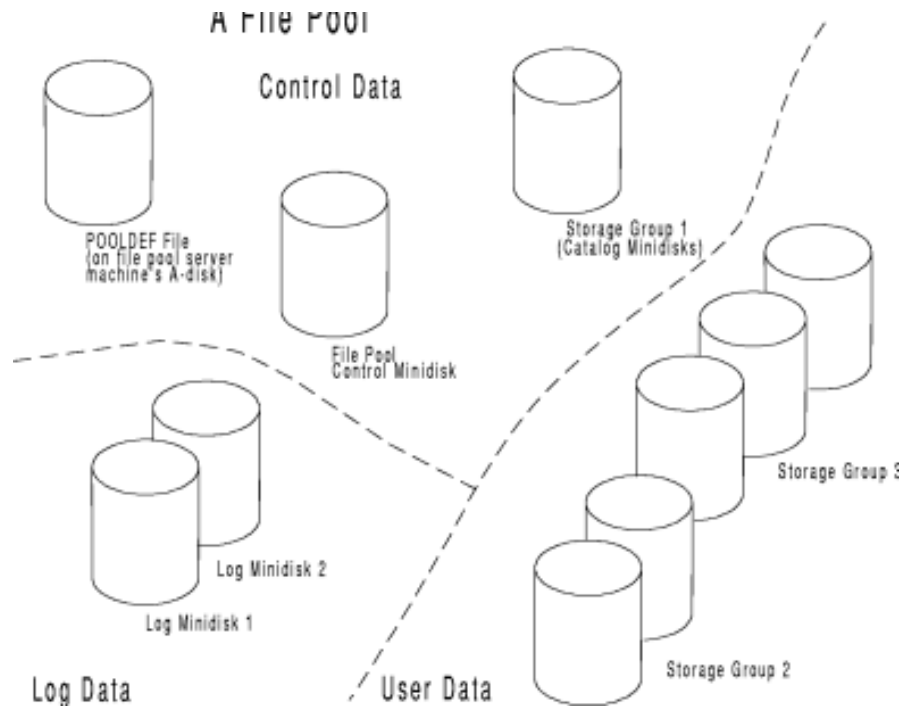


Figure 1. File Pool Structure

## How Users Access Their Files

When users read or write files or otherwise operate on objects in a file pool, these operations are not done directly to a minidisk. Instead, CMS, which is running in the user machine, asks the file pool server machine to do the operations. See [Figure 2 on page 8](#) for an illustration of a file pool structure throughout the following discussion.

## What is a File Pool?

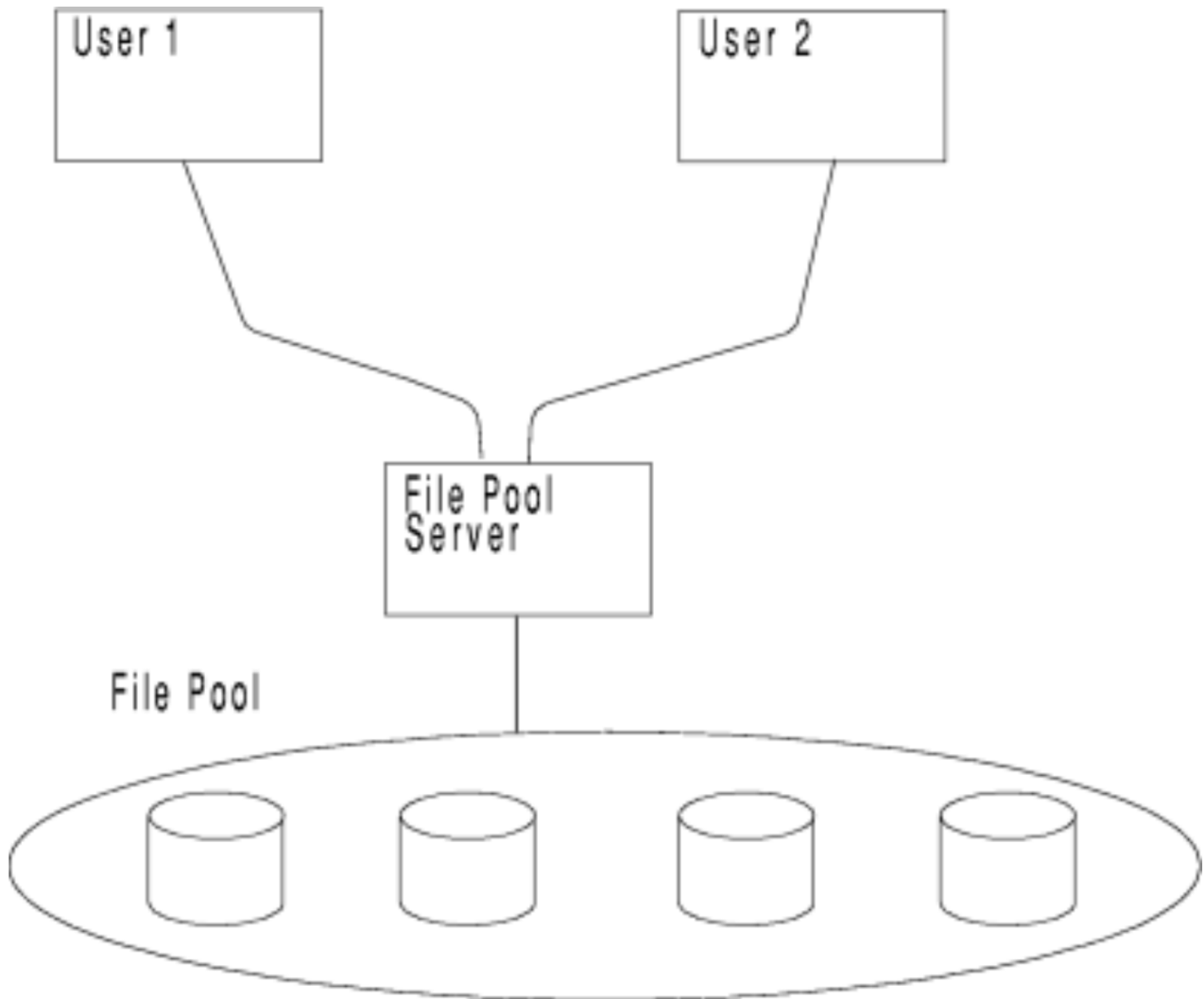


Figure 2. File Pool Structure

The file pool server receives requests from user machines and acts accordingly. When a user writes to a file, CMS in the user machine sends the data to be written to the server. The file pool server receives the request, finds the file in its collection of minidisks, and does the write. When a user reads from a file, CMS (in the user machine) again sends a request to the file pool server machine. The file pool server machine looks for the file in its collection of minidisks and, upon finding it, returns the file to the user machine.

When a user or associated application first establishes communications with a file pool by entering commands that use files or other objects within it, the user or application is said to *connect* to the file pool. This connection is not directly known to the user. There are some restrictions for controlling access (connection) to a file pool server. These restrictions allow you to prevent unnecessary consumption of file pool resources from unneeded file pool server accesses by unauthorized users. Even when no repository objects are involved, certain file pool server resources can be consumed by just managing connection activity and retaining connection resources.

The connection rules are somewhat different depending on the type of file pool operations required to complete a user request.

Requests for SFS type file pool services must meet at least one of the following requirements:

- The user is enrolled in the file pool. The user may be enrolled by name through the ENROLL USER command, or through ENROLL PUBLIC, which lets everyone on the z/VM system connect to the file pool.
- The user is a file pool administrator.
- The user has been assigned a non-default POSIX user ID (UID).



If the user is not enrolled in the file pool and is not a file pool administrator, only the following types of requests are allowed:

- QUERY ENROLL USER
- QUERY ENROLL ADMINISTRATOR
- QUERY FILEPOOL DISABLE
- QUERY FILEPOOL DISABLE FILESPACE
- QUERY FILEPOOL CONFLICT
- QUERY LIMITS

All other requests are denied.

The ENROLL commands can be issued only by a file pool administrator. Even though an SFS user can implicitly connect (establish communications) with a file pool, that user may not necessarily be able to access or create files within it. The user must either be given space in the file pool to create new files, or must have been authorized by others to use their files.

Requests for BFS type file pool services must meet at least one of the following requirements:

- The user has been assigned a non-default POSIX user ID (UID).
- The user has been designated as a BFS superuser.
- The user is a file pool administrator.
- The file pool administrator has established public connect capability for the file pool through the ENROLL PUBLIC command. In this case, the enrollee virtual machine also needs POSIX permissions to make effective use of the connect capability, or the user is established as a user through the ENROLL USER command.

**Note:** There are some special file pool operations defined to be SFS operations, but which also allow for operating upon BFS objects or allow for administering BFS file spaces. These are mostly operations that require the user to be an administrator for the associated file pool. These operations are considered to be SFS operations from a standpoint of file pool server connection restrictions. Because a file pool server administrator can connect either for SFS or BFS operations, this is generally not a problem. A BFS superuser has access permission for all BFS objects, but superuser status does not confer authority for SFS objects, nor does it confer any capabilities to connect to use SFS operations.

A particular user may be both a superuser and a file pool administrator, extending the capabilities and scope appropriately for the combination.

## Data Repository Services

Many of the services of a file pool are associated with maintaining the data repository for the file pool and not with the particular file system that uses the repository. These include:

- most aspects of installation and startup
- maintenance of recovery logs, backup, restore, and data archiving
- management of the storage devices
- managing communications with user virtual machines
- dividing file pool repository storage units called storage groups and managing DASD assignments
- supporting file pool virtual machine operator commands
- monitoring and managing security and performance
- enrolling users and creating file spaces
- controlling the amount of space supported by a file space
- defining file space type (at creation)

However, many of the repository services are tailored to the expected characteristics of the particular file system. These include:

## What is a File Pool?

- data sharing rules
- update/change consistency rules
- object types supported
- object relationships
- naming schemes
- object access authority or permission schemes

A file system is distinguished both by the functions that it supports and the type of the file spaces associated with its objects. In order to create files in the file pool repository, you must have addressability and authority to use at least one file space and that file space must have been created with a file system type that matches the file you create. The file pool administrator creates file spaces and establishes permissions for their use by CMS users. File spaces are a component of the hierarchical addressing scheme, as well as providing basis for controlling the consumption of storage space in the file pool for users or groups of users.

The type (SFS or BFS) established for a file space by the file pool administrator when it is created determines the file system and set of commands and functions that operate on it.

## Shared File System (SFS) Repository Services

The Shared File System (or SFS) is a CMS file system that has the following characteristics:

- Files have record orientation, but data is managed by the file pool in units of 4K bytes (data blocks).
- Files are associated with hierarchical directories
- For each file space there is a single top directory which has a name that corresponds to the name of that file space. This name is also commonly associated with a user virtual machine that is thereby authorized to connect to the file pool and use that file space.
- Files consume storage blocks assigned to a file space in proportion to the size of the file. Space (a limit of so many storage blocks) is assigned to a file space by the file pool administrator. When this space is depleted by committing logical units of work in the file pool, no more files can be created in the file space unless space is freed or more space is assigned. However, space (blocks) can be used in a file space temporarily (before commit) up to the limit of the space in the storage group of which the file space is a part.
- Files (and directories) are shared with other users when the owner or the file pool administrator grants authority to particular users (or PUBLIC) by name - with READ or WRITE capabilities.
- Both local and remote (system) users can be enrolled to use the file system.
- Logical units of work allow for multiple file system operations to be dynamically grouped for purposes of controlling data commit points. Through this facility, data changes can be grouped in the manner of a transaction; then either committed (to permanent storage) or rolled back (undone) as a unit.
- Coordinated Resource Recovery permits coordinated commits (or rollback) of multiple logical units of work in the same or in multiple systems in a network.
- DFSMS/VM provides storage management functions. It erases expired files and migrates low activity files to auxiliary storage which enables better utilization of high performance DASD. See [z/VM: DFSMS/VM Planning Guide](#) for more information.
- Object naming accommodates both CMS schemes (file name, file type, file mode) and an independent, fully-qualified name based the file pool name, the file space name, the hierarchy of directories down to the object itself. File names have an 8 byte limitation, but directory names are limited to 16 bytes.
- In addition to files and directories, some special object types are allowed. Examples are **aliases** which make a file appear to be in a directory where the alias is defined, while the file actually resides in another directory.
- Directory Control Directories allow for special performance enhancements, as well as supporting data update consistency similar to CMS minidisks. These are intended for read-only or read-mostly directories. You can also assign these directories to *data spaces*. A data space is an area of virtual storage that can be used by several virtual machines. When directories reside in data spaces, local

users avoid the overhead of communicating with a file pool server machine. Instead, CMS uses the data space directly, as though it were part of the user's virtual machine. This improves performance for local users. For more information, see [Chapter 14, "Using Data Spaces \(SFS Repository Servers Only\),"](#) on page 241.

- A file can be concurrently opened for read by more than one SFS user. Concurrently with these readers, or independently of them, there can be a single open for update (write) of the file. SFS rejects attempts for concurrent writes to the same file.
- Normal data consistency rules control when you see data repository changes made by other users. In SFS, you normally see such changes only after the logical unit of work has been "committed", so you see changes at fairly high level of consistency. Additionally, an SFS user that opens a file for read access sees a frozen version of the file as it exists (at the level of commit) when the file is opened. This version is kept for that instance until the open file is closed. This open-to-close consistency is realized for each read-level open.

## Byte File System (BFS) Repository Services

The Byte File System (or BFS) is a CMS file system that has the following characteristics:

- Files contain streams of bytes. Nevertheless, data is managed by the file pool in units of 4K bytes (data blocks).
- Files are associated with hierarchical directories
- File spaces (file systems) or portions thereof can be mounted to extend the current path address to objects for a user.
- For each file space there is a single top (root) directory which has a name that corresponds to the name of that file space. This name is **not** necessarily associated with a user virtual machine and has no association with authority to connect to the file pool.
- Files consume storage blocks assigned to a file space in proportion to the size of the file. Space (a limit of so many storage blocks) is assigned to a file space by the file pool administrator. When this space is depleted, no more files can be created in the file space unless space is freed or more space is assigned. For BFS file spaces, there is no capability for consumption of space beyond the file space capacity. That is, you cannot consume blocks in excess of the file space limit temporarily before commit (as you can in SFS).
- Objects are shared with other users based on permission bits and owner UID (user ID) and GID (group ID) that are associated with the object. These are compared to the UID and GID of the file accessor. The permission bits allow for read, write, or execute at the individual user level, group level, or world (public) level.
- File system users can be local or remote (accessed with a network).
- Although atomicity of file system requests is enforced, multiple SFS requests are not considered grouped in logical units of work for purposes of controlling commit points or rolling back (undoing) requests to the file system. The guarantees for atomicity means user or application functions for BFS either complete or fail as a unit. This atomicity is accomplished through a combination of serializations in the CMS user machines where a request originates and the use of logical units of work in the file pool server, where applicable. The file pool server treats each BFS request to the repository server as a logical unit of work. More than one server request may be required to satisfy a single (atomic) operation against a BFS file. BFS requests are not associated with CMS work units.
- Coordinated Resource Recovery does not apply.
- DFSMS/VM provides storage management functions. It erases expired files and migrates low activity files to auxiliary storage which enables better utilization of high performance DASD. See [z/VM: DFSMS/VM Planning Guide](#) for more information.
- Object naming is through POSIX standard path naming or fully qualified names, based on a VMBFS identifier (constant that indicates BFS type file space), the file pool name, the file space name, and the hierarchy of directories down to the object itself. Object names have a length limitation of 256 bytes and path names are limited to 1024 bytes.

## What is a File Pool?

- In addition to files and directories, various special object types are allowed according to the POSIX standard. Examples are FIFOs, hard links, symbolic links, and external links. These allow for inter-process communications, extensions to the scope of addressing, and even extensions to the types of objects.
- BFS allows concurrent reads and writes and does not have open-to-close consistency for file changes. However, concurrent writes must wait for completion of any read or write operations that have already started. That is, the concurrency respects atomicity rules previously described.
- Normal data consistency rules control when you see data repository changes made by other users. In BFS you see changes as soon as an atomic request completes. That is, you do not see a change in progress, but you see the change as soon as it completes. There is no grouping of change requests (file updates) into transactions or logical units of work.
- You can lock byte ranges in files to synchronize access with other users. You can lock against reads and writes to the files, or just against writes. This locking is voluntary. That is, participation is by convention between users or applications. Locking is only enforced when it is used, and, even then it does not restrict data access.

## Coordinated Resource Recovery (CRR) Services

A file pool server machine may also perform Coordinated Resource Recovery (CRR) services. At most one such server is permitted for a system. The CRR facility ensures data integrity among participating resource managers and distributed applications. When a file pool server machine performs CRR services, the file pool server machine is also a CRR recovery server. It may be dedicated to this work, or it may also perform repository services (not recommended). For more information on CRR and SFS's participation in CRR, see [Chapter 18, "CRR Overview," on page 287](#) and see [Chapter 17, "Participation in CRR \(SFS only\)," on page 279](#).

IBM-supplied (SFS/BFS) repository file pool servers and a dedicated IBM-supplied CRR recovery server are installed during the z/VM installation process. Or, you can generate your own dedicated (SFS/BFS) repository file pool server and dedicated CRR recovery server in a post-installation procedure described in [Chapter 15, "Generating a File Pool and Server," on page 247](#). (See ["Generate a CRR Recovery Server" on page 317](#) for more information regarding why IBM recommends the SFS and CRR services run in separate servers.)

Whether or not file pools elect to participate in CRR, there is an exchange of log names with the CRR Recovery Server when the first SFS request arrives at the file pool server. If no CRR Recovery Server exists (this is called "limp mode"), there is still an attempt to do this exchange of lognames. The failure to complete the exchange is tolerated (does not cause a file pool failure). However, failure to get past this initial exchange can cause repeated attempts to exchange with subsequent SFS requests. As long as there is no need to use CRR services, there is no functional affect on SFS operations, but limp mode causes a serious performance degradation and therefore should be avoided. This means you should install a CRR recovery server even if you do are not using CRR services.

For BFS, which does not use CRR services directly, the installation of a CRR recovery server is not as important. However, because the file pool administrator occasionally uses SFS connections to the file pool server (occurs implicitly with certain administrative functions), the absence of a CRR recovery server (limp mode) can affect the performance of these operations that use SFS connections. Examples of these operations are ENROLL USER, MODIFY USER, QUERY FILEPOOL functions, and FILEPOOL commands (FILEPOOL CONTROL BACKUP, FILEPOOL LIST BACKUP, etc.). IBM recommends you retain the VMSYSR file pool to avoid this performance degradation. (See ["Servers and File Pools" on page 13](#) for more information.)

## FIFO Services

For file pools that contain file spaces of type BFS, and where FIFO special objects exist therein, you may choose to off-load the FIFO services for those FIFO objects to another file pool server in the same system. This option is selected with a FIFO startup parameter for the file pool in which the FIFO objects are defined. This option can have performance benefits by avoiding disruption of normal repository services where FIFO activity is significant.

Off-loading of FIFO services does not affect the definition of the FIFO objects. That is, the objects themselves are not moved or redefined in to the file pool specified in the FIFO startup parameter. Only the read and write activity to the object is affected. Users are not functionally aware of the off-loading of FIFO services.

You should decide to off-load FIFO services if the volume of FIFO activity warrants it.

The default is to retain FIFO services in the file pool where the FIFOs are created. The default is taken when the FIFO startup parameter is not specified.

When a file pool server machine performs FIFO services, the file pool server machine is also a FIFO server. It may be dedicated to this work, or it may also perform repository services.

## Servers and File Pools

The z/VM installation process installs these servers and file pools:

- VMSERVS and its file pool VMSYS
- VMSERVP and its file pool VMPSFS
- VMSERVU and its file pool VMSYSU
- VMSERVR and its file pool VMSYSR

VMSYS is an IBM supplied system data file pool. The system data you load onto VMSYS should not be modified.

VMPSFS is the IBM default product service file pool. The system data you load into VMPSFS should not be modified.

VMSYSU is the user data repository file pool. After installation, VMSYSU (or MAINT 193) contains sample files needed to do the exercises in the following manuals:

- [\*z/VM: CMS User's Guide\*](#)
- [\*z/VM: CMS Primer\*](#)

VMSYSR is the CRR recovery server file pool.

The IBM-supplied file pool server machines and file pools are described in [Chapter 3, “Post-installation Activities,”](#) on page 37.

## File Pool Contents

---

There are three kinds of data stored in a file pool, as [Figure 1 on page 7](#) shows. They are:

- Control data
- Log data
- Repository Data

## File Pool Control Data

A typical file pool holds the files of at least a few dozen people. (The file pool server does not limit the number of users that can be enrolled, although other real limits, such as available DASD space, could place a practical limit on the number that can be enrolled.) To manage these user files, each file pool has a certain number of files and minidisks devoted to control information.

The first item needed to control a file pool is a file known as the POOLDEF file. The POOLDEF file is created when the file pool is generated. It contains information a file pool server machine uses to locate all the minidisks in the file pool and is used every time a file pool server machine is started. <sup>1</sup>

---

<sup>1</sup> Internally, a file pool server machine builds and processes FILEDEF commands based on the information in the POOLDEF file.

Because a file pool server machine uses the POOLDEF file to find its minidisks, server commands that change the definitions of the file pool minidisks automatically update the POOLDEF file. For example, when you add one or more new minidisks to the file pool, you use either the FILESERV MINIDISK or the FILEPOOL MINIDISK command. These commands automatically update the POOLDEF file to show the addition of the new minidisks. Unless this manual explicitly instructs you to do so, never manually update the POOLDEF file, and never erase it.

Each file pool has its own POOLDEF file. POOLDEF is actually the file type. The file name is the ID of the file pool. For example, the file pool ID of the IBM-supplied file pool for user files is VMSYSU, so the POOLDEF file is VMSYSU POOLDEF.

The second item every file pool needs is a *control minidisk*. The control minidisk can be as large as a single DASD volume, but because it is a single minidisk, it cannot span DASD volumes. The control minidisk is formatted for use when the file pool is generated.

It is the control disk that keeps track of all the data in the file pool. To do this, it works with data in chunks rather than bit-by-bit. Each chunk, or *block*, contains 4096(4KB) bytes. The control minidisk contains a map of all the blocks in the file pool. The map tells the file pool server machine which blocks are being used and which are not. Note that because the control disk must keep track of every 4KB block in the file pool, *the size of the control minidisk effectively limits the size of the file pool*. The size of the control disk limits the number of 4KB blocks it can map.

If the control minidisk were to suddenly disappear, the file pool server machine would have no idea where to find anything in the file pool—the file pool would be quite useless. Naturally, you should never modify the control minidisk directly. Nor should you enter CMS commands like COPYFILE or RENAME against it.

The final item of control data is the *catalog storage group* or, as it is often called *storage group 1*. The catalog storage group can have one or more minidisks; you are not limited to a single DASD volume. It, too, is formatted for use by a file pool server machine when the file pool is generated.

Within storage group 1, *catalogs* contain information about the files and directories that exist in the file pool, who owns them, who is authorized to look at them, and so on. As the number of files in a file pool grows, so does the size of the catalogs. You can add minidisks to the catalog storage group by entering the FILESERV MINIDISK or the FILEPOOL MINIDISK command. Later you will see how to do this.

As with all file pool minidisks, you should never manually change the catalog minidisks or enter commands like COPYFILE and RENAME against them.

## File Pool Log Data

To help protect the integrity of the control data, as well as repository data, a server maintains two file pool *log minidisks*. In the file pool logs, the server records changes to the file pool so that if the system encounters a problem and stops in the middle of an operation, the file pool is not damaged. In some cases the file pool logs are also needed so applications can roll back (undo) changes they have made.

To appreciate the importance of the file pool log minidisks and how file pool integrity is maintained, it is necessary to understand a little about how the server processes user requests. Some background knowledge in this area could also be useful in helping programmers with errors they may experience while developing applications that use SFS.

If the server is a CRR recovery server, there are also two CRR log minidisks. The CRR logs record the state of all resources participating in a CRR coordinated transaction. CRR uses the CRR log information to ensure updates to all protected resources within the coordinated transaction are either done or not done.

## Work Units

A central concept to maintaining the integrity of data is the *CMS work unit*. A CMS work unit is a group of related operations that can be either committed or rolled back as a unit. When the operations associated with a work unit are committed or rolled back, new operations can be associated with the same work unit. These operations can also be committed or rolled back. (The work unit is, in a sense, reusable.) Multiple work units can be active in a single user machine.

**Note:** This discussion refers only to recoverable files. Nonrecoverable files are not subject to CMS rollback support. Generally, a rollback causes updates to nonrecoverable files to be committed. For more information about both recoverable and nonrecoverable files, see [z/VM: CMS Application Development Guide](#).

It is important to remember CMS work units are not limited to SFS operations. You should think of CMS work units as *system-level* objects. Any program or set of routines can be coded to support the use of work units. A database program, for example, may support CMS work units. Such a program must ensure that all changes it makes to a resource will either succeed or be unsuccessful as a unit.

**Note:** BFS operations are normally atomic. That is, they are committed individually and implicitly when they succeed and are normally rolled back implicitly when they fail. When BFS operations are included by an application with other operations in a CMS Work Unit, the successful completion (implicit commit) or failure (implicit rollback) of BFS operations does not affect the rest of the work unit. There is only one case where BFS operations can be **explicitly rolled back by the originating application**. This case involves files that are opened, but not yet closed. In this case an explicit rollback reverses the changes to the open BFS files to the state that existed at the time of the opens. This rollback terminates the work unit and rolls back any SFS operations that preceded it in the work unit, but does not affect any other completed BFS operations in the work unit. This is the one case where BFS is responsive to an explicit rollback. However, BFS operations completed before the rollback (closed, for example), are not affected by the explicit rollback.

**Note:** The mixture of BFS operations with SFS operations in the same work unit is only possible when they are directed at separate file pool servers.

File pool services require that all file pool operations are associated with some CMS work unit. When a user enters a CMS command, CMS automatically makes the association. That is, when a user enters a CMS command against a file in a file pool, CMS (internally) associates that command with a work unit and forwards the request to the appropriate file pool server. In its request, CMS also tells the server to commit the changes if all is well. This is transparent to CMS users, so it is possible some CMS users may not know what a work unit is.

In writing programs, the programmer controls which work unit an operation is associated with. The programmer also controls the timing of commits and rollbacks. A programmer may, for example, associate all operations on one set of files with one work unit, and all operations with a second set of files with another work unit. These work units may be active simultaneously within the program. When the program is running, CMS uses a different communication path to the server for each active work unit. At some point in the program, the programmer can explicitly request that all changes in a particular work unit be committed or rolled back. This request is transmitted to the server that does the work.

For SFS, CRR coordinates the committing or rolling back of changes among multiple file pools in a single CMS work unit. This allows a programmer to write to multiple file pools in a single CMS work unit.

CRR will even let a programmer write to multiple file pools and multiple non-SFS resources in a single work unit. But, the non-SFS resources must be participating in CRR. SFS already is participating in CRR. For more information, see [Chapter 18, "CRR Overview,"](#) on page 287 and [Chapter 17, "Participation in CRR \(SFS only\),"](#) on page 279.

CRR also coordinates protected conversations between distributed application programs within the same CRR logical unit of work. This means program A could write to one or more file pools and then begin a protected conversation with program B, which also could write to one or more file pools. All the updates to all the file pools can be committed or rolled back in a single CRR logical unit of work. For more information on CMS work units, see [z/VM: CMS Application Development Guide](#), and for more information on protected conversations, see [Chapter 18, "CRR Overview,"](#) on page 287.

## Logical Units of Work

To provide support of CMS work units, as well as to provide atomicity of file pool server requests (either complete or fail), a file pool server has a request grouping for commit or rollback called a *logical unit of work* (LUW). You can think of a logical unit of work as the server's representation of the work it is doing on behalf of a user. File pool server processing is designed so all operations in a logical unit of work must succeed or be unsuccessful as a unit. A server starts a logical unit of work for a user when the user first



sends a request to the server. It ends a logical unit of work when it receives a request to commit or roll back the changes. Many file pool server requests are defined to be atomic. That is, the CMS work unit completes with one file pool server request and the file pool server logical unit of work is just a single request. BFS requests to a file pool server are normally atomic. One exception to this is for BFS files opened by an application as described in the note in section [“Work Units” on page 14](#). Because multiple CMS work units can be active in a user machine, a user can have multiple corresponding logical units of work active in a server.

To help enforce the rule that all changes in a CMS work unit must succeed or be unsuccessful as a unit, file pool server repositories use the file pool log minidisks and the CRR recovery server uses the CRR log minidisks. The filepool log minidisks are critical to the server's ability to recover from system errors. In the file pool logs, server processing records changes that occur during all logical units of work. It also records when logical units of work are committed or rolled back. In the CRR logs, server processing records sync point activity. If the system encounters a problem and stops, the server can determine what work is incomplete the next time it is started. During this process, known as *restart recovery*, the server rolls back any work that was not committed or rolled back at the time of the error. It also re-does work that was committed, but that was not yet permanently changed in the file pool itself. If the server is a CRR recovery server, restart recovery discards any coordinated transactions (logged sync points) that are not prepared, and the coordinated transactions that are prepared go into resynchronization processing. By doing so, the server eliminates any partial changes to the file pool and satisfies the definition of CMS work units.

Maintaining a file pool log is so important that the server maintains two copies of it, each on a separate minidisk. This lets the server protect the integrity of the file pool even if there is a damaged track on one of the file pool logs or if there is a media error (the device breaks) that makes one of the file pool logs useless. The server can detect a damaged spot on one of the file pool logs and automatically compensate for it on the other. In the case of a DASD error, you still have to replace the *damaged* minidisk, but the integrity is protected by the other file pool log. To protect against media errors, define your file pool log minidisks on different DASD volumes.

The file pool logs are initialized when the file pool is generated. They can be reconfigured (moved to a different device, made larger, made smaller) by running the FILESERV LOG command.

Each file pool log is limited in size to a single DASD volume. And, because the file pool logs are mirrors of each other, both logs must be of identical size. [Chapter 15, “Generating a File Pool and Server,” on page 247](#) describes how to estimate the size needed and provides recommendations on where to place the file pool repository logs and CRR logs.

If the server is a CRR recovery server, there must also be two CRR logs associated with the CRR recovery server in addition to the two file pool repository logs. The CRR log minidisks are critical to the CRR recovery server's ability to complete coordinated transactions, by means of resynchronization processing, when there has been a system error. The CRR recovery server records the state of all protected resources and protected conversation partners that are participating in a coordinated transaction. The CRR resynchronization function uses the data on the CRR logs to commit or roll back all the incomplete updates within a CRR logical unit of work to recover from:

- Application errors
- Server (participating resource manager) errors
- Communications errors
- System errors

Maintaining a CRR log is so important that the CRR recovery server maintains two copies of it, each on a separate minidisk. This lets the CRR recovery server protect the integrity of the coordinated transaction even if there is a damaged track on one of the CRR logs or if there is a media error (the device breaks) that makes one of the CRR logs useless. The CRR recovery server can detect a damaged spot on one of the CRR logs and automatically compensate for it on the other. In the case of a DASD error, you still have to replace the *damaged* minidisk, but the integrity is protected by the other CRR log. To protect against media errors, your CRR log minidisks should be defined on different DASD volumes.



## File Pool Repository Data

The rest of the space in the file pool is for repository (user) data. Hundreds of minidisks can be allocated to a single file pool for user data. To let you maintain some control over which users have files on which DASD volumes, the server uses *storage groups*.

A storage group is a collection of minidisks within the file pool. Storage groups are created during file pool generation and when minidisks are later added to the file pool. Every time you assign a minidisk to the file pool, you must assign it to some storage group. A new storage group is created when you assign a minidisk to it. The minidisks themselves can be as large as a single DASD volume.

Each storage group is identified by a number. The numbers can range from 1 to 32767. Storage group 1 is for catalog information (control information about your files and other repository objects). Storage groups 2 through 32767 are for repository (user) data.

For establishing users of a file pool, the file pool administrator must create file spaces and establish the appropriate authority or permissions to them.

For SFS, both the file space creation and the authorization are established through the ENROLL USER command. This is the case because SFS users are each assigned a file space and that file space name corresponds to the user's own virtual machine ID. Both the authority to connect to the file pool and the authority to use the file space are established by the ENROLL USER command. In a sense, the SFS user is the permanent owner of the assigned file space. Of course, the SFS ownership of a file space by a particular user does not affect that user's ability to share its content with other SFS users through the SFS authorization and sharing facilities.

However, when the ENROLL USER command is used to create BFS file spaces, the administrator has considerable flexibility for controlling the ownership. That is, a BFS can be dedicated to a particular user, as in SFS, or, it can be assigned jointly to several users by creating directories for each of them. Additionally, for BFS, the file space carries no implicit authorization for connecting to the file pool. The various BFS file space creation and enrollment options are discussed in [“ENROLL USER” on page 422](#).

Whether a file space is dedicated to a particular user or shared, the administrator can assign a particular amount of logical space to a file space. This space is specified in units of 4K blocks and is used to limit and measure the amount of DASD space that can be consumed by files in that file space. The file space is also assigned to a particular storage group.

To use a file pool, a person must be enrolled in it or be authorized to use (connect to) the file pool. As a file pool administrator, you can enroll people individually, or you can enroll everyone on your system all together at the same time (which is known as *enrolling public*). When you enroll someone individually, you can also assign the person an amount of space in a particular storage group. The person can then create files using that space. All the user's files will reside on one or more of the minidisks assigned to the storage group.

A single storage group can hold many file spaces. A file space represents a logical data space allocation. That is, no physical DASD space within the storage group is set aside for the user. The file pool server allocates physical disk space to a file space as it is used until the allocation limit (size) of the file space is reached. The size of a file space can be increased or decreased online by the file pool administrator.

Within a storage group, the server manages repository data in 4096-byte blocks. Users do not have to worry where the blocks are stored—the server does that. It allocates blocks to file spaces as needed from the collection of minidisks in the storage group to which the file spaces are assigned.

The server tries to allocate blocks evenly from all physical DASD volumes in the storage group. That is, the server uses a few blocks on one volume, a few on the next, and so on. This helps balance the input or output activity to the volumes in the group. It also means the blocks of a user's file might be scattered among several DASD volumes. There is no way for a user to determine where in the storage group the blocks of a particular file reside. (The user does not need to track where the blocks are—the server does it.)

Because a file pool server consumes blocks on an *as needed* basis, the maximum number of allocated blocks for all file spaces in the file pool can far exceed the actual amount of real DASD space allocated to

it. You would not often allocate all the real DASD needed to support the maximum number of blocks that might be used in all the file spaces. If you did, much of the DASD space would be unused most of the time.

When the limit of real (as opposed to assigned) DASD storage is approached in any storage group, the server issues a warning message to its console. You then add minidisks using the FILESERV MINIDISK or the FILEPOOL MINIDISK command. Because the file pool provides users (through file spaces) with space as they need it (up to the current file space limits), you do not need to worry about assigning minidisks to individual users or file spaces. You can add an entire physical volume to a storage group and let the server worry about managing the space.

You can define as many or as few user storage groups as you wish. Usually what governs the number of storage groups you define is the amount of time it takes to back up a single storage group. As a storage group grows through the addition of minidisks, it takes more time to back up. Before the storage group grows to a size that makes backups inconvenient, define another storage group.

If the users in a storage group use all the physical space you wish to allocate to that storage group, move some file spaces to a different storage group. The FILEPOOL UNLOAD and FILEPOOL RELOAD commands (or the FILESERV MOVEUSER command) can be used to move a file space from one storage group to another. For more information see [“Overview of Moving Users and File Spaces” on page 95](#).

## File Pool Server Machines

---

A *file pool server machine* is a virtual machine that has been set up with the following characteristics:

- Owns the minidisks assigned to a file pool or at least has write authority to the minidisks.
- Has z/VM system directory control statements that let it communicate with other virtual machines.
- Has 32MB of virtual storage, which is adequate for most file pool server machines. XC mode is recommended.
- Has read access to the minidisk on which the FILESERV commands reside. (The FILESERV commands are discussed in this section.) The commands reside on the MAINT 193 minidisk.
- Has read/write access to a CMS minidisk (usually accessed as file mode A). The read/write space is used for the POOLDEF file as well as for temporary work files.

The four IBM-supplied file pool server machines are named VMSERVS, VMSERVP, VMSERVU, and VMSERVR. The VMSERVS machine owns the minidisks for the VMSYS file pool, which contains system files. The VMSERVP machine owns the minidisks for the VMPSFS file pool, which contains product service files. The VMSERVU machine owns the minidisks for the VMSYSU file pool, which is for your use. The VMSERVR machine owns the minidisks for the VMSYSR file pool, which is for the CRR recovery server. These server machines have all of the preceding characteristics. [Chapter 15, “Generating a File Pool and Server,” on page 247](#) describes how to create additional server machines and file pools.

Aside from the preceding characteristics, the virtual machine is quite ordinary. It has a console and the usual virtual devices—reader, punch, printer, tapes—and perhaps owns additional minidisks. It may also have access to a file space in another file pool. Of particular interest is the server operator console. [Figure 3 on page 19](#) shows a file pool, a file pool server machine, and a file pool server machine operator console. (The actual display device can be any that z/VM supports.)

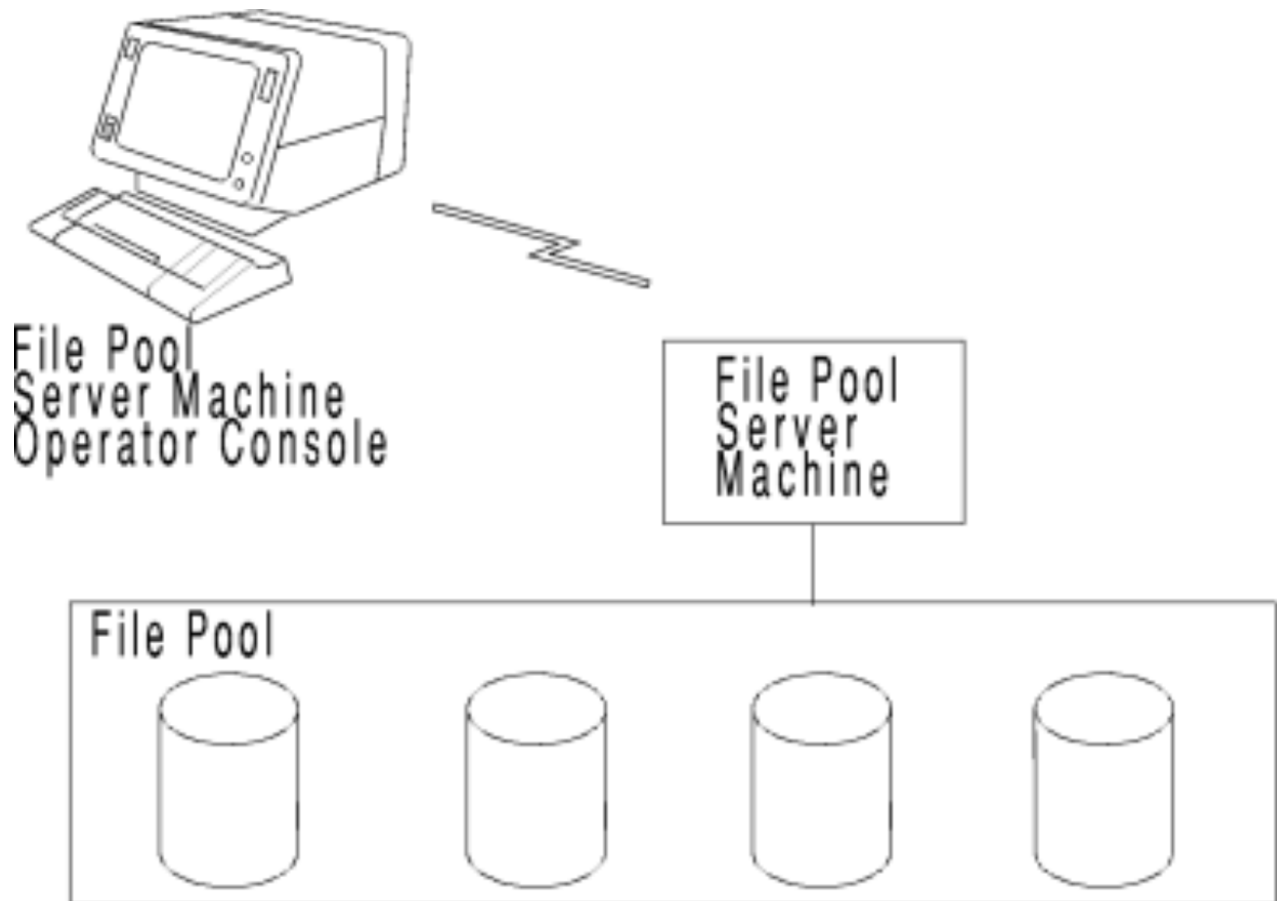


Figure 3. A File Pool Server Machine Operator Console

For brevity, the file pool server machine operator console is referred to as the *operator console*. From the operator console, you can enter FILESERV commands, which operate on a file pool. (CMS must be running in the file pool server machine before any FILESERV command can be entered.)

All FILESERV commands, except one, perform maintenance operations on a file pool. When a maintenance command is running against a file pool, the file pool is not available to other users. Because of this, the commands are known as *dedicated maintenance mode* commands. All file pool server dedicated maintenance mode commands are listed in [Table 5 on page 19](#).

When any of the commands in [Table 5 on page 19](#) is entered, the server machine is running in dedicated maintenance mode. When the command completes, dedicated maintenance mode ends.

Table 5. Dedicated Maintenance Mode Commands

Command	Function
FILESERV BACKUP	Makes a backup copy of the file pool control data (which is the POOLDEF file, the control minidisk, and the catalog storage group)
FILESERV CRRLOG (CRR-only)	Reconfigures the CRR log minidisks
FILESERV DEFBACKUP	Defines the output file for subsequent backups
FILESERV DEFAUDIT	Defines the output file for any subsequent security audit trace
FILESERV DEFCCRLOG	Updates CRR log minidisk definitions in the CRR recovery server's POOLDEF file
FILESERV GENERATE	Generates a new file pool
FILESERV LIST	Lists the contents of the file pool catalogs

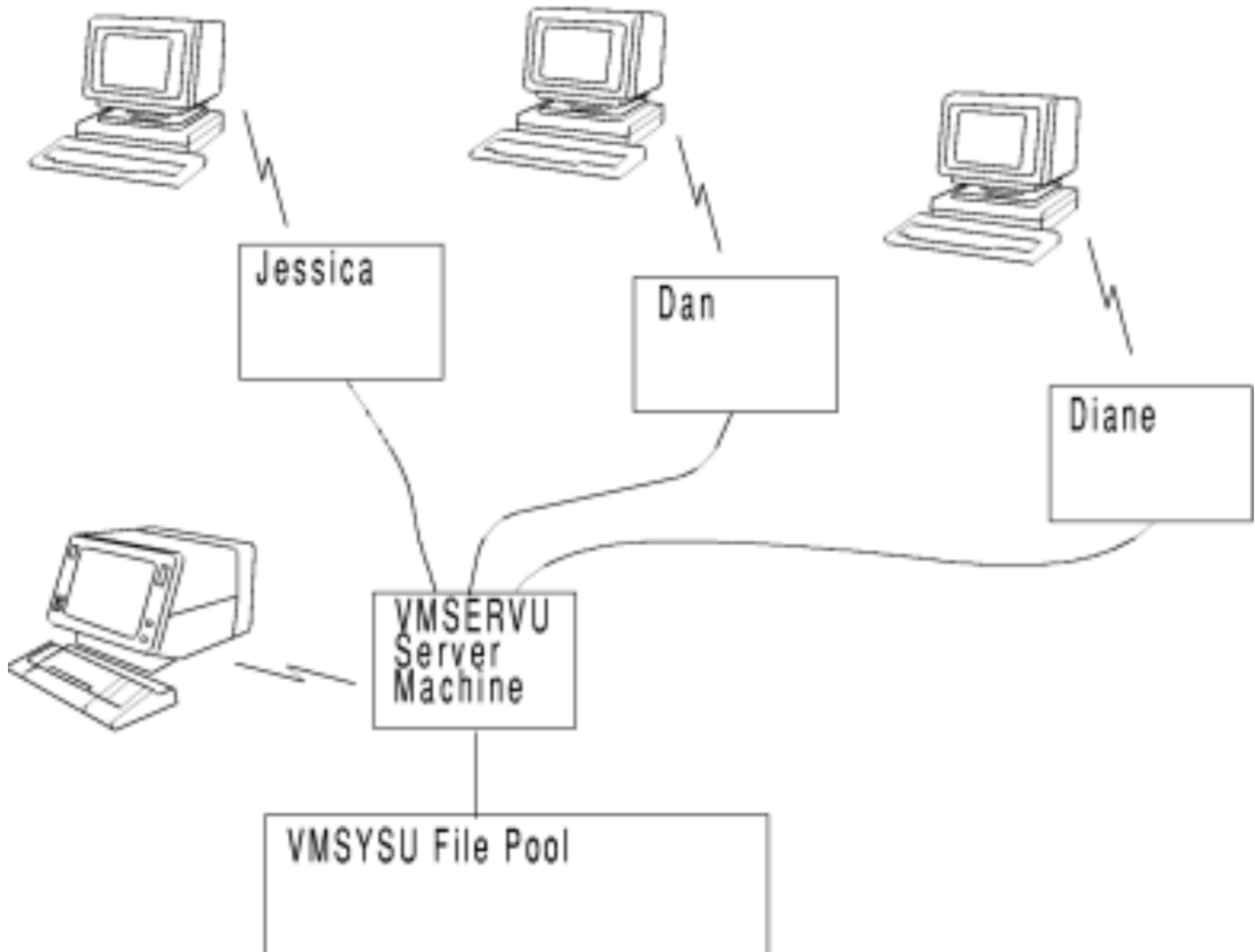
*Table 5. Dedicated Maintenance Mode Commands (continued)*

<b>Command</b>	<b>Function</b>
FILESERV LOG	Reconfigures the file pool log minidisks
FILESERV MINIDISK	Adds storage group minidisks to a file pool
FILESERV MOVEUSER	Move a user or file space from one storage group to another
FILESERV REGENERATE	Expands the maximum size of a file pool
FILESERV REORG	Reorganizes the file pool catalogs for better performance

You can control various aspects of FILESERV command processing by specifying *startup* parameters in a CMS file. Because this file's file type must be DMSPARMS, a file containing startup parameters is often referred to as DMSPARMS files. All startup parameters are described in [Chapter 20, "File Pool Server Startup Parameters,"](#) on page 339.

You do not need to memorize the list of FILESERV commands or the startup parameters that can be used with them. Later chapters in this manual describe all the FILESERV dedicated maintenance mode commands and tell you when they should be used and with which startup parameters.

Only one FILESERV command, named FILESERV START, is not a dedicated maintenance mode command. It is the command that starts the server for multiple user access. When you enter FILESERV START, the file pool is opened for service through communications from other *user* or *client* virtual machines, which means users and applications can create and access files in the file pool. For SFS it also means that participating applications can perform CRR logging for coordinated transaction processing. The server is said to be running in *multiple user mode*, which is shown in [Figure 4 on page 20](#).



*Figure 4. Multiple User Mode Processing*

When the server is running in multiple user mode, it cannot be used for other applications. Multiple user mode processing requires the continual use of the virtual machine. Therefore, after entering FILESERV START, you will usually want to run the server in *disconnected mode*. In disconnected mode, the server virtual machine continues operation, but your terminal is disconnected from the z/VM system. (See the CP DISCONN command in *z/VM: CP Commands and Utilities Reference*.)

When you disconnect from a virtual machine, any messages generated by that machine are usually lost. (Because you have disconnected your terminal, there is no place for the messages to be displayed.) For some applications, it does not matter whether the messages it generates are lost. Server messages, however, are important and should not be lost or ignored.

To avoid losing server messages, the use of the Single Console Image Facility (SCIF) is strongly recommended. SCIF lets you define a *secondary user* to which all messages are sent when a virtual machine (known as the *primary user*) is running in disconnected mode. In this case, the server machine is the primary user. When the server is running in disconnected mode, all server messages appear on the console of the secondary user. [Figure 5 on page 21](#) shows server operation when a primary user is running in disconnected mode.

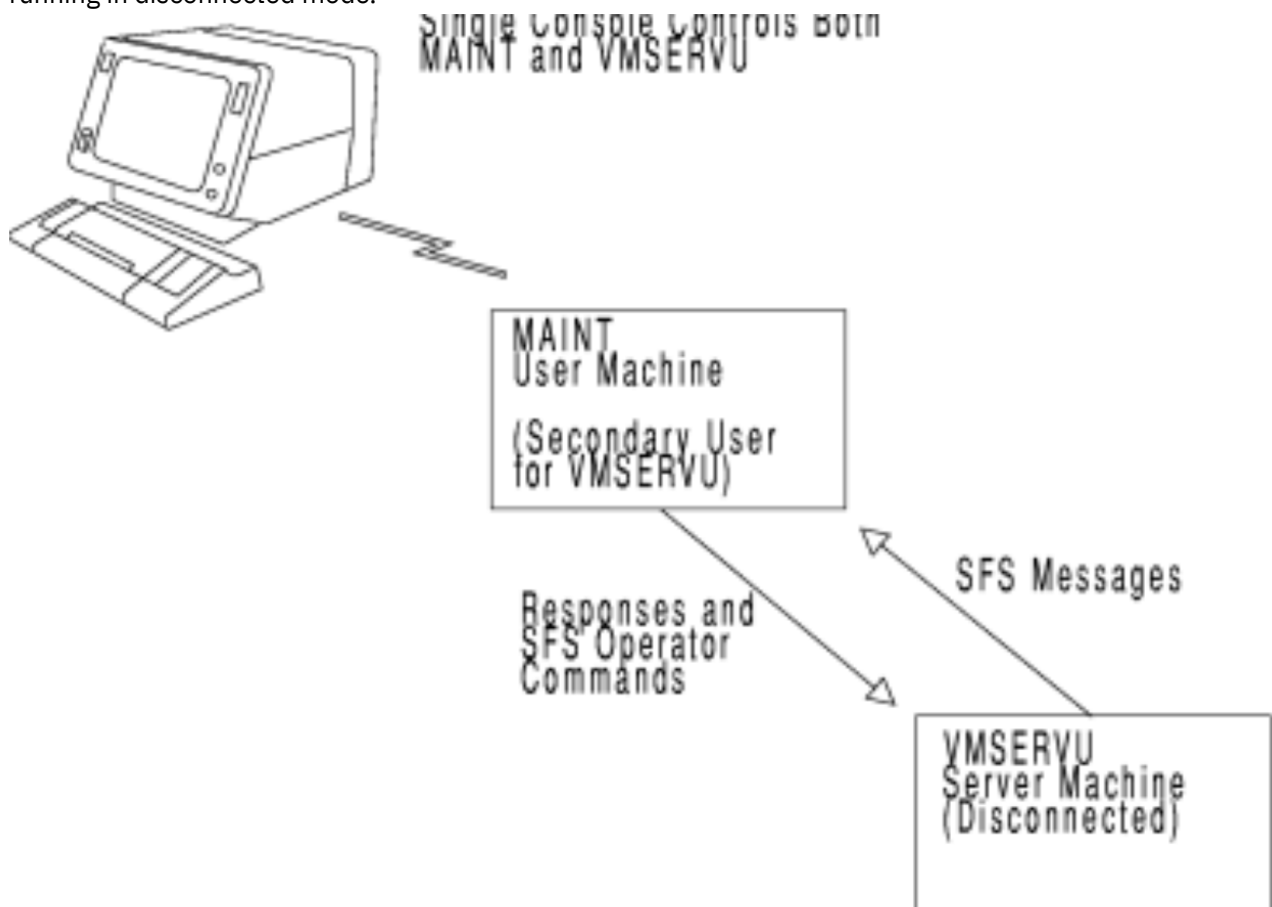


Figure 5. Disconnected Server Operation

VMSEVRV, VMSEVRP, VMSEVRU, and VMSEVRV file pool server machines are set up so that MAINT is the secondary user. When VMSEVRV, VMSEVRP, VMSEVRU, or VMSEVRV is running in multiple user mode and is disconnected, all of their messages are routed to the console of the MAINT machine. Assuming that MAINT is logged on, the messages are displayed with a prefix that lets you know where the message is from. If MAINT (or any other defined secondary user for the server) is not logged on or is running in disconnected mode, the messages are lost. You can save the server's messages, even though the secondary user is not logged on or is running in disconnected mode, by using the CP SPOOL CONSOLE command in the server's and secondary user's PROFILE EXEC. For more information, see [Chapter 3, "Post-installation Activities,"](#) on page 37.

## File Pool Server Machines

Some server messages require a response. To reply to a message from a secondary user console, use the CP SEND command. (Instructions for using the CP SEND command are in [Chapter 5, "Operation,"](#) on page 55.) You can also use the CP SEND command to enter server *operator* commands.

Server operator commands let you control multiple user mode execution, and CRR processing. They can be used only after FILESERV START has been issued and the server has completed its initialization process. You can enter them at the server machine console or from its secondary user console. All file pool repository and CRR recovery server operator commands are listed in [Table 6 on page 22](#).

Table 6. File Pool Repository and CRR Operator Commands

Command	Function
AUDIT	Starts an audit trace of security. (During an <i>audit trace</i> , the server records information about file pool usage and about operator intervention into CRR activity in a file. Later you can format and review the information in the file to determine if there were any unauthorized attempts to use file pool resources or to intervene into CRR activity).
BACKUP	Makes a backup copy of the file pool control data.
CRR (CRR-only)	Allows operator to do CRR log management, problem management, and resynchronization management.
DEFBACKUP	Defines the output file for subsequent control data backups.
DISABLE	Prevents users from accessing a particular storage group or file space.
ENABLE	Allows access to a previously disabled storage group or file space.
ERASE LUNAME (CRR-only)	Removes history of all previously forced SFS logical units of work from the file pool logs.
ETRACE	Starts or stops an external trace of server machine execution. (Records are written to a CP spool file.)
FORCE	Stops a user's current work with the file pool, and commit or roll back SFS prepared work.
GRANT ADMIN	Grants file pool administration authority to another user.
ITRACE	Starts or stops an internal trace, during which records are written to an internal buffer, of server APPC/VM communication activity.
QUERY DEFBACKUP	Shows the current default assignment for the control data backup file, or shows the destination where the last backup file was created.
QUERY DISABLE	Tells whether a storage group or file space is disabled and who disabled it.
QUERY LOGTABLE (CRR-only)	Display the specified LU name and transaction program name (TPN) for entries in the SFS log name table.
QUERY PREPARED (CRR-only)	Display information about SFS prepared work and forced work.
REVOKE ADMIN	Revokes file pool administration authority from a user.
STOP	Stops multiple user mode processing.

For brevity, file pool server operator commands are referred to, simply, as operator commands. This manual describes all the operator commands in later chapters.

If you scanned the commands in [Table 6 on page 22](#), you may have noticed the term *file pool administration authority*. File pool administration authority is a special authority. Those who have it can control file pool resources and can read from or write to other users' files and directories. Remember, though, that the person who controls the operator console has the "keys" to the file pool server machine

and the file pool. That person can grant file pool administration authority to any other user, and has access to all the file pool minidisks.

## Multiple Production File Pools

You can have multiple file pools on one z/VM system. A z/VM system normally has multiple file pools. In addition to the VMSYS, VMPSFS, VMSYSU, and VMSYSR IBM-supplied file pools, which are generated during z/VM installation, there can be one or more production file pools too. The production file pool is the file pool the end users primarily deal with and is the file pool they are enrolled in with space allocation. There may also be a read-only file pool and one or more additional file pools for test purposes.

Small and intermediate systems would normally require and would normally choose to have just one production file pool. On large systems, multiple production file pools may be appropriate. To plan for this, follow these steps:

1. Estimate maximum file pool size. This will determine how many production file pools are required. See [“Step 1: Estimate Maximum File Pool Size”](#) on page 23.
2. Decide on whether you wish to have additional production file pools. See [“Step 2: Decide on Additional Production File Pools”](#) on page 24.
3. Decide how users will be enrolled in the production file pools. See [“Step 3: Enroll Users in File Pools”](#) on page 24.
4. Begin implementing the file pools. See [“Step 4: Implement Multiple File Pools”](#) on page 24.

### Step 1: Estimate Maximum File Pool Size

There is a practical upper limit to the rate at which any given file pool server can process requests. The number of enrolled users required to generate this request rate will vary greatly depending upon how often these users are logged on and, when logged on, how active they tend to be.

For SFS, enrolled users can be directly associated with file spaces created with the ENROLL USER command. For BFS, which offers more flexibility in the relationship to users and file spaces, you should consider enrolled users to be all BFS users that can work with BFS file spaces. When estimating the maximum file pool size for your system, the following formula should be used:

```
maximum enrolled users = 300 * (# system defined users / # system active users)
```

**System defined users** are defined in the system directory which equates to the total users for the system. **System active users** is the number of logged on users who have interacted with the system (in any way) during a typical one-minute interval. Maximum enrolled users is the number of users that primarily use the file pool. In general, this will be the number of users enrolled in the file pool and have one or more file spaces with available storage blocks assigned thereto.

#### Note:

1. This formula was generated to reflect the limiting case where all the enrolled users place all their files in that file pool. Much higher numbers of enrolled users can be supported by one file pool in partial file pool usage scenarios where the users continue to have many of their actively referenced files on minidisks. Such partial usage is expected and typical when file pools are first made available. However, it is strongly recommended you limit the maximum size of each production file pool based on the assumption of eventual full usage. This is to ensure the file pool can continue to handle the load that is placed on it by its enrolled users, even if all of those users ultimately put all of their files in that file pool. In this way, you avoid later having to transfer some of the users to another production file pool. Although it is possible to transfer a user to another file pool, the procedure for doing so is a fairly complex task. (See [“Overview of Moving Users and File Spaces”](#) on page 95 for more information.)
2. The limiting factor that determines the maximum practical file pool size can vary from one installation to another, but the most likely factor will be that rollbacks because of deadlock rise to unacceptable levels. See [“Monitoring Rollbacks Because of Deadlock”](#) on page 65 for background information on rollbacks that arise from deadlock situations and how to monitor those rollbacks.



3. Another factor for consideration is the recovery time for control data. See the section on recovery in [z/VM: Performance](#).

### Step 2: Decide on Additional Production File Pools

After you have established how many production file pools you require, the next step is to decide whether you wish to have additional (more than the minimum number required) production file pools. To help you make that decision, this section lists potential disadvantages and advantages associated with having more production file pools than the minimum requirements:

#### Disadvantages

- End user perspective:
  - There will be more cases where a file that a user wishes to create an alias on resides in a different file pool. SFS does not support aliases on files that reside in another file pool.
  - There will be more cases where a user wishes to use files that reside in a file pool that is not the one that user is enrolled in. In such cases, the user will need to remember and supply the name of the other file pool when accessing the directory or directories those files reside in.
- System administrator perspective:
  - Administrators and system programmers who concern themselves with the operation of the system as a whole will have more production file pools to deal with. This can increase the level of effort associated with doing system-wide administration.

#### Advantages

- The size of each file pool can be set up to match the desired scope of administration. For example, one administrator could be assigned to each file pool and could carry out most of his or her daily administrative tasks independently. Furthermore, security is enhanced as each administrator only has administrator authority on the one file pool for which he or she is responsible.
- If one file pool becomes unavailable, users assigned to other file pools are unaffected (except when they wish to use shared data in that file pool).
- In the event of a DASD error on one of the volumes containing the control minidisk or one of the storage group 1 minidisks, recovery time is less with a smaller file pool.
- There is greater flexibility for responding to future changes in the hardware configuration. For example, if you were moving from a large processor to a smaller processor, it would be much easier to move a whole file pool to the smaller processor than it would be to just move some of the users from an existing file pool to that new processor.

### Step 3: Enroll Users in File Pools

This is often best done in conjunction with the previous step (deciding whether to have additional production file pools). Here are some things to consider:

- It is helpful to put into the same file pool those users who are most likely to be sharing data with one another. This minimizes the end user disadvantages referred to in [“Step 2: Decide on Additional Production File Pools”](#) on page 24.
- If each production file pool is to have a separate administrator, security and other considerations suggest you may wish to put separate organizations or projects into separate file pools.
- If it does not matter which file pool any given user is enrolled in, you can implement the production file pools on an as-needed basis. (See [“Step 4: Implement Multiple File Pools”](#) on page 24.)
- See [Chapter 6, “Managing Users and File Spaces,”](#) on page 71 for more information.

### Step 4: Implement Multiple File Pools

After you have determined the maximum number of users you can enroll in each file pool, how many production file pools you will have, and how you plan to assign users to file pools, you are ready to start implementing those file pools.



**Staging the Implementation:** If you have determined you will ultimately have multiple production file pools, give some consideration to how you will stage their implementation. Basically, the two choices are to start with one production file pool and add others as needed, or to immediately implement some or all of the production file pools.

The first approach is workable if it does not matter which users are placed in which file pools. Under those conditions, you could choose to implement just the first production file pool and enroll users in it up to the maximum you have established. You would then implement the next production file pool and enroll subsequent users in it (up to the maximum), and so forth.

If, on the other hand, you wish to map specific categories of users to different file pools, you will need to implement the production file pools that are intended for the groups of users that you currently plan to enroll. You could defer implementation of your other file pools until the time has come to enroll the additional categories of users you have planned for those file pools.

**DASD Configuration and Tuning Guidelines:** All the DASD configuration and file pool server tuning guidelines that apply to implementing a single production file pool on a system also apply to the implementation of each file pool on a system that contains multiple production file pools. Those guidelines are described in [z/VM: Performance](#).

In addition, the following consideration applies:

- Any given DASD volume should only contain minidisks from one file pool. This is to ensure a DASD error on any given volume will only affect one file pool. (Consequently, you can disregard this recommendation for cases such as test file pools where recovery is not a significant consideration.)

## File Pool Administration Machines

---

A *file pool administration machine*, or *administration machine*, is a virtual machine that has been granted file pool administration authority. From an administration machine, you can control resources in the file pool and execute all of the file pool administration commands listed in Table 7 on page 26. That is, you can enter commands that enroll users, create BFS file spaces, or allocate space in the file pool to users or file spaces. You are also able to enter commands that back up or restore users' data or that inquire about the status of the file pool. For SFS, you can place directory control directories and their file data into data spaces and monitor the use of those data spaces. All of these tasks are done while the server is running in multiple user mode.

File pool administration authority also lets you do anything to users' SFS objects that the SFS users themselves can do. You can, for instance, read from or write to any user file in the file pool. You can also grant or revoke authority on files you do not own, or create and drop aliases in other users' directories. Because file pool administration authority is so powerful, an installation should **strictly control the granting of that authority**.

For BFS, a user designated as a superuser has permission for all BFS objects in all file pools in the local z/VM system. However, a superuser has no authorization for SFS objects or for file pool administration functions in those individual file pools.

In addition to those capabilities described above, a file pool administrator also has some permissions for the BFS objects and functions, but these are limited in scope to the file pool for which the administration authority applies. A file pool administrator can operate on BFS objects using BFS functions or through a limited set of SFS functions that operate on either SFS or BFS objects. When operating on BFS objects, a file pool administrator basically has superuser authority, but this is limited to objects in the administrators file pool. See Chapter 8, “Security,” on page 137 for additional restrictions.

If your virtual machine has file pool administration authority, you can still use it to do other work. You can edit files or run other programs as well as enter file pool administration commands.

There are three ways you can give a virtual machine file pool administration authority:

1. You can identify it as an administration machine in startup parameters you supply when you start multiple user mode in the server machine.

2. The file pool server machine operator can grant file pool administration authority after starting multiple user mode. (The server operator enters a GRANT ADMIN operator command.)
3. A user machine that already has file pool administration authority can grant that authority to another virtual machine. (The user with file pool administration authority enters an ENROLL ADMINISTRATOR command while the server is processing in multiple user mode.)

Administration authority lasts only until file pool server processing ends. When file pool server processing is next started, administration authority must be granted again. To maintain a *permanent* set of administrators, you specify the desired user IDs in the startup parameters. Even though those administrators, like any others, lose their authority when server processing ends, they will automatically be granted that authority again when the server is next started and processes the startup parameters.

A virtual machine that has file pool administration authority is referred to as an *administration machine*. In most installations the virtual machine that serves as the secondary user is also an administration machine. That is how the IBM-supplied file pools are set up. The file pool server machines define MAINT as the secondary user. They also grant administration authority to MAINT for the file pools they manage.

After VMSERVS, VMSERVP, VMSERVU, and VMSERVR are running in multiple user mode and are disconnected, any person who logs on to MAINT can enter file pool server operator commands as well as administration commands to both server machines. All file pool server and CRR recovery server administration commands are shown in [Table 7 on page 26](#).

Table 7. File Pool Administration Commands

Command	Function
DATASPACE	Makes an SFS directory and its file data eligible for use in a data space. The DATASPACE command can also remove eligibility.
DELETE ADMINISTRATOR	Revokes file pool administration authority from another user.
DELETE LOCK (for another user)	Unlocks a file or directory that another user explicitly locked. Users without administration authority can also enter this command to delete their own locks, but they cannot delete the locks of another user.
DELETE PUBLIC	Removes the ability of all z/VM users to connect to the file pool.
DELETE USER	Removes an SFS user or a BFS file space and the contained objects from the file pool.
ENROLL ADMINISTRATOR	Gives file pool administration authority to another user.
ENROLL PUBLIC	Lets all z/VM users connect to the file pool and exercise any SFS authorities that SFS users have granted to PUBLIC.
ENROLL USER	Enrolls an SFS user, with or without file space, or creates a BFS file space in the file pool.
FILEPOOL BACKUP	Makes a backup of all the data in a storage group.
FILEPOOL CLEANUP	Frees resources that remain held after certain errors occur during FILEPOOL BACKUP or FILEPOOL RESTORE.
FILEPOOL CONTROL BACKUP	Schedules a control data backup for a specified file pool.
FILEPOOL DISABLE	Disables (locks) a storage group or file space for write access (SHARE) or all access (EXCLUSIVE).
FILEPOOL ENABLE	Reinstates use of a locked or disabled storage group or file space.
FILEPOOL FILELOAD	Restores one or more files from a copy of the storage group data created by FILEPOOL BACKUP.
FILEPOOL FORMAT AUDIT	Formats a security audit trace file for display. (Security audit traces are started by the AUDIT operator command.)

Table 7. File Pool Administration Commands (continued)

Command	Function
FILEPOOL LIST BACKUP	Lists SFS base files and Byte File System (BFS) pathnames that are contained in a user storage group backup or unload file created by the FILEPOOL BACKUP or FILEPOOL UNLOAD command.
FILEPOOL LIST MINIDISK	Lists the files that have at least one data block on a specific file pool storage group minidisk.
FILEPOOL MINIDISK	Adds one or more storage group minidisks to a file pool.
FILEPOOL RELOAD	Uses the output of the FILEPOOL UNLOAD command to recreate the contents of SFS or Byte File System (BFS), file space(s) in another storage group or file pool, or to restore the contents of user file space(s) in the same storage group or file pool.
FILEPOOL RENAME	Renames an SFS user (and the associated file space) or a BFS file space, while retaining the associated file pool authorizations/permissions, objects, and locks.
FILEPOOL RESTORE	Loads data into a storage group from a storage group backup. (Resets the data in the storage group to the state it was in at the time the backup was made.)
FILEPOOL UNLOAD	Use the FILEPOOL UNLOAD command to create a file that contains a copy of all data in an SFS or Byte File System (BFS) file space or storage group and all associated file pool catalog data. Then the FILEPOOL RELOAD command can use the resultant file (from UNLOAD) to recreate the file space or storage group.
MODIFY USER	Changes the amount of the file space allocated to an enrolled SFS user or BFS file space.
QUERY ACCESSORS - SFS-only	Displays information about current accessors of directory control directories, optionally including usage of data spaces.
QUERY DATASPACE - SFS-only	Displays information about the eligibility of directories for use in a data space.
QUERY FILEPOOL AGENT	Displays information about the users or internal processes for which the file pool server is currently doing work.
QUERY FILEPOOL CATALOG	Displays information about the file pool catalog space.
QUERY FILEPOOL CONFLICT	Displays information about lock conflicts in the file pool. Users without administration authority can also enter this command. When this command is entered by a file pool administrator, however, CMS displays an additional column of information.
QUERY FILEPOOL COUNTER	Displays a logically ordered list of counters for a file pool server processing against a file pool.
QUERY FILEPOOL CRR (CRR-only)	Displays CRR counter information.
QUERY FILEPOOL DISABLE	Determines disable lock information.
QUERY FILEPOOL LOG	Displays information about the file pool log minidisks.
QUERY FILEPOOL MINIDISK	Displays all file pool minidisk information.
QUERY FILEPOOL OVERVIEW	Displays overview information about a specified file pool.
QUERY FILEPOOL REPORT	Displays information about server processing and about the file pool.

Table 7. File Pool Administration Commands (continued)

Command	Function
QUERY FILEPOOL STATUS	Displays information about server processing and about the file pool.
QUERY FILEPOOL STORGRP	Displays information on storage groups.
QUERY LIMITS (for another user or for all users)	Displays information about SFS user's or BFS file space status. SFS users can enter this command to display information about their own file space status, or the file space status of another user. BFS users can use this command for any BFS file space.
SET THRESHOLD (for an SFS user)	Indicates when a warning message and return code should be issued which indicates that a specified amount of a SFS user's allocated file space in a file pool has been used. Users without administration authority can also enter this command to set their own threshold limit, but they cannot set the threshold limit of other users. This command is not applicable to BFS file spaces.

All the commands that require file pool administration authority are described throughout this manual.

Some administration functions are available as Callable Services Library (CSL) routines. You can use CSL routines in assembler programs, in high-level language programs, and in REXX execs. For information about CSL routines see:

- [z/VM: CMS Application Development Guide](#)
- [z/VM: CMS Callable Services Reference](#)
- [z/VM: CMS Application Development Guide for Assembler](#)

## A Note about File Pool Maximums

All the maximum values documented in this manual for startup parameters, control statement inputs, and so on, are architected limits. That is, they are the real limits the server code can process. These limits were intentionally chosen to exceed the practical capacity of many of today's systems.

Although there is no architected limit to the number of users you can enroll in a file pool, there is some practical limit. Practical limits are difficult, if not impossible, to set accurately because they vary with factors often beyond your control. For example, suppose you want to enroll 4000 users in the file pool. SFS uses some space in the file pool catalogs to record enrolled users, and BFS does similarly for BFS file spaces you create. Although minimal space is needed to record each user, you might discover (from server messages) you need to add physical space to storage group 1. If you do not have available DASD space, your practical limit for enrolled users is something less than 4000.

When specifying startup parameters, keep in mind that setting startup parameters unrealistically high can cause the server to attempt to use resources that exceed the capacity of your system. The CATBUFFERS startup parameter, for example, specifies the number of 4096-byte virtual storage buffers the server uses when reading and writing file pool catalogs. The architected limit of CATBUFFERS is 32767. That is, the server will accept a CATBUFFERS value of 32767. Specifying 32767, however, makes the server attempt to acquire over 100MB of virtual storage.

To avoid exceeding practical maximums, it is recommended you use the defaults and follow the guidance provided in this manual for selecting values. Choosing arbitrarily high values to avoid server administration can cause server errors or poor performance. See [Appendix B, "File Pool Server Maximums,"](#) on page 685 for information on some of the maximums of SFS.

## Other File Pool Facilities

Servers have other functions you may find beneficial. The use of any of these facilities is optional. You can decide whether you want to use them after you have become experienced in working with servers. You can also start or stop them at any time.

## Accounting Facility

Servers can contribute records to the z/VM accounting file. The records describe the amounts of various system resources consumed by users of the server. Records are also generated for server processes, such as startup processing, that cannot be attributed to a single user.

In addition to the information that servers provide in the accounting records, there is a 16-byte area in which you can provide information. This area can be used for installation-dependent accounting information, such as internal account numbers. For more information about the accounting facility, see [Chapter 12, “Accounting,” on page 223](#).

## Security Audit Trace

File pool servers include a *security audit trace* facility that lets you monitor or audit the security of file pool authorization/permission attempts, as well as operator intervention into CRR activity. When a security audit trace is active, a server traces users' attempts to access file pool resources and traces operator intervention into CRR activity. The server records this information in a file you can later format and review. For more information about security auditing, see [“Auditing Security” on page 143](#).

## External Security Manager

File pool servers let you use an external security manager to check authorizations when users attempt to access file pool resources. The external security manager can be coded such that it either co-exists with SFS's authorization or BFS permission checking or completely replaces them.

To activate an external security manager for a particular file pool server machine, specify the ESECURITY startup parameter. For information about using an external security manager, see [“Using an External Security Manager” on page 161](#). For information about coding your own external security manager, see [“Writing Your Own ESM Exit Routines” on page 176](#).

**Note:** External security managers are not recommended for CRR recovery servers.

## Service Aids

Servers include four facilities to help diagnose defect problems should they occur. The facilities are:

### Mini-dumps

When server processing unusually terminates, it displays internal information that existed at the time of the error. This information is known as a *mini-dump*. Mini-dumps are displayed on the server machine console.

See [z/VM: Diagnosis Guide](#) for more information about SFS mini-dumps.

### Dumps specified at startup

Servers have startup parameters that allow you to request various kinds of dumps should server errors occur. You can request either a partial or full virtual machine dump. Or you can request no dumps be made.

The startup parameters DUMP, FULLDUMP, and NODUMP control what a server will dump. The default is for the server to generate partial virtual machine dumps (DUMP). For a description of the startup parameters, see DUMP, FULLDUMP, and [NODUMP](#).

### Tracing facilities

Servers include facilities that let you trace server processing. Two kinds of tracing are provided: internal tracing, and external tracing. Internal trace processing is referred to as *ITRACE processing*. External trace processing is known as *ETRACE processing*.

Both kinds of traces can be started in a server machine by startup parameters or by server operator commands (in multiple user mode). To start a trace using startup parameters, specify ETRACE or ITRACE startup parameters in the DMSPARMS file. See [Chapter 20, “File Pool Server Startup Parameters,” on page 339](#) for a description of the ETRACE and ITRACE startup parameters.

To start a trace using server operator commands use the ETRACE or ITRACE operator commands (or both). See [“ETTRACE” on page 429](#) for a description of the ETRACE operator command, and [“ITRACE” on page 542](#) for a description of the ITRACE operator command.

### FILESERV LIST

This dedicated maintenance mode command lists the contents of the file pool catalogs. Such a list could be useful if you suspect the file pool catalogs are damaged. See [“FILESERV LIST” on page 521](#) for a description of the FILESERV LIST command.

If you suspect there is a defect in the server code, you should save any console messages and any other data related to the problem. See *z/VM: Diagnosis Guide* for more about debugging file pool servers and CRR recovery servers. They also describe what data you should gather should it be necessary to call your IBM Support Center for assistance.

If it is necessary to call your IBM Support Center, you might be asked to gather more information about the problem by changing the dump-related startup parameters, entering a FILESERV LIST command, or by using the trace facilities. Unless your support group directs you to do so, there is no need to use any of the preceding service aids.

## What to Do Next

---

What you should do next varies depending on what file pools you will be administering and whether someone has already generated those file pools for you. Consider the following:

1. If you are the administrator of the VMSYS, VMPSFS, VMSYSU, or VMSYSR file pools, you should now follow the post-installation instructions in the [Chapter 3, “Post-installation Activities,” on page 37](#).
2. If you are planning to generate your own file pool, follow the instructions in [Chapter 15, “Generating a File Pool and Server,” on page 247](#).
3. If you are the administrator of a file pool that someone else at your installation generated, you should start the file pool server in multiple-user mode (see [“Starting Multiple User Mode Processing” on page 55](#)), and enroll the initial set of users (see [“Enrolling SFS Users and Creating BFS File Spaces” on page 71](#)). After the file pool server is running in multiple user mode and the users have been enrolled, you should learn how to monitor the file pool server as described in [“Monitoring Server Operation” on page 56](#). You should also devise a procedure for backing up the file pool if one has not been devised for you (see [Chapter 7, “Recovery Procedures,” on page 101](#)).
4. If you are planning to install DFSMS/VM to manage one or more file pools, see [z/VM: DFSMS/VM Planning Guide](#).

Finally, before beginning any procedure described in this manual, please read all the directions for that procedure. Some procedures require passwords to be supplied, tapes to be mounted, processing to be halted, and so on. Read the directions before starting a procedure so you are sure you have everything necessary to complete it.

## Chapter 2. Installation Planning for File Pools

During z/VM installation, you load the following IBM-supplied servers and file pools from the z/VM System DDR image:

- VMSERVS (which manages the VMSYS file pool)
- VMSERVP (which manages the VMPSFS file pool)
- VMSERVU (which manages the VMSYSU file pool)
- VMSERVER (which manages the VMSYSR file pool)

This chapter lists the benefits of installing and using the IBM-supplied file pools.

### VMSYS and VMPSFS

VMSYS is the IBM-supplied system data file pool managed by the VMSERVS server machine while VMPSFS is a file pool managed by the VMSERVP server machine. VMSYS holds system data and VMPSFS holds product service data. The benefits of VMSYS and VMPSFS are:

- Significant DASD space savings over minidisks are possible because file pools make more efficient use of DASD space for SFS and BFS repository data.
- Easier serviceability of products on VMSYS and VMPSFS may be possible.

For example, for the experienced file pool administrator, it would be easier to add space to a file pool than to increase the size of a minidisk and then copy all the files from the smaller minidisk to the larger minidisk.

- Higher availability of system data may be possible.

System data is read-only data that is available to everyone in the system and is needed by all systems (for example, goodies files) or specific product users. Placing system data into a separate, local, read-only file pool such as VMSYS (as opposed to a production file pool like VMSYSU) has the advantage of offering the users high availability of this often critical data.

Using a read-only file pool is a good choice for placing directories you plan to use with SFS data spaces because data spaces are only used for read-only (DIRCONTROL directory) purposes. For more information on SFS data spaces, see [“Selecting a File Pool” on page 242](#).

If you plan to install DFSMS/VM, the installation of VMSYS is required. There may be licensed programs you will be installing that may require or optionally use VMSYS. VMSYS is appropriate for licensed programs because it is a local file pool, which appropriately restricts usage of the contained programs to the local system. See the appropriate licensed program's planning documentation.

### BFS Root File Space in the VMSYS File Pool (Default)

Insure you are familiar with the concepts that will be discussed by referring to the section on setting up OpenExtensions in *z/VM: OpenExtensions User's Guide*.

The installation process establishes the Byte File System (BFS) root file space and root directory in the VMSYS file pool. This BFS file space is named ROOT.

Even though you may not have immediate need for the BFS root, you are advised to retain it for possible future usage. Because the few directories established in it do not yet contain files, it consumes minimal resources in the file pool.

As a prerequisite of the BFS root object ownership and security (permission) requirements, as well as for a convenient base for conventional UNIX application usage, the installation process also includes some default system directory entries that establish the following POSIX user names, UIDs, primary group names, and GIDs:



If SYSENTRIES is in effect with the DIRPOSIX utility (see *z/VM: OpenExtensions User's Guide* for the utility description), the following POSIX groups are added, if groups with these names are not already defined:

Group name	GID
system	0
staff	1
bin	2
sys	3
adm	4
mail	6
security	7
nobody	4294967294

In addition, the following POSIX users are added, if they are not already defined:

Userid	UID	Primary group
root	0	system
daemon	1	staff
bin	2	bin
sys	3	sys
adm	4	adm
nobody	4294967294	nobody
default	4924967295	DEFAULT

The POSIX user names are lower case derivations of the upper case userids that are provided by the USER or IDENTITY directory statement. You should avoid conflicts between these default names and IDs and any userid, UID, primary group names, and GID assignments you specifically assign in your system. These names and identifiers are also consistent with the operation of the DIRPOSIX Utility. In addition to these default user names, there are also three default file spaces in the VMSYSU file pool.

- TMP
- VAR
- ETC

You should also take care that these names do not conflict with SFS enrolled users names or other BFS file space names in your system. See *z/VM: OpenExtensions User's Guide* for more information on the DIRPOSIX Utility. See *z/VM: CP Planning and Administration* for more information on the system directory control statements.

If you have reason to change these defaults, you can delete and replace the corresponding system directory control statements, then change the ownership of the BFS root objects with the OPENVM OWNER command. If you have reason to change the BFS root default objects, you can delete them and replace them according to your specific requirements.

A BFS root is necessary to resolve a BFS path name. Although this default root in the VMSYS file pool is built during the z/VM installation process, the BFS root is not really known for a particular BFS client until a mount is executed that establishes the BFS root. This mount can take place in several ways, in the order of precedence:

- An explicit OPENVM MOUNT command (default START option)
- An explicit OPENVM MOUNT command (NOSTART option)



- A statement in the client virtual machine PROFILE EXEC for the OPENVM MOUNT command (NOSTART option)
- A statement in the system profile for CMS for the OPENVM MOUNT command (NOSTART option) (see *z/VM: CMS Planning and Administration*).
- An FSROOT specification in the client virtual machine z/VM system directory (see the "POSIXINFO" directory control statement in *z/VM: CP Planning and Administration*).

Whatever method is used to mount the BFS root, you should use the following to specify the VMSYS default root:

```
/. ./VMBFS:VMSYS:ROOT/
```

`/. ./VMBFS:VMSYS:ROOT/` is also used when you supply a fully-qualified path name. Implicit connections to the VMSERVS file pool server are used to initially resolve the BFS root, such as the initial mount of the root described above.

By now it should be apparent that establishing the BFS root in VMSYS does not restrict BFS usage in any way. Because you can mount other BFS file spaces using as mount points directories or external links in this BFS root file space as mount points, and because those mounted file spaces can reside in any local or remote file pool, this root is nothing more than a convenient anchor from which users of the local system can build BFS addressable paths to objects in multiple local or remote file pools. See the section, "Establishing Clients in BFS File Spaces for more information" for examples of using the BFS root to mount additional file spaces. The BFS root in VMSYS is also a recommended repository for the OpenExtensions Shell and Utilities on each local system.

The BFS root directory is installed with the following subdirectories and other BFS objects:

- `usr`
- `usr/bin`
- `usr/include`
- `usr/include/sys`
- `usr/include/sys/time.h` (CMSDATA external link to SYS\_TIME H \*)
- `usr/lib`
- `usr/lib/nls`
- `usr/lib/nls/msg`
- `usr/lib/nls/msg/En_US.IBM-1047`
- `usr/lib/nls/msg/C.IBM-1047`
- `usr/lpp`
- `usr/pub`
- `tmp` (MOUNT external link `/. ./VMBFS:VMSYSU:TMP`)
- `var` (MOUNT external link `/. ./VMBFS:VMSYSU:VAR`)
- `var/spool`
- `var/spool/mail`
- `etc` (MOUNT external link `/. ./VMBFS:VMSYSU:ETC`)
- `dev`
- `dev/tty` (special file)
- `dev/null` (special file)
- `opt`
- `home`
- `lib` (symbolic link to `usr/lib`)
- `u` (symbolic link to `home`)

- `usr/mail` (symbolic link to `var/spool/mail`)
- `usr/spool` (symbolic link to `var/spool`)
- `usr/lib/nls/msg/En_US`. (symbolic link to `usr/lib/nls/msg/EN_US`. IBM-1047)
- `usr/lib/nls/msg/C`. (symbolic link to `usr/lib/nls/msg/C`. IBM-1047)

Note: `/tmp`, `/var` and `/etc` are external links to other file spaces that are automatically mounted upon first reference to the `tmp` or `var` path name components. If you elect to install the IBM OpenExtensions Shell and Utilities, it is loaded by default into the `usr/bin` directory of the BFS root. This means users of this feature also implicitly connect to the VMSEVS file pool server when executing OpenExtensions shell commands.

Although you are not bound to use the default BFS root installed in the VMSYS file pool, it is highly recommended you do so. You can build another BFS root in another file pool. Then construct your BFS root mount to use this alternate root in place of the default in VMSYS. You should do this only for a special purpose such as to temporarily override a version of the OpenExtensions Shell and Utilities installed in the default root.

When you installed z/VM, the BFSROOT EXEC created the BFS root file system (file space) in the VMSYS file pool. It also created two other file systems (file spaces) in the VMSYSU file pool and linked these additional file systems to the root file system in VMSYS with mount external links.

## BFS Root File Space in a Non-default File Pool

If you need to override these default file systems (file spaces), by building and mounting alternates for them, you can construct a modified version of the control file named BFS LOADBFS found on the MAINT 193 minidisk. Then locate this new version of BFS LOADBFS in the CMS search order such that it will be found before the one on the 193 disk. The control file controls where a BFS root structure is loaded by the BFSROOT EXEC. Your modification of the control file would substitute different file pools for VMSYS and or VMSYSU. Then when you run the BFSROOT EXEC, it will create another copy of the default file system root and associated parts in a file pool of your choice. Having done this, you can choose, through the root mount procedure described above, which root to use dynamically. This can be different for individual users or user groups. This would allow testing on different file pools, for example, with different roots.

If you take this alternative, you are building an alternate BFS root file system which does not contain objects that may have been created after the initial installation or migration. For example, if you installed the OpenExtensions Shell and Utilities, it would be stored in the original BFS root structure, but not in the alternate BFS root structure. Therefore, you would also need to install the OpenExtensions Shell and Utilities in the alternate BFS root file system. The same is true for any other objects added to the original BFS root file system.

**Note:** The LOADBFS command used to install certain utilities must be run on the VM system where the file pool resides. LOADBFS cannot be used to install data into a file pool on a remote VM system. The BFS limits the amount of data that can be sent to a remote file pool and LOADBFS sends more data than that limit.

## VMSYSU

---

VMSYSU is a file pool managed by the VMSEVSU server machine and is used as a production file pool to hold users' repository data. The benefits of VMSYSU are:

- Significant DASD space savings over minidisks are possible because file pools make more efficient use of DASD space for SFS and BFS repository data.
- Ability of users and applications to use SFS or BFS after completion of the z/VM installation. SFS and BFS offers the user these benefits:
  - Organization of files into hierarchical directories
  - Files and directories can be read- and write-shared among many users
  - Read or write permission can be controlled at the directory and file levels.

## VMSYSU Extensions to the BFS Root

The installation process establishes a BFS file space in VMSYSU named TMP. This file space is used implicitly by BFS functions for temporary intermediate storage purposes. This file space is found and mounted through normal BFS path resolution and a Mount External Link (MEL) in the VMSYS file pool. The /tmp object in the BFS ROOT of the VMSYS file pool ROOT file space is actually an external link (mount type) that causes the root directory of the TMP file space in VMSYSU to be mounted whenever there is a reference to /tmp in the VMSYS ROOT file space.

Similarly, two more file spaces, VAR and ETC, are also created in the installation process. VAR is associated with a /var external link (MEL) object and can be used for dynamic variable storage by various BFS functions. ETC contains the configuration files for the OpenExtensions Shell and Utilities.

For more information about SFS or BFS read the entire [Chapter 1, “File Pool Administration Overview,”](#) on [page 3](#).

## VMSYSR

---

Whether or not file pools elect to participate in CRR, there is an exchange of log names with the CRR Recovery Server when the first SFS request arrives at the file pool server. If no CRR Recovery Server exists (this is called "limp mode"), there is still an attempt to do this exchange of log names. Being unable to complete the exchange is tolerated and does not cause a file pool error. However, not being able to get past this initial exchange can cause repeated attempts to exchange with subsequent SFS requests. As long as there is no need to use CRR services, there is no functional affect on SFS operations, but limp mode causes a serious performance degradation and therefore should be avoided. This means you should install a CRR recovery server even if you are not using CRR services.

For BFS, which does not use CRR services directly, the installation of a CRR recovery server is not as important. However, because the file pool administrator occasionally uses SFS connections to the file pool server (occurs implicitly with certain administrative functions), the absence of a CRR recovery server (limp mode) can affect the performance of these operations that use SFS connections. Examples of these operations are ENROLL USER, MODIFY USER, QUERY FILEPOOL functions, and FILEPOOL commands (for example, FILEPOOL CONTROL BACKUP, FILEPOOL LIST BACKUP). IBM recommends you retain the VMSYSR file pool to avoid this performance degradation. (See [“Servers and File Pools”](#) on [page 13](#) for instructions.)

VMSYSR is a CRR file pool managed by the VMSEVR server machine, which is the CRR recovery server. The benefits of VMSYSR are:

- Improves SFS performance for both local and remote file pools. (A remote file pool is a file pool not on this system, but is referenced by programs on this system.) Without the CRR recovery server, there is significant SFS performance degradation.
- Allows applications to update multiple protected resources with integrity (two-phase commit processing), which is part of CRR's coordination function. Without the CRR recovery server, attempts to commit work that involves multiple updated resources will be unsuccessful and protected conversations are not permitted.
- There may be licensed programs you will be installing that participate in CRR. See the appropriate licensed program's planning documentation.

For more information about CRR, see [Chapter 18, “CRR Overview,”](#) on [page 287](#).



---

## Chapter 3. Post-installation Activities

This chapter describes the following IBM-supplied file pools:

- VMSYS (file pool managed by VMSEKVS server machine)
- VMPSFS (file pool managed by VMSEKVP server machine)
- VMSYSU (file pool managed by VMSEKVV server machine)
- VMSYSR (file pool managed by VMSEKVR server machine)

and the steps for tailoring these file pools and CMS to suit your installation's needs. Do not skip the tailoring steps.

---

### VMSYS, VMPSFS, VMSYSU, and VMSYSR

After the file pool servers and file pools are loaded, three data repository file pools and one CRR file pool exist. The repository file pools have a file pool ID of VMSYS, VMPSFS, and VMSYSU. The CRR file pool has a file pool ID of VMSYSR. You need to read and follow the instructions in this chapter only if you are the administrator for the three repository file pools, the CRR file pool, or both.

After installation, you have the option of using VMSYS, VMPSFS, VMSYSU, and VMSYSR. If you choose to use one or all of these file pools, follow the steps in this chapter for tailoring each file pool.

VMSYS is used for IBM system data such as programs that are to be shared among users. Examples include the Byte File System (BFS) root file space and OpenExtensions Shell and Utilities commands. It is not intended for other user files and directories. The VMSYS file pool is managed by the VMSEKVS server machine. If you plan to install DFSMS/VM, VMSYS is required. If you plan to use the OpenExtensions Shell and Utilities, or use BFS file spaces, VMSYS is highly recommended. Using an alternative BFS root system for storing these features involves complexity you should avoid unless you have a particular need for it.

VMPSFS is used to hold IBM product service data. During installation, you may have provided an alternative name for the VMPSFS file pool.

VMSYSU is intended to hold your users' data. The VMSYSU file pool is managed by the VMSEKVV server machine.

VMSYSR is minimal in size and does not contain user generated data, but is required for CRR recovery server processing. The VMSYSR file pool is managed by the VMSEKVR server machine.

---

### Characteristics of VMSYS

After VMSYS is installed, there are a few users established in the system directory, including at least one SFS user ID. That SFS user ID is MAINT. MAINT is enrolled with an SFS file space named "MAINT" in storage group 2, which is the only user storage group that is defined.

After install, the VMSYS file pool also contains a single BFS file space named ROOT in storage group 2. This file space can be mounted as the root file space in the byte file system hierarchies in the client machines. For more details on how to establish the root of a byte file system hierarchy, see [“BFS Root File Space in the VMSYS File Pool \(Default\)”](#) on page 31. If you plan to use the Byte File System and take advantage of this default BFS root (highly recommended), you should follow suggestions in the remainder of this section for adjusting the file pool startup parameters and storage space assignments for VMSYS to reflect your installation's usage of BFS. Even if you do not use the default root on VMSYS, and you want to use BFS, similar actions must be taken for the alternate BFS root that you choose.

Similarly, VMSYSU also has BFS file spaces named VAR and ETC in storage group 2. VAR can be used for dynamic variable storage by various BFS functions, and ETC contains the configuration files for the OpenExtensions Shell and Utilities.

**Note:** During z/VM installation, you can choose to load some components, features, or licensed products into the IBM-supplied file pools.

For example, if you loaded GCS to SFS, you have the following Group Control System (GCS) directories in VMPSFS:

```
VMPSFS:MAINT620.GCS.APPLYALT
VMPSFS:MAINT620.GCS.APPLYINT
VMPSFS:MAINT620.GCS.APPLYPROD
VMPSFS:MAINT620.GCS.DELTAPROD
VMPSFS:MAINT620.GCS.LOCALMOD
VMPSFS:MAINT620.GCS.OBJECT
VMPSFS:MAINT620.GCS.SAMPLE
```

If you moved features or licensed products from minidisk to SFS, each of them also has a set of directories in VMPSFS. Some may also have directories in VMSYS.

If DFSMS/VM is installed, these directories are created:

```
VMSYS:DFSMS.CONTROL
VMSYS:DFSMS.ACSEXITS
VMSYS:DFSMS.ACTIVEACSEXITS
VMSYS:DFSMS.ACTIVECONFIG
```

The first two directories (CONTROL and ACSEXITS) contain files that are tailorable for your system. The last two directories (ACTIVEACSEXITS and ACTIVECONFIG) are strictly for use by DFSMS/VM and should not be changed. For a listing of files that are placed in these directories, see [z/VM: DFSMS/VM Customization](#).

In addition to being an enrolled file pool user, the MAINT machine serves as the secondary user for all four IBM-supplied file pools. Thus, for example, when VMSERVS is running in disconnected mode, all messages that would have appeared on the VMSERVS console are routed to the MAINT machine console. From MAINT you can use the CP SEND command to respond to prompting messages or to enter server operator commands.

MAINT, MAINT<sub>vr</sub>m, and vVMTCP<sub>rm</sub> also serve as the initial administration machines. During installation, a DMSPARMS file is created in which these user IDs are identified as administration machines. Whenever server processing starts, these user IDs are automatically granted file pool administration authority.

z/VM is set up so VMSERVS is automatically logged on whenever z/VM itself is started. When VMSERVS is logged on, CMS is automatically initialized in it (IPL CMS). After CMS is initialized, it calls the PROFILE EXEC. The PROFILE EXEC of VMSERVS contains a FILESERV START command, which starts multiple user mode processing for the VMSYS file pool. So, whenever you start z/VM, VMSYS should be available for use.

Note that VMSYS is a *local only* file pool. That is, VMSERVS accepts requests only from users on your processor. It identifies itself to CP as a local resource.

To familiarize yourself with the structure of the VMSYS file pool, enter the QUERY FILEPOOL REPORT command. Because this command returns a lot of data, it is best to enter it while editing a file. When the XEDIT option is specified, the information returned by QUERY FILEPOOL REPORT is inserted into a CMS file. For example, edit a file (using XEDIT):

```
xedit temp data
```

Then enter the QUERY FILEPOOL REPORT command from the XEDIT command line. Specify the XEDIT option as follows:

```
====> query filepool report vmsys (xedit
```

The command results are automatically inserted into the file for you.

The QUERY FILEPOOL REPORT command lists:

- File Pool counter information
- CRR counter information (only if CRR Recovery Server)
- File pool compare information

- File pool information
- Currently defined minidisk information
- Agent information
- Repository file pool log information
- Catalog space information (optional)

For details, see [“QUERY FILEPOOL REPORT”](#) on page 592.

At some time when multiple user mode processing is stopped, you might also review the VMSYS POOLDEF file, which resides on the VMSEVS 191 minidisk. See [“FILESERV GENERATE”](#) on page 513 for descriptions of the meanings of the statements in a POOLDEF file. You might also review the z/VM system directory entries for the VMSEVS machine as well as the VMSEVS DMSPARMS file which contains the server startup parameters and resides on the VMSEVS 191 minidisk. See [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339 for more about startup parameters. For further insight into the definition of a file pool, see [Chapter 15, “Generating a File Pool and Server,”](#) on page 247.

## Characteristics of VMSYSU

---

After VMSYSU is installed, it has one storage group (storage group 2) for user data. The only SFS users enrolled in VMSYSU after installation is the user MAINT.

After install VMSYSU also has a BFS file space named TMP in storage group 2. This file space provides a default temporary space for BFS operations. It is addressed through a mount external link (MEL) in the VMSYS ROOT directory. (See [“BFS Root File Space in the VMSYS File Pool \(Default\)”](#) on page 31.) If you plan to use the Byte File System and take advantage of this default temporary space (highly recommended), you should follow suggestions in the remainder of this section for adjusting the file pool startup parameters and storage space assignments for VMSYSU to reflect your installations usage of BFS. Even if you do not use the default TMP in VMSYSU, similar actions must be taken for the alternate root you choose.

The MAINT user ID is the secondary user for the VMSEVU server machine. MAINT is also the initial administration machine for VMSEVU. During installation, a DMSPARMS file is created in which MAINT is identified as an administration machine. Whenever server processing starts, MAINT is automatically granted file pool administration authority.

Within MAINT's file space in VMSYSU after installation is at least one directory: VMSYSU:MAINT.SAMPLES. This directory contains the sample files used in exercises in the following manuals:

- [z/VM: CMS Primer](#)
- [z/VM: CMS User's Guide](#)

To make it easier to copy these files, during installation:

- SFS read authority is granted to PUBLIC on the sample files
- SFS read authority is granted to PUBLIC on the VMSYSU:MAINT.SAMPLES directory
- An SFS ENROLL PUBLIC command is entered for the VMSYSU file pool.

**Note:** You can also find the sample files on the MAINT 193 minidisk. Whenever a new file pool is generated, you should copy those sample files to the new file pool so users can follow the exercises in the documents. ([“Step 18: Copy the Sample Files to the New File Pool \(SFS only\)”](#) on page 273 provides instructions for copying the files.)

VMSEVU is automatically logged on when z/VM is started. When VMSEVU is logged on, CMS is automatically initialized in it (IPL CMS). After CMS is initialized, it calls the PROFILE EXEC. VMSEVU's PROFILE EXEC contains a FILESERV START command, so VMSYSU should be available in multiple user mode whenever z/VM itself is available.

Like VMSYS, VMSYSU is a *local only* file pool. It is identified to CP as a local resource for SFS and BFS functions.



You should familiarize yourself with VMSYSU in the same manner in which you studied VMSYS.

## Characteristics of VMPSFS

---

After VMPSFS is installed there are users enrolled for each component, licensed product and feature you selected to load to the file pool. All are enrolled in storage group 2, which is the only storage group that is defined.

The MAINT user ID is the secondary user for the VMSEVP server machine. MAINT and MAINT`vr`m serve as the administration machines for VMSEVP. During installation, a DMSPARMS file is created in which MAINT and MAINT`vr`m are identified as administration machines. Whenever server processing starts, MAINT and MAINT`vr`m are automatically granted file pool administration authority.

VMSEVP is automatically logged on when z/VM is started. When VMSEVP is logged on, CMS is automatically initialized in it (IPL 190). After CMS is initialized, it calls the PROFILE EXEC. VMSEVP's PROFILE EXEC contains a FILESERV START command, so VMPSFS should be available in multiple user mode whenever z/VM itself is available.

VMPSFS is a global file pool. It is identified to CP as a global resource for SFS and BFS functions. You should familiarize yourself with VMPSFS in the same manner in which you studied VMSYS.

## Characteristics of VMSYSR

---

No users are enrolled and no BFS file spaces are in in VMSYSR and no directories or files exist in VMSYSR.

The MAINT user ID is the secondary user for the VMSEVR server machine. MAINT and MAINT`vr`m are also the initial administration machines for VMSEVR. During installation, a DMSPARMS file is created in which these user IDs identified as an administration machine. Whenever server processing starts, these user IDs are automatically granted file pool administration authority.

VMSEVR is automatically logged on when z/VM is started. When VMSEVR is logged on, CMS is automatically initialized in it (IPL CMS). After CMS is initialized, it calls the PROFILE EXEC. VMSEVR's PROFILE EXEC contains a FILESERV START command, so the CRR recovery server should be available whenever z/VM itself is available.

VMSYSR is a *local only* file pool. It is identified to CP as a local resource for CRR administration functions.

You should familiarize yourself with VMSYSR in the same manner in which you studied VMSYS.

## Tailoring VMSYS, VMPSFS, VMSYSU, and VMSYSR

---

Some of the startup parameters of VMSYS, VMPSYS, VMSYSU, and VMSYSR were chosen to simplify the installation of z/VM and to limit the DASD space the file pools occupied. Prior to making these file pools available for general use, you should tailor these startup parameters to better suit your installation's needs. You should also define recovery procedures for the repository file pools, but not for the CRR file pool.

Except where indicated, the tailoring steps that follow apply to each file pool. You should follow these steps for each IBM-supplied file pool:

1. Log on or reconnect to *serverid*.

*serverid* could be VMSEVS, VMSEVP, VMSEVU, or VMSEVR.

This step is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

Because the server is set up for automatic starting, it is likely to be running in disconnected mode with the file pool available for multiple user access.

If that is the case, enter a STOP operator command:

```
stop
```



If you log on the server machine (it was not running in disconnected mode), you should avoid executing the PROFILE EXEC. Otherwise, the PROFILE EXEC calls a FILESERV START command, which is not desired at this time.

To avoid having the PROFILE EXEC process, enter the following command immediately after IPL CMS:

```
access (noprof
```

If you forget to enter the above command and the server successfully starts, stop the server by entering a STOP operator command:

```
stop
```

## 2. Create a SETUP EXEC, if none exists.

This step is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

When you log on a server machine but inhibit the processing of the PROFILE EXEC, the minidisks on which the FILESERV commands reside are not accessed. Therefore, it is useful to have an exec that accesses the proper minidisks but does not start multiple user mode processing.

To create such an exec, make a copy of the PROFILE EXEC:

```
copy profile exec a setup exec a
```

Then edit the SETUP EXEC, delete the line containing the FILESERV START command, and file it. Now run the SETUP EXEC to provide access to the proper minidisks:

```
setup
```

The instructions in this manual assume you have a SETUP EXEC available. You can, of course, name the exec anything you choose. Just remember to substitute your exec name whenever the instructions ask you to run the SETUP EXEC.

## 3. Edit the DMSPARMS file.

```
xedit serverid dmstateParams
```

where *serverid* could be VMSESRVS, VMSESRVP, VMSESRVU, or VMSESRVR.

This step is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

When editing the DMSPARMS file, use the following figures as examples of what to update. [Figure 6 on page 41](#) shows the contents of the DMSPARMS file for VMSESRVS, [Figure 7 on page 42](#) shows the contents of the DMSPARMS file for VMSESRVP, [Figure 8 on page 42](#) shows the contents of the DMSPARMS file for VMSESRVU, and [Figure 9 on page 42](#) shows the contents of the DMSPARMS file for VMSESRVR.

```
ADMIN MAINT MAINT $\nu$ rm  $\nu$ VMTCPrm...
NOBACKUP
SAVESEGID CMSFILES
FILEPOOLID VMSYS
USERS 100
```

*Figure 6. Startup Parameters for VMSESRVS (for System Data)*

```
ADMIN MAINT MAINTvrm...
NOBACKUP
SAVESEGID CMSFILES
LOCAL
FILEPOOLID VMPSFS
USERS 100
```

Figure 7. Startup Parameters for VMSEVP (for Product Service Data)

```
ADMIN MAINT MAINTvrm...
NOBACKUP
SAVESEGID CMSFILES
FILEPOOLID VMSYSU
USERS 100
NODFSMS
```

Figure 8. Startup Parameters for VMSEVU (for User Data)

```
ADMIN MAINT MAINTvrm...
NOBACKUP
SAVESEGID CMSFILES
FILEPOOLID VMSYSR
CRR
LUNAME systemname
USERS 100
```

Figure 9. Startup Parameters for VMSEVR (for CRR Recovery Server)

Change the values as follows:

### ADMIN

This startup parameter is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

This startup parameter defines the administration machines. If you want some other virtual machine to have administration authority as well, add another ADMIN startup parameter to the file. Simply add a new record (only one startup parameter is allowed per record). You can also add the user ID of that virtual machine to the ADMIN MAINT record. Multiple user IDs are allowed per ADMIN parameter.

**Note:** If DFSMS/VM is to manage the file pool, additional ADMIN parameters will need to be added. (See [z/VM: DFSMS/VM Customization](#) for more information).

For example, if you want to define user ID PETE as an administrator, you can add another ADMIN parameter or you can add PETE to the existing ADMIN parameter. For example,

```
ADMIN PETE
```

or

```
ADMIN MAINT PETE
```

For IBM supplied file pools, do not remove any of the user IDs that are included on the ADMIN statements in the default file.

### NOBACKUP

This startup parameter is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

This startup parameter indicates the file pool control data is not going to be backed up.

For most repository file pools, this is not a recommended approach and you should change NOBACKUP to BACKUP. The file pool control data (and perhaps the entire file pool) is subject to loss should errors occur.

For dedicated CRR file pools or dedicated FIFO file pools, it is recommended you do not back up the control data and therefore you should keep the NOBACKUP startup parameter.

NOBACKUP is specified in the DMSPARMS file so decisions regarding the recovery procedures for the file pool can be deferred until after z/VM is installed. Now is the time to define recovery procedures for your file pool.

The instructions in this document assume you are backing up the repositories (but, not the CRR file pool) control data, because control data backups allow recovery to approximate the point in time at which a failure occurs.

For this reason, it is recommended that for repository file pools (for example, VMSERVS, VMSERVP, and VMSERVU) you change the NOBACKUP startup parameter to:

```
BACKUP
```

Later steps in this tailoring procedure complete the preparation of your file pool for control data backups.

### **SAVESEGID CMSFILES**

This startup parameter is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

This startup parameter indicates the server uses code that has been loaded into a physical saved segment. A *physical saved segment*, for our purposes, can be thought of as a copy of executable code stored on DASD that can be efficiently shared among users. CMS uses a saved segment for the server code because it is more efficient than loading the server code from MAINT's 193 minidisk. The default physical saved segment for the server code is CMSFILES. The CMSFILES segment is pre-built on the z/VM installed system you loaded from the z/VM System DDR image.

It is also possible for you to create another saved segment containing the server code. If you change the name from CMSFILES, you must update the SAVESEGID startup parameter with the new server code segment name. If, for example, you create a saved segment named SFSSERV which contains the server code, you would specify the following startup parameter:

```
SAVESEGID SFSSERV
```

For more information, see [SAVESEGID](#) and [NOSAVESEGID](#) startup parameters.

SAVESEGID is specified in the DMSPARMS file because using a physical saved segment is efficient, especially if multiple servers are used on your processor. You should change to NOSAVESEGID only if the physical saved segment was deleted.

If NOSAVESEGID is specified, CMS uses its usual search order through accessed file modes to find the server code and load it. (The server code resides on MAINT's 193 minidisk.)

### **FILEPOOLID *filepoolid***

This startup parameter is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

This startup parameter specifies the file pool ID. For the three repository file pool servers, it is VMSYS, VMPSFS, and VMSYSU. For the CRR recovery server, it is VMSYSR.

For VMSYS, you should not change *filepoolid*.

During installation, you may have supplied an alternate name for the VMPSFS file pool. If so, VMPSFS was defined as a nickname in the ICOMDIR NAMES file

### **CRR**

This startup parameter is applicable to VMSYSR, but is not applicable to VMSYS, VMPSFS, and VMSYSU.

This startup parameter specifies the server as a CRR recovery server.

You should not change CRR.

### **LUNAME *luname***

This startup parameter is applicable to VMSYSR, but is not applicable to VMSYS, VMPSFS, and VMSYSU.

This startup parameter specifies the fully qualified LU name of this CRR recovery server. The CRR recovery server automatically issues a \*IDENT on the LU name to generate a reserved Transaction Program Name (TPN) used for communications between the CRR recovery server and a participating resource manager (for example, an SFS file pool server). For more information about the four TPNs that are reserved for the CRR recovery server, see the CRR startup parameter documentation in [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.

You cannot generate a CRR recovery server without the LUNAME startup parameter. During z/VM installation the LUNAME startup parameter in VMSEVR DMSPARMS is updated to an LU name generated from your system name and it looks like:

```
LUNAME systemname
```

If you have to change the LU name value, first see [“CRR Log Name Table Management”](#) on page 321 for steps to take when changing the CRR recovery server TPN. Then replace *luname* with your new fully qualified LU name for this CRR recovery server. The rules for determining LU names are in the LUNAME startup parameter documentation in [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.

### **USERS 100**

This startup parameter is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

For file pool repository servers, for example VMSYSU, change this value to represent your best estimate of the number of users expected during peak system activity. A user is someone who accesses files or directories managed by the file pool repository server.

If you expect to use BFS and the default BFS root in VMSYS, you can expect BFS clients to be users of VMSYS for purposes of mounting the BFS root file space. If you have installed the IBM OpenExtensions Shell and Utilities, users of this feature are also considered VMSYS users when this feature is loaded in the BFS root structure. By default, the BFS tmp, var, and etc directories that are associated with the BFS root are located in the VMSYSU file pool. Therefore, BFS users are also users by default of VMSYSU. When establishing a value for the USERS startup parameter for VMSYS and VMSYSU, you should include these implicit BFS users cases. (The value should be the same as the USERS value used in calculating the [MAXCONN](#) value for the server machine, see [Statement 3](#).)

For CRR recovery servers, 100 is the number of logged-on CRR users that you anticipate exploiting CRR by doing sync point processing (two-phase commit).

For CRR recovery servers, the default of 100 should be sufficient for most installations.

Choose this value carefully. Server processing tries to optimize its operation based on this value.

### **NODFSMS**

This startup parameter is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

The DFSMS startup parameter is only used if DFSMS/VM is to manage the file pool. DFSMS is not applicable to VMSYSR. (NODFSMS is the default parameter.) This has implications for the file pool you should be aware of, including:

- Backup
- Moving user file space
- Enrolling users

The use of DFSMS/VM to manage a file pool should be part of an overall file pool storage management strategy. See [z/VM: DFSMS/VM Planning Guide](#) and [z/VM: DFSMS/VM Storage Administration](#) for more information about DFSMS/VM and how it can help you manage file pools repositories.

## 4. Define a recovery procedure and do an initial file pool repository control data backup.

This step is applicable to VMSYS, VMPSFS, and VMSYSU, but is not applicable to VMSYSR.

IBM does not recommend doing control data backups for CRR file pools.

It is essential you define backup and restore procedures for your file pool. If you do not back up your file pools, some data losses could result in the loss of the entire file pool.

Earlier in these instructions, you changed the NOBACKUP startup parameter to BACKUP. This changes server processing such that it will permit you to back up its control data. To complete preparations for backing up the control data portion of the file pool, you need to do the following:

- Specify the control data backup file.

Prior to making a control data backup, you must tell the server where you would like that backup to be placed. The control data can be backed up to either a disk file or a tape file. Guidance for choosing disk or tape is in [“Backing Up The Control Data”](#) on page 104.

To specify the BACKUP file, use the FILESERV DEFBACKUP command. If, for example, you wanted to direct future control data backups to the tape device at virtual address 181, you would enter:

```
fileserv defbackup tape 181
```

If, on the other hand, you wanted to direct the backups to a CMS file named BACKUP DATA B, you would enter:

```
fileserv defbackup disk backup data b
```

For a description of the command, see [“FILESERV BACKUP”](#) on page 500.

- Back up the control data.

When the BACKUP startup parameter is specified and a control data backup file is defined, your server is properly configured for backing up the control data. Now use the FILESERV BACKUP command to back up the control data:

```
fileserv backup
```

See [“Backing Up The Control Data”](#) on page 104 for more detail.

- Adjust the size of the file pool logs.

Because server processing uses its file pool log minidisks differently when the BACKUP startup parameter is in effect, it may be necessary for you to adjust the size of the log minidisks. You can defer adjusting the size of the logs until the file pool is in use and you can determine the rate at which the logs are filled. Guidance for tuning the size of the log minidisks is in [“File Pool Log Size When File Pool Backup Is Used”](#) on page 252.

## 5. Enroll the first set of file pool users.

This step is applicable to VMSYS, VMPSFS, and VMSYSU, and is not applicable to VMSYSR.

CRR file pools do not contain user data.

After you have backed up the control data and have recovery procedures in place, you should enroll your first set of file pool users. Instructions for doing so are in [“Enrolling SFS Users and Creating BFS File Spaces”](#) on page 71.

**Note:** Because the VMSYS and VMPSFS file pools are not intended for user data, your enrollments, if you enroll anyone at all, should be limited.

## 6. Consider enrolling PUBLIC.

This step is applicable to VMSYSU, but is not applicable to VMSYS and VMSYSR. VMSYS already has PUBLIC enrolled, and VMSYSR does not have user data and therefore does not have enrolled users.

At this time you might consider whether you want to enroll PUBLIC in the file pool. This allows the broadest range of users to *connect* to the server machine. *Connect* simply means the server will allow

the user to communicate with it. For a description of enrolling PUBLIC, and the security implications of doing so, see [“Enrolling PUBLIC” on page 92](#).

### 7. Do a user data backup.

This step is applicable to VMSYS, VMPSFS, and VMSYSU, but is not applicable to VMSYSR.

IBM does not recommend doing user data backups for CRR file pools, because they contain no user data to be backed up.

The control data is only a part of the file pool. To protect the entire file pool, you must also back up the user data. The user data is any data in storage groups 2 through 32767.

To back up the user data, you must back up each storage group individually. After every storage group is backed up once, you do not need to back up all the storage groups in one session. Instead, you can back up one storage group one day, another the next, and so on. You do not have to back up storage groups that do not have users enrolled in them.

Now make an initial set of user storage group backups. Start the server in multiple user mode:

```
fileserv start
```

Then disconnect from the server machine:

```
#cp disconn
```

Next, log on a file pool administration machine. Remember MAINT is the initial administrator (unless you have removed ADMIN MAINT from the startup parameters). From this administration machine, you can enter a FILEPOOL UNLOAD GROUP command for each user storage group in the file pool. Note the default IBM-supplied file pools contain only one user storage group (storage group 2), so you need to enter only one FILEPOOL UNLOAD command for each file pool.

Prior to entering the FILEPOOL UNLOAD GROUP command, you need to enter a FILEDEF command to identify the backup file. Follow the instructions in [“Backing Up User Data” on page 111](#) to make your initial user storage group backup. When you have completed the backup, return to these instructions.

To avoid errors in your backup procedures, it is important you remember the differences between control data backups and user storage group backups. Those differences are explained in [Chapter 7, “Recovery Procedures,” on page 101](#). It is recommended you read that chapter and put backup procedures in place before using the file pools for production data.

### 8. Update z/VM System Directory Control statements.

This step is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

Using your local operating procedure, update the following z/VM system directory control statements for the IBM-supplied file pool servers (VMSERVS, VMSERVP, VMSERVU, and VMSERVR). See [z/VM: CP Planning and Administration](#) for a description of the z/VM system directory control statements.

- ACCOUNT statement

Change the account number and distribution code from the IBM-supplied value of ACCOUNT 1 *serverid*, where *serverid* is VMSERVS, VMSERVP, VMSERVU, or VMSERVR, to the appropriate values according to your installation's policies. The account number is a number to which a virtual machine may charge its costs. The distribution code may be used to designate where printed output is to go.

- MACHINE statement

Use XC mode if you want to exploit data spaces (SFS). For VMSERVS, VMSERVP, and VMSERVU, the IBM-supplied value is MACHINE XC. If you are using XA mode, you may want to change MACHINE XC to MACHINE XA. In this situation, CP allows an XC entry but automatically places you in XA mode and issues a console warning message at every IPL. Changing MACHINE XC to MACHINE XA avoids the warning message.

CRR recovery servers, FIFO servers, and BFS files and directories do not exploit data spaces; therefore IBM supplies VMSERVR with MACHINE ESA and this does not need to be changed.

- XCONFIG statements

The XCONFIG statements support XC mode and therefore should be used if you want to exploit data spaces. For VMSESRVS, VMSESRVP, and VMSESRVU, if you are using XA mode you may want to consider removing the IBM-supplied XCONFIG statements, but there is no harm in leaving them.

CRR recovery servers, FIFO servers, and BFS files and directories do not exploit data spaces; therefore IBM supplies VMSESRVR with no XCONFIG statements and this does not need to be changed.

- POSIXOPT statement

The VMSESRVS, VMSESRVP, and VMSESRVU servers should have a POSIXOPT statement to allow the servers to have CP change their POSIX security values in support of opening executable files. If you have migrated existing VMSESRVS, VMSESRVP, or VMSESRVU servers, and you plan to use them to support BFS clients, you should add this directory control statement:

```
POSIXOPT SETIDS ALLOW
```

It should come before any device statements.

- CONSOLE statement

Optionally change the secondary user ID. The MAINT user ID is defined as the secondary user ID for VMSESRVS, VMSESRVP, VMSESRVU, and VMSESRVR. When VMSESRVS, VMSESRVP, VMSESRVU, and VMSESRVR are running in disconnected mode, the secondary user receives all the server console messages. This is also known as Single Console Image Facility (SCIF). If you would prefer to use some other user ID, update the z/VM system directory entries (using your usual operating procedures). You need to change the CONSOLE control statement for the virtual machine. For example, suppose the CONSOLE statement currently is:

```
CONSOLE 009 3215 T MAINT
```

You would simply change MAINT to the user ID of your choice. Note that for convenience of operation and administration, the user ID you select to serve as a secondary user should also have file pool administration authority.

## 9. Optionally spool console output.

This step is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

When VMSESRVS, VMSESRVP, VMSESRVU, and VMSESRVR are running in disconnected mode, the secondary user receives all the server console messages. But, if the secondary user is not logged on, or is running in disconnected mode, then the server console messages are lost. To prevent losing server console messages, you might want to consider spooling the server's console to the user ID of the secondary user. Add the following to the server's PROFILE EXEC:

```
CP SPOOL CONSOLE START TO secondid
```

### ***secondid***

is the user ID of the server's secondary user.

When the console is closed by issuing the SPOOL CONSOLE CLOSE command, a reader file is sent to *secondid*. The reader file contains a copy of the server's console since the last time a SPOOL CONSOLE CLOSE command was issued. Now, you can view or print the reader file for a closer examination of the console output.

Consider automating the issuing of the SPOOL CONSOLE CLOSE command, by adding the following command to the PROFILE EXEC of the secondary user:

```
CP SEND serverid #CP SPOOL CONSOLE CLOSE
```

### ***serverid***

is the user ID of the server

Every time the secondary user ID is logged on and its PROFILE EXEC is run, the SPOOL CONSOLE command is issued. Then the server sends a reader file that contains a copy of all the server's console output since the last time the secondary user logged on.

### 10. Optionally increase size of file pool.

This step is applicable to VMSYS, VMPSFS, and VMSYSU but is not applicable to VMSYSR.

If you plan to install a licensed program on to the IBM-supplied file pools then you should determine if there is enough space in the file pool VMPSFS to hold the licensed program. See the appropriate licensed program's planning documentation to determine how much space is needed. If you need to increase the size of the file pool VMPSFS, see [Chapter 9, “Managing Storage,” on page 195](#) for more information.

Only a minimal initial minidisk space allocation is established at initial installation for storage group 2 (users) of VMSYSU file pool.

If you add SFS users to VMSYSU you will need to add minidisk storage allocations to accommodate them.

If you are using the Byte File System (BFS), you will need additional minidisks and file space block allocations. See [“Adding Minidisks in Multiple User Mode” on page 199](#) and [“Changing the Amount of Space Assigned to a File Space” on page 91](#) for more information on these topics.

- For temporary space in /tmp, you will need to increase the block allocation of the TMP file space in the VMSYSU file pool and also increase the minidisk space allotted to storage group 2 in VMSYSU in which the TMP file space resides. Similarly for space for objects in /var and /etc, you will need to increase the block allocation of the VAR and ETC file spaces in the VMSYSU file pool and also increase the minidisk space allotted to the storage group 2 in VMSYSU in which the VAR and ETC file spaces reside.
- If you use VMSYSU for BFS client file spaces, you will also have to increase the minidisk space in the associated VMSYSU storage groups. See [“Overview of Adding DASD Space to a File Pool” on page 195](#).

You can also move the BFS temporary and/or variable space from the installed default storage group 2 of VMSYSU to another storage group or file pool. To do this you must replace the mount external links (MEL) object in the root directory of the VMSYS file pool with a new mount external links that identifies other file spaces, and possibly another file pool. This is done with the OPENVM CREATE EXTLINK command. See [z/VM: OpenExtensions Commands Reference](#). See [Chapter 9, “Managing Storage,” on page 195](#) for more information.

You should not have to increase the size of VMSYSR because there are no enrolled users and you would not install licensed programs on to VMSYSR.

### 11. Optionally adjust the DASD layout.

This step is applicable to VMSYS, VMPSFS, VMSYSU, and VMSYSR.

DASD placement considerations are documented in two places in this manual. See [“Step 3: Determine Initial DASD Allocations” on page 250](#) for initial DASD placement considerations used during file pool generation. See [“Determine the New Allocations” on page 196](#) for additional considerations used when adding minidisks to the file pool.

### 12. Notify users.

This step is applicable to VMSYS, VMPSFS, and VMSYSU, but is not applicable to VMSYSR.

CRR file pools do not have file pool users.

After tailoring the file pool for your installation and putting backup procedures in place, you can notify users the file pool can be used for production data (VMSYSU), or as a system file pool (VMSYS and VMPSFS).



## Tailoring CMS

---

The CMS file pool cache defaults to 20 KB. The cache affects the performance of accesses to files. It also affects the performance of accessing BFS files when accessed by a file pool administrator that is connected to the file pool through SFS (using SFS file pool operations). This is a good choice for systems that have moderate paging rates. If your system has an especially high paging rate, you may want to consider setting this to a lower value. If your system has a low paging rate, performance is likely to benefit if you increase the file cache size.

To change the file pool cache size, you need to update the `BUFFSIZ` parameter in the `DEFNUC` macro, which is in `DMSNGP ASSEMBLE` (`DMSZNGP ASSEMBLE` for z/Architecture CMS). Then assemble the file and rebuild the CMS nucleus. See [z/VM: CMS Planning and Administration](#) for information about `DMSNGP`, `DMSZNGP`, and `DEFNUC`. See [z/VM: Service Guide](#) for instructions on rebuilding the CMS nucleus.



## Chapter 4. Migration Considerations

When you migrate your file pools from a previous release to a new release, there are several activities that must be performed on the file pools. These activities are discussed in *z/VM: Migration Guide* in the section which discusses converting your file pool servers. There is also additional function you may want your file pools to exploit. This chapter identifies the activities you will need to perform to exploit any or all of these functions.

For information about migrating your file pools from a new release back to a previous release, see the section on that topic in *z/VM: Migration Guide*.

### File Pool Startup Parameters

IBM supplies the VMSYS, VMPSFS, VMSYSU, and VMSYSR file pools. When you install these file pools, you might tailor them to meet your own needs as described in Chapter 3, “Post-installation Activities,” on page 37. Or, you might generate your own file pool as described in Chapter 15, “Generating a File Pool and Server,” on page 247. In either case, you should review the startup parameters (DMSPARMS file) for each server on your current release and determine if these startup parameters apply to the servers you will have on your new release. See Chapter 20, “File Pool Server Startup Parameters,” on page 339 for more information about the startup parameters.

### Backing Up the File Pool

Control data backup files created prior to VM/ESA® version 1 release 2.1 may be used for restore only on releases prior to VM/ESA version 1 release 2.1. Therefore, you have to use FILESERV BACKUP as part of the procedure to migrate to z/VM 7.2 from a VM/ESA version 1 release 2.0 or earlier system. This procedure is documented in *VM/ESA 2.4: Conversion Guide and Notebook*.

Storage group (user data) backup files created on your old release can be used on the new release. It is not necessary to immediately create new storage group backup files on the new release.

### Use of the Byte File System (BFS)

If you plan to use the Byte File System (BFS), you need to understand the BFS default root file space, and may need to make adjustments to file pool resources. Unless you have deliberately avoided installation of the default file pools, VMSYS and VMSYSU, the BFS root structures and objects are established therein. You should review Chapter 2, “Installation Planning for File Pools,” on page 31 and Chapter 3, “Post-installation Activities,” on page 37 in general for information on the default file pools relative to BFS root file spaces. In particular you should concentrate on the instructions in the BFS topics “Tailoring VMSYS, VMPSFS, VMSYSU, and VMSYSR” on page 40 (steps “3” on page 41, “8” on page 46, and “10” on page 48).

### LU Name of CRR Recovery Server (CRR Only)

The LU name for your CRR recovery server was supplied at one of these times:

- IBM supplied the LU name during installation of the IBM-supplied CRR recovery server VMSYSR
- You supplied the LU name while generating your own CRR recovery server; see Chapter 15, “Generating a File Pool and Server,” on page 247 for details.

At a later time, the characteristics of your system may change which would require you to change your CRR recovery server's LU name. For example, if you added VTAM® and AVS to your system, you would probably have to change the LU name.

See “CRR Log Name Table Management” on page 321 for steps to take when changing the CRR recovery server TPN. Then replace *luname* with your new fully qualified LU name for this CRR recovery server.

## Data Space Exploitation

For rules on creating LU names, see the LUNAME startup parameter description in [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.

For information about migrating file pool servers back to a previous release, see [z/VM: Migration Guide](#).

## Data Space Exploitation (SFS Only)

---

The ESA/XC and z/XC virtual machine architectures have a facility called data spaces (introduced in VM/ESA version 1 release 1.1). SFS can exploit data spaces to improve the performance of directory control directories. If you have directories that meet these criteria:

- Seldom updated
- Accessed in read-only mode
- Accessed by many users on the local system

then you should read [Chapter 14, “Using Data Spaces \(SFS Repository Servers Only\),”](#) on page 241 to determine how to exploit data spaces to improve the performance of these directories.

**Note:** If you want to exploit data spaces, you must make sure that all FBA user storage group minidisks are aligned on 4KB boundaries. See [“Restrictions for Using Data Spaces on FBA Minidisks”](#) on page 244 for details on aligning FBA minidisks on 4KB boundaries.

## File Space Storage Use Exits

---

IBM supplies the DMSSFSEX CSL routine the server calls and you can modify to handle these conditions:

- Storage group space exhausted
- User's file space limit exceeded

This is not applicable to storage groups that contain only BFS file spaces.

For more information describing how you can take advantage of these SFS storage use exits, see [“Activating the Storage Use Exits \(SFS and BFS\)”](#) on page 209.

## Catalog Reorganization

---

The file pool server will now more efficiently use catalog index blocks. You will not have to enter the FILESERV REORG command as often, which means the amount of time the file pool server is unavailable to the end user is reduced. Follow the steps in [Chapter 10, “Reorganizing the File Pool Repository Catalogs,”](#) on page 213 to determine if catalog reorganization is necessary.

## DFSMS/VM Exploitation

---

DFSMS/VM Function Level 221 automates storage management tasks for SFS and BFS files and moves CMS minidisks from one physical device to another. With Interactive Storage Management Facility (ISMF), an interface to perform storage management operations, DFSMS/VM lets you as the file pool administrator:

- Convert file pool storage to DFSMS-managed storage by assigning *management classes* to files and directories. Each management class tells DFSMS/VM how to treat its members in the course of its management of the file pool.
- Automatically manage files based on the criteria in each management class. This management may consist of deletion of files, automatic migration of files, or both.
- *Migrate* (or move) files from DFSMS-managed storage to DFSMS-owned storage by using the assigned *management class*. The files can be automatically recalled when referenced with the CMS SET RECALL command, while explicit recall is performed with the DFSMS RECALL command.

You must specify DFSMS in your DMSPARMS file to exploit DFSMS/VM.

See the DFSMS/VM books for more information about DFSMS/VM.

## What to Read Next

---

Next you should read [Chapter 5, “Operation,” on page 55](#). That chapter discusses the day-to-day operation of file pool servers.



## Chapter 5. Operation

This chapter contains the following procedures for operating server machines:

- Starting multiple user mode processing \*
- Monitoring server operation \*
- Stopping multiple user mode processing \*
- Changing the file pool ID.

The procedures followed by an asterisk (\*) are often-used procedures for any server machine. If you are new to the job of file pool administration, you may want to browse through these procedures.

This chapter applies to both repository (SFS and BFS) file pool servers, CRR recovery servers, and FIFO servers. The topic on "Monitoring Server Operation" has limited applicability to CRR recovery servers and FIFO servers.

**Note:** The subsection on [“Using QUERY FILEPOOL REPORT to Monitor Limits”](#) on page 58 may be interesting to a CRR administrator who is monitoring the status of one or more CRR recovery servers (see [“Set Up Remote User Machine Administration of CRR Recovery Server”](#) on page 326).

For a description of the possible combinations and types of file pool servers, see [Chapter 1, “File Pool Administration Overview,”](#) on page 3.

### Starting Multiple User Mode Processing

During z/VM installation, the VMSERVS, VMSERVP, VMSERVU, and VMSERVR machines are usually set up so multiple user mode processing is automatically started for them when z/VM itself is started on your processor. If z/VM fails and is restarted, multiple user mode processing for VMSERVS, VMSERVP, VMSERVU, and VMSERVR is also restarted.

The only time you should need to start these servers is after you have explicitly stopped them (perhaps for dedicated maintenance mode activity), or if they have stopped because of a problem, or if someone at your installation has deleted the automatic logon for the server.

The instructions in [Chapter 15, “Generating a File Pool and Server,”](#) on page 247 for defining other servers include setting up the server for automatic starting. If the instructions were followed, those servers will also be started automatically when z/VM is started. As with VMSERVS, VMSERVP, VMSERVU, and VMSERVR, the servers should need to be restarted only in special circumstances.

If you are already logged on to the server machine, and want to start multiple user mode processing, enter the FILESERV START command:

```
fileserv start
```

If you are not logged on to the server machine, do the following to start multiple user mode processing:

1. Log on the server machine that owns the file pool minidisks. CMS is automatically loaded in the server machine.
2. If you see the VM READ status notice after CMS is loaded, press ENTER. The PROFILE EXEC issues a FILESERV START command to start multiple user mode operation.

You see a series of informational messages ending with these messages:

```
DMS3060I Initialization complete
DMS3045I Ready for operator communication
```

When you see message DMS3060I you know the server is running in multiple user mode. If, instead, you receive an error message, fix the cause of the error and enter the FILESERV START command:

## Monitoring Server Operation

```
fileserv start
```

If you are not sure how to fix the problem, see [z/VM: CMS and REXX/VM Messages and Codes](#).

The amount of time it takes the server to start varies depending on the amount of *restart recovery* the server must do. *Restart recovery* is a process during which the server looks at the log minidisks to determine whether there is any incomplete work that needs to be resolved. If there is, the server resolves the incomplete work as appropriate, then immediately starts multiple user mode processing.

Incomplete work may be recorded in the logs if the server did not end properly the last time it ran (perhaps the system failed) or if you entered a STOP IMMEDIATE operator command (see “[Stopping Server Processing](#)” on page 65). The file pool repository logs would record logical units of work that need to be complete and the CRR logs would record coordinated transactions (sync points) that need resynchronization processing.

For more information, see “[FILESERV START](#)” on page 537.

3. Now disconnect from the server by entering:

```
#cp disconn
```

4. Finally, log on the secondary user machine. The secondary user machine typically has file pool administration authority. Ensure CMS is running in that virtual machine. (The secondary user and initial administration machine for VMSERVS, VMSERVP, VMSERVU, and VMSERVR is MAINT.)

It is important for either the primary or secondary user to be logged on during multiple user operation. The server displays few messages during its operation, but those that are displayed should not be ignored. They are often early warnings of conditions that will require attention.

## Monitoring Server Operation

---

After a server is started, it requires little monitoring. Server machines detect conditions that require attention and give you early warnings in the form of server operator messages. You can take a *leave it alone* approach to operating a server, reacting to server messages as required.

Typically, you would be doing other work on the virtual machine serving as the secondary user when a message from a file pool server machine appears. Messages from a file pool server machine will appear in this form:

```
VMSERVU : DMS3202W Storage group 3 is short on storage
```

The first part of the message tells where the message is from. In this case, it is from the system file pool server machine VMSERVU. The text following the colon (:) is the actual message that would have appeared on the VMSERVU console.

If the message asks for a response, use the CP SEND command to send a reply. Suppose you receive a message that asks for you to enter a **1** for **yes** or **0** for **no**. You want to respond with **1**, so you enter:

```
#cp send vmservu 1
```

The `#cp` portion of the command gets CP's attention. `send` is the command. After `#cp send`, type the user ID of the file pool server machine, which in this example is VMSERVU, followed by the response.

In addition to responding to messages, you may want to enter file pool server machine operator commands. These are also entered using the CP SEND command. (Operator commands are covered in other sections as needed.)

## Automatic Backups

This section applies only to repository file pool servers. It does not apply to dedicated CRR recovery servers or dedicated FIFO servers.



If the BACKUP startup parameter is in effect, the server may occasionally start backing up the control data during multiple user mode processing. The server starts an automatic backup whenever its file pool log minidisks are 80% full. When the log is 78% full, the server displays a message warning of the impending backup. The message is repeated again when the log is 79% full. By backing up the control data, the server frees log space. If the server allowed the log to fill, it would not be able to continue processing.

While automatic backups allow the server to continue processing, they have the undesirable quality of being unscheduled. Automatic backups tend to start during times of peak activity, because that is when the logs are most likely to fill.

Furthermore, if you are away from the console when an automatic backup is started and you are backing up to tape, the server can stop all processing in certain cases. For example, messages may be issued if something happens during Open, (such as correct tape is not mounted), or if you run out of space on the tape.

If you are backing up to DASD (to another file pool or to minidisk), you need not be concerned with tape problems interfering with server processing. However, backing up to DASD or tape may cause increased response time for other users of that file pool if the backup occurs during peak usage hours. This disruption to other users can be significantly reduced if you choose to back up to another file pool.

To avoid automatic backups, increase the size of your log minidisks. Your goal should be to tune the size of the logs such that you can schedule all your backups without wasting DASD space by defining excessively large log minidisks. To change the size of the log minidisks, see [“Replacing Both File Pool Log Minidisks” on page 133](#). For guidance on selecting initial sizes, see [log minidisk](#).

If you ever suspect the file pool control data is damaged (perhaps the POOLDEF file was damaged and refers to incorrect minidisks), avoid making a control data backup. Enter a STOP IMMEDIATE operator command to end server processing. (See [“Stopping Server Processing” on page 65](#).) Then correct the problem and restart multiple user mode processing.

If you allow a backup to be made in this case, it may not be valid for restoring the control data. Because only the most current backup can be used for restoring the control data, backing up incorrect control data means, essentially, that your control data is not protected.

## Forcing a Server User

This section applies only to repository file pool servers. It does not apply to dedicated CRR recovery servers or dedicated FIFO servers.

While the server is running in multiple user mode, you may occasionally find it necessary to force an active user to end his or her use of the server. What typically happens is the user starts a long-running application and leaves the terminal unattended. The application locks files or directories that other users need to access. Eventually, the users call you to complain. You would then use a QUERY LOCK or a QUERY FILEPOOL CONFLICT command to determine who is holding the resource and then try to contact that user (if the complaining user has not already done so).

If the user holding the resources is doing so for a valid reason, you would let that user continue using the server. Otherwise, you should consider forcing the user so that others may use the locked files.

To force a server user, enter the server FORCE operator command. For a description of the command, see [“FORCE” on page 539](#). The QUERY LOCK command is described in [z/VM: CMS Commands and Utilities Reference](#). For a description of the command, see [“QUERY FILEPOOL CONFLICT” on page 563](#).

## Forcing Prepared Work

This section applies only to repository file pool servers. It does not apply to dedicated CRR recovery servers or dedicated FIFO servers.

There may be times when you have to force *prepared* work. For example, let's assume there is an application using CRR's two-phase commit to perform coordinated updates in multiple file pools. If the two-phase commit is able to complete the update in one of the file pools but an error occurs so the two-phase commit completes only the first phase in the other file pool, then the logical unit of work waits

for CRR resynchronization. The logical unit of work that is waiting for resynchronization is called *prepared*, which means it is prepared to commit the update but an error prevented the update. If there will be a long delay before CRR resynchronizes the prepared work, you may receive complaints from other users that are prevented from accessing files or directories involved in the work unit in that same file pool. It may be necessary to manually synchronize the prepared work and not wait for resynchronization. For detailed steps on forcing prepared work, see [Chapter 17, “Participation in CRR \(SFS only\),” on page 279](#).

## Periodic Catalog Maintenance

This section applies only to repository file pool servers. It does not apply to dedicated CRR recovery servers or dedicated FIFO servers.

Every few months check the file pool catalogs for fragmentation. When fragmentation is detected, you should reorganize the file pool catalogs. When the catalogs are properly organized, the server can access the file pool efficiently. See [Chapter 10, “Reorganizing the File Pool Repository Catalogs,” on page 213](#) for complete instructions on how to determine if reorganization is necessary.

## Limit Monitoring

This section applies to all file pool servers. However, SFS monitoring is more important than CRR monitoring. The only things that might be of interest to monitor in CRR recovery servers are:

- MAXCONN value
- USERS startup parameter
- CRR log size
- Virtual storage (possibly)

You should periodically (perhaps weekly or monthly) monitor various file pool (not CRR) limits associated with your file pools and server machines. These limits, which include items like the anticipated number of logged on users and the amount of available physical storage, are established when you define the file pool and server. As these limits are reached, the server, when possible, continues regular operation. In some cases, it takes *corrective* action that allows it to continue operation. For example, when all the physical file pool repository DASD space allocated to a user storage group is used, the server continues operation. Users are able to read files, but are not able to write files until additional space becomes available.

When some limits are reached, the server cannot take corrective action and stops. Such is the case, for instance in file pool repositories when the catalog storage group runs out of physical DASD storage.

When the server detects it is reaching a limit, it displays a warning message. The warning message is intended to allow you enough time to shut down the server and increase the limit. But, for some kinds of limits, the server cannot detect that it is approaching the limit. In these cases, the server displays an error message when the limit is reached.

It is important to monitor limits periodically so you can plan for the activity that must be done before the limit is reached. If you never check the limits, you may encounter problems such as having to add a minidisk to a file pool repository storage group but not having available DASD space to add.

The following sections list the limits, how to monitor them, and the effects of reaching them. The sections refer to *logical* and *physical* space. *Logical space* is potential space the file pool can address. *Physical space* is real DASD space. A file pool repository might, for instance, be able to address 10GB of space (which is logical space), but may only have 5GB of DASD storage (physical space) currently allocated to it.

## Using QUERY FILEPOOL REPORT to Monitor Limits

This section applies to all file pool servers.

To monitor many of the following limits, you use the QUERY FILEPOOL REPORT command. When the CATALOG option is specified, this CMS command must be issued from an administration machine. (Remember the MAINT machine is the default administration machine as well as the secondary user for VMSERVS, VMSERVP, and VMSERVU.)

Because the QUERY FILEPOOL REPORT command returns many lines of information, it is best to enter it using the XEDIT option. When the XEDIT option is specified, the information returned by QUERY FILEPOOL REPORT is inserted into a CMS file. To use the XEDIT option, you must already be editing a file using XEDIT. For example, suppose you want the results of a QUERY FILEPOOL REPORT command to be returned in a new file named 88JUNE STATUS. You would first edit the file:

```
xedit 88june status
```

Then enter the QUERY FILEPOOL REPORT command from the XEDIT command line:

```
=====> cms query filepool report (xedit
```

The data inserted into the file will show various kinds of information about the file pool followed by about 100 lines of counter information. The counter information tells the number of times certain events have occurred since the server was started. Before the counter information you see information about the maximums established for the file pool and about physical storage use. For more information, see “QUERY FILEPOOL REPORT” on page 592.

## Monitoring the USERS Startup Parameter

This section applies to all file pool servers.

When server processing starts in multiple user mode, it uses the USERS value to configure itself to handle the work that the specified number of users might cause. When the USERS startup parameter is set, the number of agents created should be sufficient such that a user making a server request never has to wait for an agent. If there is an insufficient number of agents, users are queued until one is available—this may result in poor server response time. If there are too many, virtual storage in the server machine is wasted.

To monitor this limit, enter a QUERY FILEPOOL AGENT command and look at the Active Agents Highest Value in the *Agent Information*. It is best to check this limit late in the day after peak work loads have occurred. If you check the Active Agents Highest Value and it is close to or equal to the Total Number of Agents the server has created, you should consider increasing the USERS value in the DMSPARMS file. (For more information, see USERS parameter.)

Suppose, for example, your USERS value is set to 129. At the end of the day you enter QUERY FILEPOOL AGENT and find the Active Agents Highest Value is 19 and Total Number of Agents is 20. This indicates the server is operating near peak capacity for its configuration. If the Active Agents Highest Value is equal to the Total Number of Agents, you know that at some time during server operation the server was operating at peak capacity and may have, in fact, been overloaded. Because the Active Agents Highest Value never exceeds the Total Number of Agents, an equality of the two values indicates an overload. Further support for this conclusion would be complaints from users that their applications are failing with reason codes of 97400 (or they are receiving message DMS1151E), even though the server is available and can process other commands.

The amount by which you should increase USERS is subjective. If file pool use has in the past had relatively slow growth, a nominal increase might be sufficient. If your file pool has had rapid growth, a proportionally higher increase is warranted.

### ***What Happens When the Limit is Reached***

When the actual user workload increases beyond what the server has configured itself to handle, the server may run inefficiently. For file pool repository servers, there may be increased user response times.

A need to increase the USERS value may also be indicated if users complain their applications are failing because their connections are being severed.

In this case, the application receives a reason code of 97400. A reason code of 97400, by itself, would not indicate you should increase the USERS value. 97400 merely indicates the connection was severed (without giving a specific reason). Further evidence you should increase the USERS value is that after the application failed, the server would still be available and the user would be able to process other commands and less-complex applications. (A user might say, "My application fails intermittently with a 97400, but I am always able to process other commands against that file pool immediately after the

error".) In most cases that do not involve the USERS value, a 97400 would typically indicate the server is no longer available for any number of reasons (server errors, communication errors, and so on)—the user would be unable to successfully process other commands that used the server.

The above condition could also occur when the user is entering a CMS command. In this case the symptoms would be the same, except the user would receive message DMS1151E instead of reason code 97400. The text of DMS1151E is:

```
File pool filepoolid is unavailable
```

Depending on what facilities are being used, the user may also receive a message that includes reason code 97400 in its display. (Some facilities display the reason code as part of the message.) OpenExtensions users or applications may receive return code ECMSERR accompanied by reason code JrFilePoolServer.

## Monitoring Physical Storage in User Storage Groups

This section applies only to file pool repository servers, not dedicated CRR recovery or dedicated FIFO servers.

To check this limit, periodically enter a QUERY FILEPOOL MINIDISK command. Check the Minidisk information for each storage group. Find the number of 4K blocks in-use for the storage group as well as the total number of blocks allocated. To determine the total number of blocks allocated to the storage group, add the 4K Blocks In-Use values to the 4K Blocks Free values in the storage group. You may want to record this value for future monitoring. You would increment the value whenever you add a minidisk to the storage group or just reissue QUERY FILEPOOL MINIDISK and recalculate.

Next, determine the ratio of total blocks in use to the total blocks. If it is near the limit (about 90 to 95 percent), it may be time to add a minidisk or move users to other storage groups. See [“Adding Minidisks in Multiple User Mode” on page 199](#) or [“Adding Minidisks in Dedicated Maintenance Mode” on page 201](#) for more about adding minidisks. See [“Moving an SFS User or BFS File Space in Multiple User Mode” on page 95](#) or [“Moving an SFS User in Dedicated Maintenance Mode” on page 96](#) for more about moving users.

If you monitor the physical storage often, you can avoid having to manually tally the QUERY FILEPOOL MINIDISK output by coding an exec to do it. A sample of such an exec is in [“TALLY EXEC” on page 677](#) which issues QUERY FILEPOOL STATUS.

**Note:** The contents of output from QUERY FILEPOOL MINIDISK may change in any future releases and should not be used by permanent applications to acquire data.

### ***What Happens When the Limit is Reached***

When physical storage in a storage group is exhausted, server processing continues, but users attempting to write to the storage group will receive error messages or error return codes. Unless users erase enough files to continue usual operation, you have to add a minidisk to the storage group, or, after shutdown, move users to a storage group that has free space. See [“Adding Minidisks in Multiple User Mode” on page 199](#), [“Adding Minidisks in Dedicated Maintenance Mode” on page 201](#), [“Moving an SFS User or BFS File Space in Multiple User Mode” on page 95](#) or [“Moving an SFS User in Dedicated Maintenance Mode” on page 96](#) for appropriate instructions.

**Note:** It is possible an incorrectly coded application is writing data to the storage group in an endless loop. Such a loop could consume all the physical space in the storage group. Any application that tries to write to the storage group after the space is exhausted will be rolled back unless the User Storage Group Full exit has been tailored to take different action. See [“Activating the Storage Use Exits \(SFS and BFS\)” on page 209](#) for more information. If a storage group's space is periodically exhausted, an incorrectly coded application may be causing the problem. To find the application, periodically enter QUERY FILEPOOL AGENT commands and look in the agent information for a user who frequently consumes a high number of uncommitted blocks. You can then enter a QUERY LIMITS command for the user to determine whether the user is in the storage group that periodically fills. Or, if you are using the accounting facility, you can examine the accounting records around the time the storage group filled to determine who caused the problem. (This is a good way to detect malicious users.)

## Monitoring Physical Storage in the Catalog Storage Group

This section applies only to file pool repository servers, not dedicated CRR recovery or dedicated FIFO servers.

Monitor the use of physical storage in the catalog storage group (storage group 1) the same as you monitor user storage groups (see previous section).

### *What Happens When the Limit is Reached*

When physical storage is exhausted in the catalog storage group, server processing stops. If the space was consumed legitimately, you have to add more DASD space to the catalog storage group. See [Chapter 9, “Managing Storage,” on page 195](#) for instructions.

**Note:** It is possible an incorrectly-coded application created file pool objects (for example, base files, aliases, and directories) in an endless loop. Such a loop could consume all the physical space in storage group 1. In this case, you would not add space to the storage group. Instead, you would restart multiple user mode processing. During the restart, the server would roll back any uncommitted changes made by the application. If the application happened to be periodically committing its erroneous changes, the user should delete the objects as soon as possible to prevent another server termination.

If you suspect a malicious user is periodically exhausting storage group 1 space, use the security audit facility to determine who is creating objects.

## Monitoring the MAXCONN Value

This section applies to all file pool servers.

The MAXCONN value is an operand of the OPTION directory control statement. It defines the maximum number of Inter-User Communication Vehicle (IUCV) connections for the server machine. Server machines use IUCV to communicate with user machines and to read and write file pool minidisks.

You can minimize the need to monitor MAXCONN by setting it to a high value, such as 2000. (Setting MAXCONN higher than 2000 when not necessary would waste too much virtual storage in the server machine.) For the formulas used to calculate the [MAXCONN](#) value.

To monitor the MAXCONN value, enter a QUERY FILEPOOL OVERVIEW command and look at the Connections Highest Value and the Maximum Number of Connections value in the File Pool Overview Information. (For a description of the command, see “[QUERY FILEPOOL OVERVIEW](#)” on [page 589](#).) If the “Connections Highest Value” is approaching the Maximum Number of Connections value, consider increasing the MAXCONN value.

### *What Happens When the Limit is Reached*

The server rejects additional requests to connect to the file pool server. Users will receive error messages or error return codes. Additional users are able to connect to the SFS file pool or CRR recovery server as other users end their connections by logging off or reloading CMS. You may need to increase the MAXCONN value for the file pool server machine.

## Monitoring the MAXDISKS Value

This section applies only to file pool repository, not dedicated CRR recovery or dedicated FIFO servers.

Monitoring the MAXDISKS value is applicable only to repository file pool servers. If you set MAXDISKS to a sufficiently high value during file pool generation, you seldom have to check this limit. In [Chapter 15, “Generating a File Pool and Server,” on page 247](#) a MAXDISKS value of 500 is recommended. Most file pools will not reach that limit.

If you do want to check whether you are approaching the limit, enter a QUERY FILEPOOL OVERVIEW command and look at the Minidisks in Use value. This includes in the count all storage group minidisks, and excludes the control and log minidisks. Also in the QUERY FILEPOOL OVERVIEW display is the current MAXDISKS value (Maximum Number of Minidisks in the display). So long as the Minidisks in Use value is less than MAXDISKS, you will be able to add another minidisk to the file pool. After the MAXDISKS value

is reached, you must regenerate the file pool. (See [Chapter 11, “Regenerating a Repository File Pool,”](#) on page 217.)

### **What Happens When the Limit is Reached**

When the number of storage group minidisks reaches MAXDISKS, you will be unable to run FILESERV MINIDISK or FILEPOOL MINIDISK to add more physical storage. You have to regenerate the file pool. See [Chapter 11, “Regenerating a Repository File Pool,”](#) on page 217.

### **Monitoring SFS Log Minidisk Usage**

This section applies only to file pool repository, not dedicated CRR recovery or dedicated FIFO servers.

The way that the server uses the file pool log varies depending on whether the BACKUP or the NOBACKUP startup parameter is in effect. Considerations for monitoring log usage in either case follow:

- If the BACKUP startup parameter is in effect, the server reclaims log space whenever the control data is backed up. The server automatically detects when the log is 80% full and starts a backup to free log space when BACKUP is in effect. Therefore, you know the log is too small if automatic control data backups are started during multiple user mode operation. Instead of waiting for automatic backups to occur, you can periodically enter QUERY FILEPOOL LOG commands to get an idea of the rate at which the log is filling. You would then be able to project whether the log would fill before your next scheduled backup and act accordingly.

Suppose, for example, you back up the control data every two days. At the end of the first day (about halfway to your next scheduled backup) you enter a QUERY FILEPOOL LOG command and see the following log information:

Log Information	
290	Number of Log Minidisk 4K Blocks
75%	Percent(%) of Log Space Used
93%	LUW Rollback/Suspend Threshold (% Log Space)
80%	Backup Threshold (% Log Space)
mm/dd/yy	Date of Last Control Backup
hh:mm:ss	Time of Last Control Backup

Because you are checking the logs about halfway between scheduled backups, you would expect the percentage of log space used to be about 40%. (40% percent was obtained by dividing the Backup Threshold value by 2.) Instead, the percentage of space used is 75%, which means you have used more than three-quarters of the log space available before an automatic backup will be started at 80% full. If file pool activity continues at that rate, an automatic control data backup will be started well before your next scheduled backup. It is time to increase the size of the logs, or schedule control data backups more frequently.

You can also use the QUERY FILEPOOL LOG command to tell if your logs are too large. The first clue you might be able to decrease the log sizes is that an automatic backup is never triggered. This would either mean your log minidisks are about the right size or they are too large. The only drawback of having logs that are too large is that DASD space is wasted. To determine whether your log minidisks are too large, you should enter a QUERY FILEPOOL LOG command whenever you back up the control data. Enter the command just before making the backup and record the Percent of Log Space Used value. If the percentage is consistently low, you can reduce the sizes of the log minidisks by an appropriate percentage. (Or, you can back up the control data less frequently.)

For example, suppose you back up the control data on Wednesdays and Fridays. Before each backup, you enter a QUERY FILEPOOL LOG command. After a few weeks of checking, you find the log minidisks never exceed 50% usage. This clearly indicates the log minidisks are too large. You might reduce the size of the log minidisks by about 30 or 35%. Be conservative. Otherwise you may find the server starting automatic backups again. It is more convenient to reduce the logs by small amounts rather than cope with unscheduled backups.

- If NOBACKUP is in effect, the server attempts to reclaim log space after it writes numerous blocks of data on the log. After numerous blocks are written, the server starts an operation that permanently records committed changes in the file pool and frees log space. (This is called a *checkpoint*.) Because



the server periodically reclaims log space when NOBACKUP is in effect, it is not necessary to monitor log usage. Repeatedly issuing QUERY FILEPOOL LOG would show wide fluctuations in log usage. The information cannot be used to tune the log. Instead, you have to use a *react* approach. That is, start with an average size log (as recommended in [Chapter 15, “Generating a File Pool and Server,”](#) on page 247) and increase the size if users complain their long-running applications fail. Before increasing the size of the log, you might ask the user to check the coding of the application to ensure it is committing its work units as soon as possible.

**Note:** The Backup Threshold, Date of Last Control Backup, and Time of Last Control Backup are not displayed in QUERY FILEPOOL LOG output when the NOBACKUP startup parameter is in effect.

See [“Replacing Both File Pool Log Minidisks”](#) on page 133 for instructions on changing the size of the logs.

### ***What Happens When the Limit is Reached***

If the log minidisks begin to fill and the BACKUP startup parameter is in effect, an automatic backup will be started. The backup should free enough log space to allow usual operation to continue.

If the NOBACKUP startup parameter is in effect, the log should seldom fill because the server frequently tries to reclaim log space. If the log minidisks are very small, however, and there is a long-running logical unit of work, it may not be possible for the server to reclaim the log space. In this case, the server rolls back the long-running logical unit of work when its log is becoming full. The server can then reclaim log space and continue processing. If this occurs frequently, you should increase the size of the log minidisks.

## **Monitoring Server Machine Virtual Storage Consumption**

This section applies to all file pool servers.

You can minimize the need to monitor the server machine's virtual storage consumption by defining the server to have ample virtual machine size. A virtual machine size of 32MB is adequate for most server machines.

If you want to monitor the server's virtual storage usage to avoid possible problems, enter a QUERY FILEPOOL OVERVIEW command and look at the Virtual Storage Highest Value in the File Pool Overview Information. (For a description of the command, see [“QUERY FILEPOOL OVERVIEW”](#) on page 589.) If the Virtual Storage Highest Value is approaching the Virtual Storage Size in bytes, you should increase the virtual storage size of the server machine or take other corrective action.

### ***Corrective Action***

If the Virtual Storage Highest Value is approaching the Virtual Storage Size in bytes, increase the virtual storage size of the server machine.

Reducing USERS, CATBUFFERS, or CTLBUFFERS could cause poor performance for SFS and CRR file pool users unless the demand on the file pool is also reduced. To reduce file pool demand, move users to other file pools.

### ***What Happens When the Limit is Reached***

When all virtual storage in a server machine is used, the server tries to continue processing. Users may get error messages (DMS1176E) and error codes (reason code 55000). OpenExtensions users or applications may receive return code ENOMEM accompanied by reason code JrSerStorageObtainErr. The server may sever the connections of selected users in order to free up virtual storage. In some rare cases, the server will stop because of insufficient virtual storage.

## **Monitoring Logical Catalog Space**

This section applies only to file pool repository servers, not dedicated CRR recovery or dedicated FIFO servers.

To determine how close you are to reaching the maximum amount of logical catalog space, enter a QUERY FILEPOOL REPORT command with the CATALOG option, or enter the QUERY FILEPOOL CATALOG command. You will see information similar to this portion of the QUERY FILEPOOL CATALOG display:

```
CATALOG SPACE INFORMATION
4291 Data Blocks
3004 Occupied Data Blocks
70% Percent Occupied Data Blocks
4292 Index Blocks
2747 Occupied Index Blocks
64% Percent Used Index Blocks
```

Near the bottom of the command results in the Catalog Space Information, look at the Percent Occupied Data Blocks value and Percent Used Index Blocks value. If either the data blocks or index blocks are about 90% full, you should begin planning for a file pool regeneration. During the regeneration, you should probably increase the MAXUSERS value by about 30% to 40%. Increasing the MAXUSERS value during a regeneration will cause additional control disk space to be used, so you should also plan on increasing its size by an appropriate amount. (Guidance is provided in [Chapter 11, “Regenerating a Repository File Pool,”](#) on page 217.) When regenerating the file pool, you should also re-evaluate your current MAXDISKS setting and increase it if necessary.

In the previous example of a portion of the QUERY FILEPOOL CATALOG display, the percent occupied data blocks is 70% while the percent occupied index blocks is 64%. A regeneration is not yet necessary, in this case, to increase MAXUSERS.

### ***What Happens When the Limit is Reached***

Logical catalog space is reserved during file pool generation. FILESERV GENERATE processing uses the MAXUSERS value to estimate and set the maximum logical catalog space. When the server runs out of logical space, it displays a warning message on its console and continues processing. Depending on their use of the file pool, users may receive error messages (DMS1146E) and error return codes (with reason codes 51010 or 51020). When the logical catalog space is exhausted, you need to increase the MAXUSERS value for the file pool. Follow the instructions in [Chapter 11, “Regenerating a Repository File Pool,”](#) on page 217.

## **Monitoring Total Physical File Pool Space**

This section applies only to file pool repository servers, not dedicated CRR recovery or dedicated FIFO servers.

To determine whether your current physical DASD allocation is approaching the limit of what the file pool can hold, enter a QUERY FILEPOOL OVERVIEW command. Make a note of the Potential Addressable 4K Blocks in File Pool value. This number is the maximum number of 4KB blocks that can exist in the file pool. If Defined Addressable 4K Blocks in the File Pool value is close to the potential addressable space, you should start planning a file pool regeneration. (In regenerating the file pool, you would increase the size of the control minidisk so more 4KB blocks can be addressed.)

**Note:** Undefined Addressable 4K blocks in the file pool is the Maximum number of 4KB blocks that can still be added to the storage groups in the SFS file pool (using either the FILESERV MINIDISK or FILEPOOL MINIDISK command).

### ***What Happens When the Limit is Reached***

If you do not do periodic monitoring, you know you have reached the maximum amount of space a file pool can hold when you try to add a minidisk to the file pool, and the addition fails with the following message:

```
DMS3913E File Pool CONTROL disk is incorrect size.
Reason=3
```

To increase the total file pool space, you must increase the size of the control minidisk and regenerate the file pool. See [Chapter 11, “Regenerating a Repository File Pool,”](#) on page 217.



## Monitoring Rollbacks Because of Deadlock

This section applies only to file pool repository servers, not dedicated CRR recovery or dedicated FIFO servers.

A deadlock occurs when user A holds a lock that user B needs, while user B holds a lock that user A needs. Without some sort of system intervention, both users would wait forever. The file pool server machine detects such deadlock situations and, if unable to resolve them internally, rolls back one of the logical units of work that is involved in the deadlock so the other can proceed.

Usually, chances are remote that any given logical unit of work will be unsuccessful because of a deadlock. However, the likelihood increases as the work load on that file pool increases.

Because rollbacks from deadlock are undesirable, you may wish to monitor them. To do this, enter a `QUERY FILEPOOL COUNTER` command after the file pool of interest has run for an extended period of time and look at the SFS and Byte File counter information: `Divide Rollbacks Due To Deadlock by Begin LUWs`. The result is the likelihood that any given logical unit of work will experience a rollback because of deadlock. You want this likelihood to be quite low. If you find involuntary rollbacks are occurring more often than the limit you have established, there are certain actions you can take that may reduce their frequency. These are described in *z/VM: Performance*.

## Performance Monitoring

This section applies to all file pool servers as explained in “Limit Monitoring” on page 58. But, file pool repository server monitoring is more important than CRR or FIFO server monitoring.

Monitoring the performance of your system is optional. Your decision on whether to monitor performance should be based on the importance of response time at your installation. When response time is critical, it is advisable to do some monitoring to identify performance trends. This lets you take appropriate actions before performance becomes a problem. Further, to effectively solve performance problems it is often helpful to have performance data that was gathered prior to the time of poor performance. This data is used for comparisons. If you wait until a performance problem occurs, it will not be possible to create the comparison data.

Short-term actions involve tuning the system to optimize performance, while long-term actions include anticipating and planning for hardware needed to support increased capacity.

In addition to response time, the size of your processor and the number of users it supports also are factors in deciding the worth of monitoring performance. Performance adjustments are often relatively fine. Their effects tend to be amplified on larger systems. On smaller systems, however, the effect of many performance enhancements are not as greatly amplified and in some cases may be unnoticeable. Thus, it is often not worthwhile to monitor performance on smaller systems.

For example, suppose your department provides computer services to other departments in the same company. Suppose, also, that as part of your agreement, you must provide a certain maximum average response time. In this case it is well worth your time to continually monitor performance. You would try to identify performance trends so appropriate actions can be taken before performance violates your service agreement.

If, on the other hand, your processor is used for departmental operations, response time may not be as critical. On smaller processors that serve fewer users, performance monitoring will likely have marginal benefit. In this case, you might not do any performance monitoring. Instead, you would wait until performance became unacceptable and then take corrective action, as described in *z/VM: Performance*.

If you are interested in monitoring a server for performance, see *z/VM: Performance*. Also see Chapter 10, “Reorganizing the File Pool Repository Catalogs,” on page 213 for more information that can affect performance.

## Stopping Server Processing

---

This section applies to all file pool servers.

## Stopping Server Processing

To stop a server machine from its secondary user console, use the CP SEND command to send a STOP operator command. For example, to stop the VMSERVU server machine from the MAINT administration machine, you would enter:

```
#cp send vmseivv stop
```

You can also enter the STOP command from the server machine console, in which case you would enter only:

```
stop
```

The STOP command does not immediately shut down the server. While the server does not accept any new work from users, it does allow any work currently in process to complete.

For file pool repository servers, *work in process*, means any user that is currently processing a logical unit of work in the server. As each logical unit of work completes, the following message is displayed so you can estimate when shutdown will complete:

```
DMS3029I nn logical units of work are still in-process
```

For *nn* you see the number of remaining logical units of work.

For CRR recovery servers, *work in process*, means any sync points currently in process. As each sync point completes, the following message is displayed so you can estimate when shutdown will complete:

```
DMS3029I nn logical units of work are still in-process and  
mm synchronization points are still in-process
```

For *nn* you see the number of remaining logical units of work and for *mm* you see the remaining sync points.

In some situations, you may not want to wait for all work in process to complete. You can immediately stop a server by using the IMMEDIATE operand:

```
#cp send vmseivv stop immediate
```

File pool repository servers immediately stop all in-process logical units of work. Users receive reason codes that indicate what happened. The next time the server is started, it automatically either rolls back the stopped logical units or goes into resynchronization.

CRR recovery servers immediately stop all in-process sync points. Applications receive reason codes that indicate what happened. The next time the server is started, it automatically resynchronizes any logged sync points that had completed the first phase of the two-phase commit.

**Note:** If you receive a message from the z/VM system operator informing you of a pending system shutdown (or re-IPL), be sure to enter the STOP or STOP IMMEDIATE command before the system shutdown occurs. This prevents CRR from trying to resynchronize previously successfully resynchronized work.

For more information, see [“STOP” on page 666](#).

## Control Data Backups at Shutdown

This section applies only to file pool repository servers, not dedicated CRR recovery or dedicated FIFO servers.

If the BACKUP startup parameter is in effect, the server will back up the control data after all in-progress work completes. (The BACKUP startup parameter is specified for both the VMSERVU and VMSERVS server machine.)

You can avoid backing up the control data by specifying NOBACKUP on the STOP command. Suppose, for example, you know the PUBSERV server has BACKUP specified as a startup parameter, but you just want to stop the server momentarily and do not want to back up the control data. From the secondary user console, you would enter:

```
#cp send pubserv stop nobackup
```

After in-progress work completes, the server stops without making a backup.

You can also enter the STOP IMMEDIATE command when the BACKUP startup parameter is in effect. You would enter STOP IMMEDIATE if you did not want to back up the control data and did not want to wait for in-progress work to complete.

## Other Ways to Stop Multiple User Mode Processing

This section applies to all file pool servers.

If you enter the HX command from the operator console, the server might not process it immediately. When it does process the command, the command is, in effect, the same as a STOP IMMEDIATE command. The same considerations apply.

One difference between HX and STOP IMMEDIATE is when HX is processed, the server will ask if you want a dump of the server machine. The dump is optional. You should respond "0" for no if there is no server problem.

If you enter CP and re-IPL CMS to stop multiple user mode processing, the effect is the same as a system error. That is, server execution stops immediately. The next time the server is started, all in-progress logical units of work at the time of the errors are rolled back.

## Changing the File Pool ID

This section applies to all file pool servers. You can change the file pool ID of a server permanently, or you can change the file pool ID temporarily to restrict users from accessing a file pool for a limited time. If you want to change the file pool ID temporarily, see [“Restricting User Access in Multiple User Mode”](#) on page 69. Otherwise, if you are permanently changing the file pool ID, then follow the steps in this section.

Permanently changing the ID of a file pool can cause a disruption to the file pool user community. You should plan, therefore, for a file pool ID change well in advance and make sure you are aware of the consequences to users and, perhaps, their applications. To permanently change the file pool ID, do the following:

1. For repository file pools, inform the file pool users of the impending change. They may have to update execs or programs to use the new file pool ID. If the file pool is available to remote users, remember to inform them as well.
  - Notify all CRR recovery server administrators where those CRR recovery servers have done an exchange of log names with this file pool. See [“SFS Log Name Table Changes”](#) on page 283 for more information. Both the CRR recovery servers and this repository file pool will need to perform an erase of the log name table entry for their respective partner. See [“QUERY LOGTABLE”](#) on page 658 for more information to find the list of affected CRR recovery servers.
  - Use the ERASE LUNAME command to erase the log name table entry for each CRR recovery server found in the list from the QUERY LOGTABLE command. The quiescing of CRR work is accomplished by the function of this erase which fails if there is an outstanding syncpoint for the affected log name table entry. For a description of the command, see [“ERASE LUNAME”](#) on page 427.
  - Each CRR recovery server on the list should erase its log name table entry for this SFS file pool by using the CRR ERASE LU command. For a description on how to use the command, see [“CRR ERASE LU”](#) on page 368.
2. For CRR file pools, inform the CRR administrator of the impending change. There are no file pool users for CRR.
3. Log on or reconnect to the server machine as follows.
  - If the server is currently processing in multiple user mode, reconnect to it and enter a STOP operator command. (For more about stopping multiple user mode processing, see [“Stopping Server Processing”](#) on page 65.)
  - Otherwise, if the server is not currently processing in multiple user mode, log on it.

## Changing the File Pool ID

If you set up the server machine for automatic starting, as instructed in Chapter 15, “Generating a File Pool and Server,” on page 247, you have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC issues a FILESERV START command, which is not desired at this time.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

Next, run the SETUP EXEC to access minidisks needed by the server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

If you forget to inhibit the execution of the PROFILE EXEC and the server successfully starts, stop the server by using any variation of the STOP operator command:

```
stop
```

Enter STOP NOBACKUP if the BACKUP startup parameter is in effect and you do not want to back up the control data at this time.

### 4. Select a new file pool ID.

The file pool ID should be unique among all other file pool IDs on the processor (or in the TSAF or CS collection, if your processor participates in one). For a description of startup parameter, see [FILEPOOLID](#) for the rules for selecting file pool IDs.

### 5. Update the DMSPARMS file.

Change the existing FILEPOOLID startup parameter to the new name. If there is not a FILEPOOLID startup parameter, specify one. For example, to specify a file pool ID of FINANCE3, you would update (or add) this parameter:

```
FILEPOOLID FINANCE3
```

For a description of startup parameter, see [FILEPOOLID](#).

### 6. Rename the POOLDEF file.

Change the file name of the POOLDEF file to the new file pool ID. For example, to change the name of the PUBS file pool to FINANCE3, you would enter:

```
rename pubs pooldef a finance3 pooldef a
```

### 7. Start the server for multiple user mode access:

```
fileserv start
```

### 8. This step applies only to repository file pools. Update the z/VM system directory entries for all file pool users that have a FILEPOOL parameter referring to the file pool. The FILEPOOL parameter is on the IPL CMS directory control statement. If, for instance, you changed the file pool ID from PUBS to FINANCE3, you would look for all IPL CMS control statements in the z/VM system directory that refer to PUBS:

```
⋮  
IPL CMS PARM FILEPOOL PUBS  
⋮
```

You would then change these control statements to refer to FINANCE3:

```
⋮  
IPL CMS PARM FILEPOOL FINANCE3  
⋮
```

Use the standard procedures at your installation for updating the z/VM system directory.

If the file pool was available to other processors in a TSAF collection, you should also review the z/VM system directories on those processors. It is possible, though unlikely, the IPL CMS control statements in those directories will refer to the file pool.

9. This step applies only to dedicated FIFO servers.

If you are using dedicated FIFO servers to perform FIFO services for one or more BFS repository file pools, you must update the DMSPARMS file of the corresponding file pools.

10. Update any IBM-level (ICOMDIR NAMES), system-level (SCOMDIR NAMES), or user-level (UCOMDIR NAMES) communication directories that refer to the old file pool ID.

**Note:** During installation, you may have supplied an alternate name for the service file pool, VMPSFS. If so, VMPSFS was defined as a nickname in the ICOMDIR NAMES file.

11. If AVS was used to make the file pool available to an SNA network, you will also need to update communications directories.

12. This step applies only to SFS file pools managed by DFSMS/VM.

If you are using DFSMS/VM to place files in migrated status and you rename a file pool, there are special considerations. For example, you must also rename the directory in the DFSMS/VM storage repository that supports the renamed file pool. See [z/VM: DFSMS/VM Customization](#) for more complete information.

13. This step applies only to repository file pools which are part of the DFSMS/VM storage repository.

If the file pool you are renaming is being used by DFSMS/VM as part of its storage repository, an entry for that file pool appears in the DFSMS control file, and you *must* update that entry with the file pool's new name. See [z/VM: DFSMS/VM Customization](#) for more complete information.

## Restricting User Access in Multiple User Mode

There may be situations in which you want to start a repository file pool server in multiple user mode, but prevent general access to it. This sort of restricted operation is useful for some recovery, test, and diagnosis procedures. During a restore of all storage groups, for example, a restricted multiple user mode session would protect users from being confused by the disappearance and reappearance of aliases and authorizations as each group is restored.

One simple technique for preventing users from accessing a file pool in multiple user mode is to temporarily change the ID of the file pool without telling anyone what it is. Because the users do not know the file pool ID, they are not able to access the file pool except by guessing. While this does not provide complete protection, it is usually adequate for the kinds of procedures being performed.

To temporarily change the name of a file pool, do the following steps while the server machine is not running.

1. Update the FILEPOOLID startup parameter in the DMSPARMS file.

For example, instead of:

```
FILEPOOLID DESIGN1
```

you might specify:

```
FILEPOOLID SYZYG
```

2. Rename the POOLDEF file accordingly.

Continuing the above example:

```
rename design1 pooldef a syzyg ==
```

If you use the FILEPOOL RESTORE or FILEPOOL RELOAD command to restore user storage groups, FILEPOOL RESTORE or FILEPOOL RELOAD processing will notice the file pool ID on the backup file does not match the current file pool ID. It will display the current and the old file pool IDs and ask you whether

## Restricting User Access

you wish to continue. If one of the displayed file pool IDs is the secret file pool ID and the other is the usual file pool ID, you should allow execution to continue. Otherwise, you are probably trying to restore the wrong backup file and should halt execution.

If the file pool is available to other remote systems in a SNA network, there is no need to update the communications directory with the temporary file pool ID. The file pool will be unavailable to users outside the local TSAF collection, which is desired in this case.

After the restricted session, change the FILEPOOLID startup parameter and rename the POOLDEF file back to its original name.

See [“SFS Log Name Table Changes” on page 283](#) for any Log Name Table implications of changing the file pool ID.

## Chapter 6. Managing Users and File Spaces

There are two types of file spaces: SFS and BFS. A BFS file space is also referred to as a Byte File System. In this section, BFS file space is used rather than the term Byte File System to help avoid confusion.

This chapter contains the following procedures for managing the use of a file pool:

- Enrolling SFS users and creating BFS file spaces
- Establishing clients for BFS file spaces
- Changing the amount of space assigned to a file space \*
- Changing a user's threshold value (SFS only)
- Checking to see who is enrolled or which file spaces exist
- Determining space assigned to a file space
- Enrolling PUBLIC
- Deleting enrolled users and file spaces
- Deleting everyone enrolled by ENROLL PUBLIC
- Moving a user or file space to a different storage group
- Moving a user or file space to a different file pool
- Renaming file spaces

The procedures followed by an asterisk (\*) are done most often. If you are new to file pool administration, you might want to review those procedures.

**Note:** This chapter is applicable only to SFS file pools, not CRR file pools, except for [“Step 4: Update the User's z/VM System Directory Entry”](#) on page 74. CRR file pools do not have users.

### Enrolling SFS Users and Creating BFS File Spaces

The ENROLL USER command can be used either to:

1. enroll a user in a file pool and create an SFS type file space for that user, or
2. to simply create a BFS file space

In the first case, SFS enrollment, the capability to connect to the file pool is implied by the enrollment. In addition, the file space is specifically associated with the enrolled user because the file space ID and top directory ID, along with SFS ownership, are the same as the user ID that is enrolled.

In the second case, BFS file space creation, the BFS file space may not be associated with the ability of a particular user to connect to and use the file space or the file pool. In BFS, file spaces are often used by a group of users. Furthermore, it is not necessary to be enrolled in BFS or its file pool in order to use the BFS file systems. A BFS user need only have permissions through POSIX UIDs (user IDs) or POSIX GIDs (group IDs) to access file space objects. When creating a BFS file space, ownership (UID) of the top directory of the file space is established, and this can be one basis for BFS establishing access to a file space and other file space objects. See the section on planning for OpenExtensions in [z/VM: CP Planning and Administration](#) for information on establishing permission to use BFS file spaces.

The ENROLL USER command is used while the file pool server is running in multiple user mode.

Whenever the ENROLL USER command is used to create a file space, four important events happen:

1. The file space is given a top directory.
2. The file space is assigned to a particular storage group.

3. Capability to connect to the file pool is established for the user that is associated with the associated top directory of an SFS file space and can exercise any privilege granted to PUBLIC as well as any privilege associated with the file space top directory ownership.
4. Optionally space is assigned (some initial number of 4K blocks).

The name of the top directory is the file space ID, followed by a period. If blocks are not initially assigned to the file space, authorized users will still be able to create directories and create objects in the directories that do not consume file space (blocks). Only files cause such space consumption and require there be unconsumed space in the file space.

Note that a file space ID need not exist in the z/VM system directory. For SFS, this means you can create file spaces before you have a virtual machine with that same name. The guidelines for BFS enrollments are discussed in [“Establishing Clients for BFS File Spaces”](#) on page 80.

For SFS, the following instructions assume the file space ID already exists in the z/VM system directory. If you prefer to enroll new z/VM users in a file pool first, you can skip over the instructions for updating the user's z/VM system directory entries. Simply make those adjustments when you later add the user ID to the z/VM system directory. Following is a list of the steps:

### Step 1

Determine the amount of space to be assigned to each file space.

### Step 2

Determine the storage groups to which the file spaces will be assigned.

### Step 3

Decide what will be accessed as file mode A (SFS only).

### Step 4

Update the z/VM system directory entry for the user.

### Step 5

Log on an administration machine.

### Step 6

Determine whether there is enough physical space.

### Step 7

Enter an ENROLL USER command.

### Step 8

Do optional activities for existing users (SFS only).

### Step 9

Back up the storage group.

### Step 10

Consider increasing USERS and MAXCONN for the server.

These steps should be followed to update the z/VM system directory to automatically access an SFS directory.

You should follow the steps until you become familiar with the process of creating file spaces, then you may instead want to see the description for [“ENROLL USER”](#) on page 422.

## Step 1: Determine Amount of Space to Assign Each File Space

For each file space being enrolled, decide whether that file space will be given space in the file pool, and, if so, how much. File pool space is assigned in units of 4KB blocks. You can assign a file space up to 2,147,483,647 4KB blocks using the ENROLL USER command. This can be increased using the MODIFY USER command. (The practical limit is the number of 4KB blocks in the file pool.) The space is used for 4KB file blocks.

You can increase or reduce the maximum amount of file space at any time by using the MODIFY USER command (See [“Changing the Amount of Space Assigned to a File Space”](#) on page 91.)



## Step 2: Determine Storage Groups to Assign File Spaces

When a file space is enrolled, it is assigned to a particular storage group. In deciding where to assign a particular file space, keep the following in mind:

- A storage group is backed up with the FILEPOOL command and restored as a unit. If you allocate a great number of file spaces to a single storage group, it may become too time consuming to back up the entire storage group at one time. You may prefer to limit the number of file spaces in a storage group simply to reduce the amount of data that needs to be backed up at one time.
- While the server balances DASD use within a storage group, you can balance overall DASD use among storage groups by assigning file spaces to under-used groups. Suppose, for example, the DASD volumes allocated to storage group 5 are overused while those in storage group 3 are under-used. You can help balance DASD use by assigning new file spaces to storage group 3.

Usually, you do not need to balance DASD use within a storage group. The server tries to use space evenly across the volumes allocated to the storage group.

- The sharing of files in a file pool is not restricted by storage groups. File spaces do not need to know the number of the storage groups in order to share files from those storage groups. Therefore, it is not necessary to have all department members, for instance, belong to the same storage group. On the other hand, you may find it convenient to place all department members in the same storage group if you want to impose a physical space limit on that department—it is your choice.
- Storage groups are recovered independently. If you have an application that updates the files of multiple users and requires the changes to these files are coordinated, consider placing the file spaces in the same storage group. Otherwise, if there is a data loss, one of the storage groups updated by the application could be restored to an earlier time. This would make the changes to those files inconsistent with the changes in the storage groups that were not restored.

## Step 3: Decide What Will Be Accessed as File Mode A (SFS Only)

This step only applies to the Shared File System file spaces. You can prepare the user's virtual machine such that CMS will automatically access either a 191 minidisk or an SFS top directory as file mode A. (The next step describes how to do this.) If you plan to give space in the file pool to a user, you should decide whether CMS should access the top directory as file mode A for that user. In making that decision, consider:

- For a new user to complete all the exercises in *z/VM: CMS Primer*, it is best to set up the user's virtual machine so a top directory in some file pool is automatically accessed as file mode A.
- Files stored in SFS directories allow the most function. In addition to the file sharing capabilities inherent in CMS commands that operate on SFS files, there are Callable Services Library (CSL) routines that provide many functions not available for minidisk files.
- Existing users may want to keep a 191 minidisk and have a file space available to them to be accessed as needed. This is perfectly acceptable. Both minidisks and SFS directories can be accessed at the same time from a virtual machine.
- Some existing users do not have applications that demand optimal A-disk input/output performance. You should consider moving the contents of those users' 191 minidisks to their top directories. You would then have CMS automatically access their top directories as file mode A. Later you can reclaim the DASD space used by these 191 minidisks. (Specific instructions are in later steps.)

Often the best candidates for this sort of migration are users that typically have a lot of unused minidisk space. When these users are moved to a file pool, physical DASD storage is consumed only for the blocks actually in use. DASD space is not wasted.

- Minidisk files afford the best performance. If the user needs optimal performance for his applications, you might still choose to have the user's top directory serve as file mode A. In this case, you would allocate a minidisk to the user that is large enough to support the data for the high-speed applications and move the remaining data to the top directory.

If you or the user prefer to keep a 191 minidisk to support the data needed by these high-speed applications, you could still move some of the data to a file pool and reduce that user's 191 allocation.

- Only one top directory can be accessed as file mode A. If the user already has space in some other file pool, you will need to determine which top directory is to serve as file mode A for that user.
- Applications written prior to VM/SP release 6 may be written in such a way that if they work on files in SFS directories, they work inefficiently. For example, some programs may be coded so that files remain open for extended periods of time when they might be closed and reopened as needed instead. This sort of application would consume file pool server resources unnecessarily. Until such applications are updated to work more efficiently, you may prefer for them to process against minidisk files.
- Applications written prior to VM/SP release 6 may be written in such a way that they do not work on files in SFS directories. If such an application is important to the user or is frequently used, you should consider whether it is practical for the user to have an SFS directory as file mode A.

Depending on how the application is coded, it might be reasonable to have an SFS directory as file mode A and a minidisk accessed as some other file mode for use by the application. Some applications, however, may be coded to expect only a minidisk for file mode A. If the application is frequently used, it could be inconvenient for the user to release and access SFS directories and minidisks whenever he or she needs to run an application. In this case, you might decide to let the user retain a 191 minidisk for file mode A and allow the user to control the accessing of the top directory. Another alternative would be to use an SFS directory as file mode A and provide the user with an EXEC (or tailor the application) to access minidisks as needed.

If you choose to have CMS access an SFS top directory as file mode A, a default file pool ID is automatically established for the user every time CMS is initialized. If the 191 minidisk is used as file mode A, a default file pool ID is not established. In this case, users must specify a file pool ID in their CMS commands that use the file pool or put a SET FILEPOOL command in their PROFILE EXECs. They should also add an ACCESS or VMLINK command to their PROFILE EXEC if they want a particular directory available whenever CMS is initialized. To give all users a default file pool, put a SET FILEPOOL command in the system profile (SYSPROF EXEC) on the 190 disk.

Finally, no matter what is accessed as file mode A, users must be told the ID of the file pool in which they are enrolled and the storage group number to which their file spaces are assigned. Users should know their storage group numbers because you, as an administrator, would sometimes communicate information about a particular group to all file pool users: for example, a warning to file pool users that storage group 3 is about to be restored. Storage group 3 users then know they will not be able to use their file spaces, and those in other storage groups will know they can continue their work but can expect some requests to be unsuccessful (those that refer to data in storage group 3).

### Step 4: Update the User's z/VM System Directory Entry

There are two updates you may need to make to the user's z/VM system directory entry:

1. If you want CMS to automatically access the user's top SFS directory as file mode A, add an IPL control statement (or update an existing one) with the operand PARM FILEPOOL *filepoolid*. For example, if user JIM has a file space in the file pool named TEST1, you would specify:

```
IPL CMS PARM FILEPOOL TEST1
```

When the user initializes CMS (IPL CMS), TEST1 is automatically established as the default file pool. (The equivalent of SET FILEPOOL TEST1: is issued for the user.) The system profile, which is processed during CMS initialization, will then automatically enter ACCESS (without parameters) for the user, thereby establishing the user's top directory as file mode A.

See [z/VM: CMS Commands and Utilities Reference](#) for a description of the SET FILEPOOL command.

2. Review the MAXCONN setting for a new SFS user or users of a new BFS file space.

If a user machine uses more than 64 IUCV or APPC/VM connections at one time, you must specify the MAXCONN operand of the OPTION directory control statement in its z/VM directory entry. (The default MAXCONN value is 64; this should be adequate for most uses.)

When a user begins using the file pool or runs applications that require IUCV connections, it is possible he or she will exceed the MAXCONN limit. In this case, CMS displays the following error message:

```
DMS1174E You have tried to establish more APPC/VM connections than is
         allowed for your user ID
```

Or, if the limit is exceeded in an application program, the application will receive an error return code (a return code of 8 and a reason code of 97250). OpenExtensions users will receive an error return code of ECMSERR with a reason code of JrMaxconnExceeded.

A short-term solution is for the user to re-IPL CMS. If the problem recurs, you should increase the user's MAXCONN setting.

As you gain experience with the amount of IUCV and APPC/VM activity at your installation, you may want to adjust MAXCONN for all users. Consider the following:

- For SFS, each CMS work unit requires a single connection for each file pool server used.
- For BFS, each process requires at least a pair of connections for each file pool server used. File pool servers are involved when a BFS client mounts a Byte File System file space (explicitly or implicitly), including the BFS root file space. The first BFS file space mounted for each file pool server requires an extra connection pair. Therefore the connections to a particular file pool server can be expanded as:

$$(\text{Number\_of\_Processes} * 2) + 2$$

where each process has mounted at least one BFS file space in the file pool.

- Where a file pool has off-loaded FIFO (named pipe) services to another file pool, an additional pair of connections is required for each process that uses named pipes off-loaded to such a file pool server.
- Additional connections are acquired depending on the CMS facilities being used. (For example, an additional connection is acquired for XEDIT.)
- Regarding CRR, users who use SFS (and other resources that participate in CRR) and protected conversations require one APPC/VM connection for the CRR recovery server's logging link.
- Even more connections may be used if the user runs application programs that either connect to file pools or that use IUCV or APPC/VM for other reasons. For instance, a user might run an application that connects to a file pool and to a Structured Query Language/Data System (SQL/DS) database. SQL/DS also uses APPC/VM.

To specify MAXCONN, add or update the OPTION control statement in the virtual machine's z/VM system directory entry. For example:

```
OPTION MAXCONN 128
```

If the MAXCONN operand must be adjusted, make the adjustment now using the procedures defined at your installation for updating the z/VM system directory.

3. For new BFS file spaces, you need to establish POSIXINFO statements for BFS clients to have permission to use the file spaces. For BFS you will need to include the POSIXINFO statements in the z/VM system directory entry. Three statements, POSIXGLIST, POSIXINFO, and POSIXOPT, are general directory control statements. The fourth statement, POSIXGROUP, is a global directory control statement. POSIXGLIST specifies all POSIX groups of which the user is a member. POSIXGROUP defines a POSIX group. POSIXINFO specifies a user's POSIX information. POSIXOPT specifies option settings related to a user's POSIX capabilities. For information about coding these statements, see *z/VM: CP Planning and Administration*. For information about establishing BFS clients use of new BFS file spaces see “Establishing Clients for BFS File Spaces” on page 80.

For more information about the IPL CMS control statement, MAXCONN, and the z/VM system directory, see *z/VM: CP Planning and Administration*.

Using the appropriate procedures defined at your installation, update the user's z/VM system directory entry.

## Step 5: Log On an Administration Machine

If you are not already logged on to an administration machine, log on one now.

## Step 6: Determine Whether There Is Enough Physical Space

When enrolling new file spaces in a file pool, you might consider checking the amount of physical DASD space available (unused) in the storage group. Use the QUERY FILEPOOL MINIDISK command and check the Storage Group Minidisk Information and Storage Group Minidisk Totals Information it displays. An alternative to QUERY FILEPOOL MINIDISK is the Query User Storage Group CSL Routine which can be used to check available storage group use. See [“QUERY FILEPOOL MINIDISK” on page 586](#) and [z/VM: CMS Callable Services Reference](#) for more about the Query User Storage Group CSL Routine.

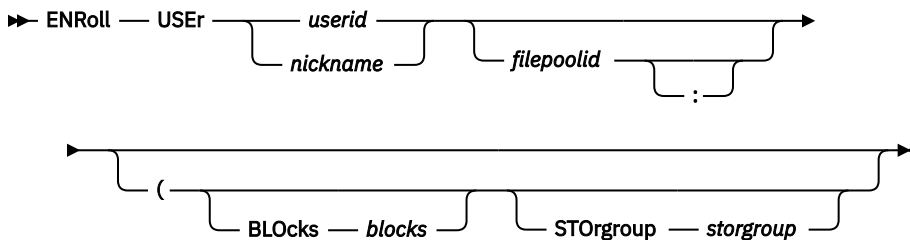
There should be enough physical space to support the file space's initial set of files. For new z/VM users, one can expect little file activity initially, so even a nominal amount of unused DASD space may be sufficient for a while. For existing z/VM users, you can expect more activity, especially if the user is moving existing files into the file pool. When the contents of a 191 minidisk is being moved to the file pool, the user might be able to tell you the number of 4KB blocks they will be immediately using. The storage group in which they are enrolled should have at least that many unused 4KB blocks.

After you have some experience with the rate at which users consume physical storage at your installation, it will become easier to decide whether the addition of a certain amount of users merits the addition of DASD space to the storage group to increase the file space. In any case, the file pool server displays a message on its console when the physical storage in a storage group is almost exhausted. At this point, you can add minidisks to the storage group, or move users to another storage group that has sufficient available space.

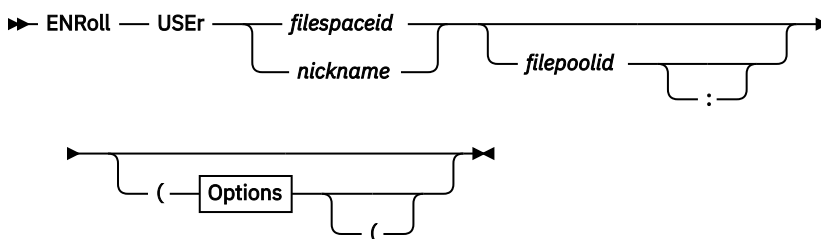
In general, it is a good policy to over-commit the storage group space. That is, it is good to assign more blocks to users than you actually have physical DASD space to support. This is particularly true if you are using DFSMS/VM to manage your storage. When infrequently used files are placed in DFSMS/VM migrated status, file blocks no longer reside in the storage group but are still counted in a user's file space usage. This lets you conserve your DASD space. Add DASD space only when necessary.

## Step 7: Enter an ENROLL USER Command

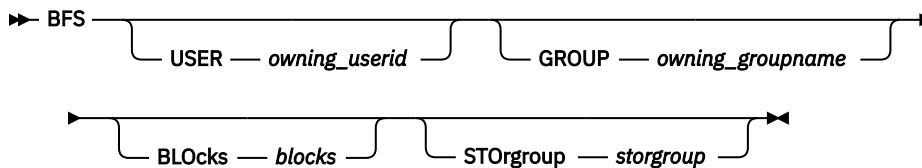
To create new SFS file spaces in a file pool, use the ENROLL USER command. You must have file pool administration authority to enter the command. An abbreviated format of the command is:



To create new BFS file spaces in a file pool, use the ENROLL USER command with the BFS operand. You must have file pool administrator authority to enter the command. An abbreviated format of the command is:



### Options



For a complete command description, see [“ENROLL USER” on page 422](#).

For SFS, the *userid* corresponds to the user ID that is to be enrolled in the file pool. This is the CP user ID of the person you are enrolling. If the user is on a processor in another TSAF or CS collection, that user should first be assigned a user ID on a processor within your TSAF or CS collection before he or she is enrolled. If the user is on a processor in an SNA network, that user should first be assigned a user ID on your processor or on a processor within your TSAF or CS collection before he or she is enrolled. See [Chapter 13, “Setting Up a File Pool for Remote Use,” on page 233](#) for additional considerations.

The BLOCKS option indicates the number of 4096-byte blocks that are to be allocated for the file space. The maximum number of blocks you can assign to a file space is 2,147,483,647. (The practical limit is somewhat less than the number of 4KB blocks in the file pool.) Remember the blocks assigned represent potential (or *logical*) space. The server does not need real DASD space until files are created in the file space.

The number of blocks specified represents the maximum number of *committed* blocks. (That is, blocks that a user or application wants to permanently write to the file pool.) For SFS, there is no restriction, other than the physical DASD space in the storage group, on the number of uncommitted blocks a user or application can consume. This lets an application, for example, create a temporary work file that exceeds the number of unused blocks available to the user. When the application no longer needs the temporary file, the application can either:

- Roll back the file instead of committing it, or
- Erase the file before committing it, then commit the file instead of rolling it back.

For BFS, this concept of uncommitted blocks is not pertinent because all blocks used are committed when they are submitted to the file pool server.

If you do not specify BLOCKS, the file space is not given any initial space (blocks). A file space without space still has a top directory. The user can create a directory structure and put objects other than files in the file space.

The STORGROUP option identifies the storage group to which the file space is to be assigned, regardless of whether you have given it any space. The number specified must be between 2 and the maximum number of storage groups defined for the file pool, inclusive. The storage group must exist. That is, it must have at least one minidisk assigned to it. If you omit this parameter, the file space is assigned to storage group 2.

The BFS operand specifies the file space is to be created is a Byte File System file space (BFS). The USER *owning\_userid* option identifies the VM user ID whose UID should become the owning UID of the top directory of the file space.

The GROUP *owning\_groupname* option identifies the group name whose GID should become the owning GID of the top directory of the file space.

If you do not remember which storage groups exist in the file pool, enter:

```
query filepool storgrp filepoolid
```

For *filepoolid*, substitute the name of the file pool in question. The resultant display shows valid storage groups for a file pool.

A portion of the display might look like this:

Storage Group No.	4K Blocks In-Use	4K Blocks Free
1	296 - 34%	580

## Enrolling SFS Users and Creating BFS File Spaces

2	264	-	18%	1218
3	0	-	0%	1482
4	0	-	0%	1482
5	0	-	0%	1482

For example, to enroll SFS user Mary in the file pool named TEST1 without space to create files, enter:

```
enroll user mary test1 (storgroup 3
```

Mary is assigned to storage group 3. To enroll user Bob in the same file pool with 1000 blocks of space (4,096,000 bytes), enter:

```
enroll user bob test1 (blocks 1000 storgroup 2
```

Bob is assigned to storage group 2. All 1000 of Bob's blocks will be allocated from that storage group.

Another example, is to enroll BFS file space JoAnne in the file pool named OURS1 without space to create files, enter:

```
enroll user joanne ours1 (bfs user joanne group ourgroup
```

JoAnne is assigned to storage group 2 by default.

When enrolling a group of SFS file spaces, try using the NAMES command to create a list of file spaces. Then enter an ENROLL USER command for the nickname. Suppose you define a nickname SALEDEPT that identifies all users in the sales department. To enroll those users, each with 100 blocks, enter:

```
enroll user saledept test1 (blocks 100 storgroup 5
```

Each name in the names files will be enrolled individually.

Do not enroll user IDs beginning with a plus (+) or minus (-) character, or that contain a period (.) or colon (:). These characters are used for directory names, of which user ID is a part. If you were to use these characters in a user ID and then specify that user ID in a directory name, CMS would not be able to tell whether the characters are part of the user ID or directory separator characters.

Also, be aware that deadlocks can occur when file spaces are enrolled concurrently by different administrators, especially if the user IDs are standardized (USER1, USER2, USER3, for example). In this case, one of the jobs is automatically rolled back. If this occurs, retry the command.

**Note:** For some administrative tasks you might find it desirable to communicate with all the users of a storage group or all the enrolled file pool users. To make it easier to communicate with groups of file pool users, consider using the NAMES command to create and maintain a list of users for each storage group. Also consider creating a nickname that contains a list of all the storage group nicknames. By doing this, you can easily communicate file pool news to users using the NOTE or TELL commands.

In some situations it might be useful to be able to list information about enrolled file spaces in storage group order. An example of an exec to do so is in [“WHO EXEC” on page 681](#).

**Note:** The CSL routine that corresponds to the ENROLL USER command is DMSENUSTR. For information on DMSENUSTR, see [z/VM: CMS Callable Services Reference](#).

## Step 8: Do Optional Activities for Existing Users (SFS only)

Existing CMS users typically have a 191 minidisk at the time you enroll them in a file pool. Following are instructions for optional activities you may want to do for these users:

1. Suppose you want to move the user's files from the 191 minidisk to his SFS top directory. If you know the z/VM password of the user's virtual machine, you can log on and move the files. Or, you can have the user move the files. How the files are moved depends on whether you have specified the FILEPOOL parameter, an IPL CMS control statement, in the user's z/VM system directory entry.

If you have not added the PARM FILEPOOL *filepoolid* operand to the user's IPL CMS directory control statement, the user can transfer the files to VMSYSU as follows:

- a. The user logs on
- b. The user enters: SET FILEPOOL VMSYSU
- c. The user accesses his top directory as some file mode. For example:

```
access . e
```

- d. The user copies all the files to his top directory.

Before issuing the COPYFILE command, temporarily ensure no other disks or directories are accessed as x/A. You can use the QUERY DISK command to determine all current access modes. Otherwise, COPYFILE will not only copy the files on the 191 disk, but will also copy files on anything that is an A extension. Now, enter the COPYFILE command as shown here:

```
copyfile * * a = = e (olddate
```

Because the FILEPOOL operand is not specified in the user's z/VM directory entry, the user's 191 minidisk is still automatically accessed as file mode A. You may now want to reaccess the disks and directories back to A extensions that you temporarily changed previously in this step.

If you have added the PARM FILEPOOL *filepoolid* operand to the user's IPL directory control statement, the user needs to access his *old* 191 minidisk as some other file mode, and then copy the files.

In either case, all or a portion of the files can be moved to the user's top directory. The user can organize these files into subdirectories at some later time.

2. If you choose not to have the user's top directory accessed as file mode A, you might consider setting-up default file pool IDs for that user. By having a default file pool ID, the user can abbreviate references to files when using fully qualified names. For example, without a default file pool ID, the user would have to specify the file pool ID when creating a directory:

```
create directory develop:.version1.routines
```

Here the user ID is allowed to default. If the default file pool ID is set to DEVELOP, the user can also let the file pool ID default:

```
create directory .version1.routines
```

To set a default file pool ID, the user can enter the SET FILEPOOL command. To avoid having the user enter the command at the beginning of every CMS session, you can:

- Add the SET FILEPOOL command to the system profile, if you want to establish a default file pool ID for your entire installation.
- Or, to establish a default file pool ID for individual users, have them add a SET FILEPOOL command to their profile execs. They can also add an ACCESS command for their top directories.

For example, suppose you enroll user JANET in the RESEARCH file pool. Janet's file mode A is still a minidisk, but you would like to set up her PROFILE EXEC so the default file pool is RESEARCH. You'd also like her top directory accessed as her file mode B. You would add these commands to her PROFILE EXEC:

```
set filepool research
access . b
```

If users choose to change the default, they can do so at any time by entering another SET FILEPOOL command.

3. If you move all the user's 191 minidisk files to the file pool, you can remove the MDISK statement for the 191 minidisk from the user's z/VM system directory entry. (Before removing the MDISK statement, be sure to establish the user's top directory as file mode A.) The DASD space can be used for other purposes.



### Step 9: Back Up the Storage Group

After a file space is added to a storage group, it is recommended you back up that storage group. This saves you from having to re-enroll the user should you need to restore the storage group. See [“Backing Up User Data”](#) on [page 111](#) for instructions on backing up user data.

### Step 10: Consider Increasing USERS and MAXCONN for Server

When you add file spaces to the file pool, you should ensure the USERS startup parameter still accurately reflects the maximum number of logged on file pool users during peak usage. You should also check the MAXCONN value on the OPTION directory control statement in the z/VM system directory entry for the server. You may need to increase the MAXCONN value to support the additional file spaces. See [“Limit Monitoring”](#) on [page 58](#) for more about monitoring the USERS and MAXCONN values.

## Establishing Clients for BFS File Spaces

---

Unlike SFS users, which generally have direct ownership for one, and only one SFS file space, BFS clients need not have this one-to-one relationship to a particular BFS file space. Another way of expressing this is that there is no implicit repository file pool connection capability associated with a BFS file space. Also, the name of a BFS file space need not have the same ID as a virtual machine (client) ID. You have considerable flexibility in establishing the relationship between BFS clients and BFS file spaces. Following are some examples of this flexibility:

- A single client may own one or more file spaces
- Groups of clients can share one or more BFS file spaces
- All clients can share all BFS file spaces
- All clients can share a single BFS file space

A file space is an administrative control point for managing space consumption by client's that write to regular files that are stored in the file space. When you create a file space with the [“ENROLL USER”](#) on [page 422](#) command, you establish a number of logical blocks that limit file creation and extension in that file space. This limit applies only to BFS regular files, not to other BFS objects such as directories and links. Even after a file space is created you can dynamically change the number of logical blocks assigned to a file space with the [“MODIFY USER”](#) on [page 544](#) command. The sum of the number of logical block limits for each file space in a file pool is expected to be greater than the minidisk space assigned to the file pool. That is, the file space limit is not a reservation of space in the file pool. Typically, not all file spaces are consumed at or near their limit, allowing the file pool limit to be considerably less than the sum of the associated file space limits. Because you do not have to reserve space on a file space basis, you have the benefit of avoiding wasted space overall.

A file space is not only a means for managing space, but is also a container for a directory and object hierarchy, which provides a context for BFS clients. When a BFS client mounts a file space or subset of a file space, the client extends the potential addressability scope to an additional set of BFS objects.

The Byte File System also controls which clients can use specific directories, files, and other objects in BFS file spaces by applying the POSIX permission checking rules. As an administrator, when you create a file space, it creates a root directory for the file space with the same name as the file space name. For this or any other object you create in the file space, you need to override the default owner (creator) and establish permissions using the OPENVM PERMIT and OPENVM OWNER commands (see [z/VM: OpenExtensions Commands Reference](#) for more information). You also need to establish appropriate UIDs and GIDs for clients and groups of clients with the POSIX system directory statements. See [z/VM: CP Planning and Administration](#) for information about coding the statements. The statements are:

- POSIXGLIST - A general statement that specifies all POSIX groups of which the user is a member.
- POSIXGROUP - A control statement that defines a POSIX group.
- POSIXINFO - A general statement that specifies a user's POSIX information.
- POSIXOPT - A general statement that specifies option settings related to a user's POSIX capabilities.



In summary, a file space provides both a context for extending addressability through OPENVM MOUNT commands and a control of logical space consumption. It also is a hierarchical collection of objects that are subject to the POSIX permission rules as [Figure 10 on page 81](#) shows.

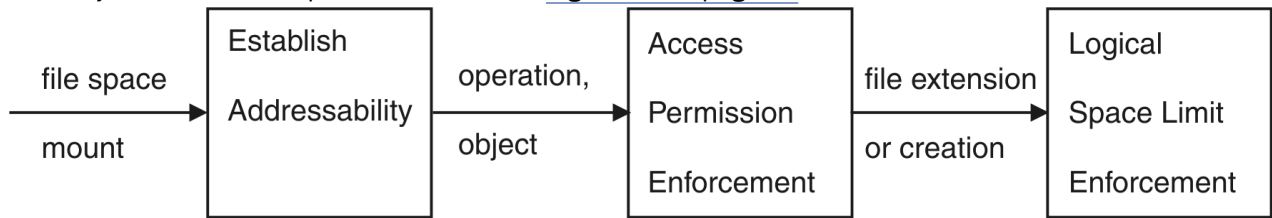


Figure 10. A File Space

Every BFS object is contained in a particular file space. In the case of regular BFS file objects, data blocks for storage of the file's data bytes are subject to the logical block limitation of the containing file space. Logical space for objects other than regular files is not managed through file space limits. That is, for example, directories and links do not affect logical block consumption and are not affected by the block limit that is assigned to a file space.

Just as in SFS, you assign the block limit for a file space when the file space is created with the “ENROLL USER” on [page 422](#) command and you can change that assignment dynamically with the “MODIFY USER” on [page 544](#) command. Unlike SFS, threshold warnings are not applied to BFS file spaces.

If you need to enforce logical space consumption on an individual client basis, you may consider assigning file spaces to individual clients. If you need to enforce space consumption by particular applications, you can assign file spaces to those applications. In addition you can combine and group clients and applications in many ways. For example, you can assign a file space to a group, such as the 6 members of department 16 (Ed, Sue, Terry, Brad, Eugene, and Betty) and establish subdirectories and permissions appropriate to the department members. Now the members of department 16 are collectively subject to the logical space limitation you have assigned to this file space. Also, any department member can mount the file space. However, you could set the permissions for the subdirectories for each department member in such a way as to give each member control over their own objects. [Figure 11 on page 81](#) illustrates this file space (before objects are added to the member's subdirectories):

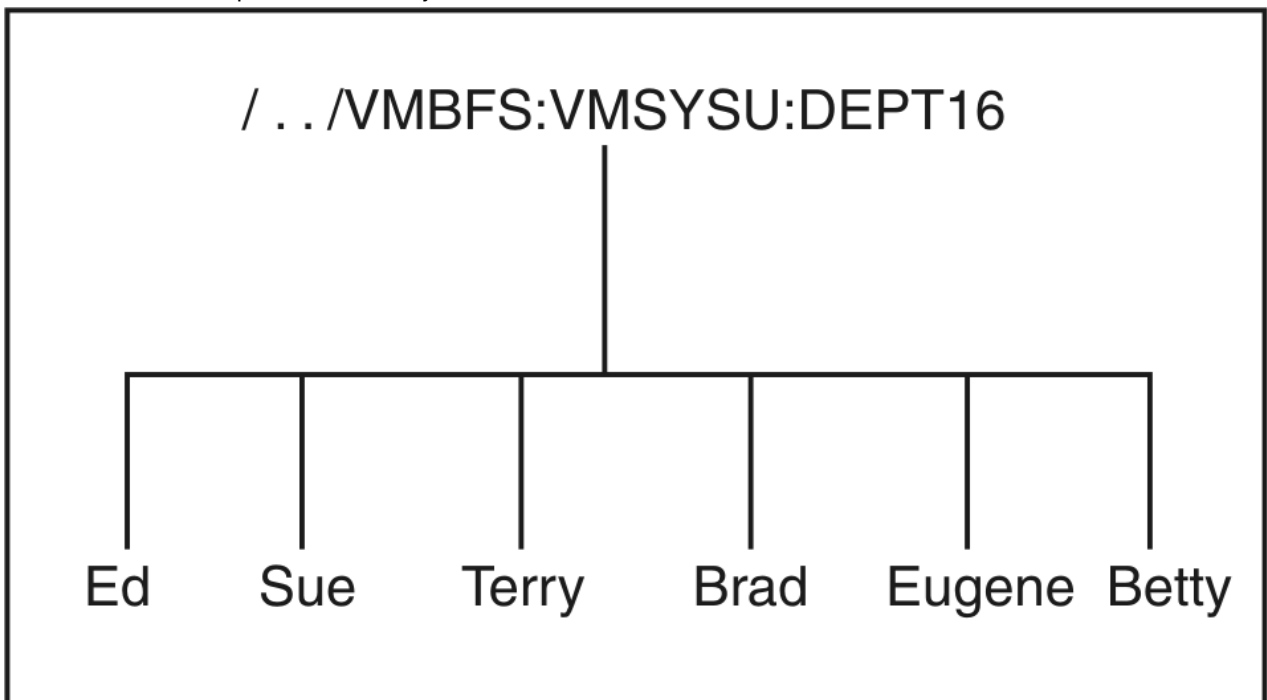
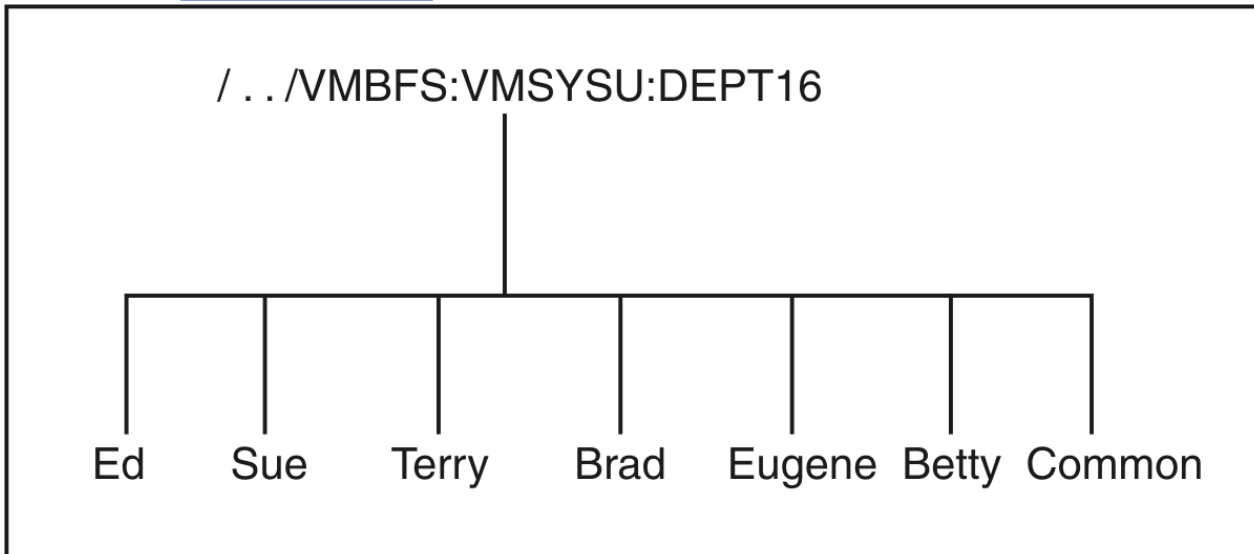


Figure 11. A File Space For Department 16

You may also require common departmental data objects that are accessible by every department member. This could be accomplished by establishing another subdirectory (Common) and using a group

permission to control access to it, (that is, permission being made available to all department members) as shown with Figure 12 on page 82:



*Figure 12. A File Space For Department 16 with Subdirectory Common*

You could also create one file space each for the SALES and the INVENTORY applications. These would contain objects and consume space for those particular application purposes. Further assume there are some subdirectories in the two applications file spaces as follows:

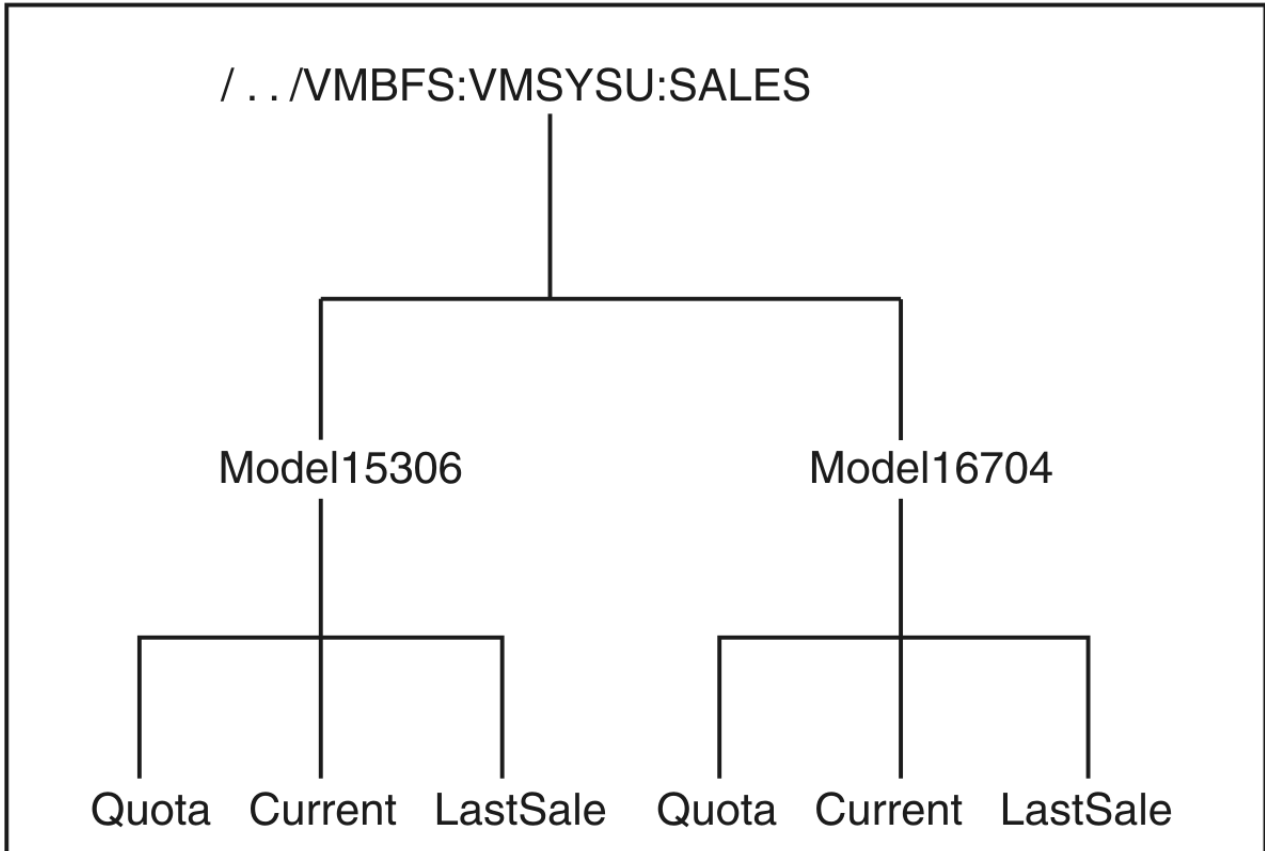
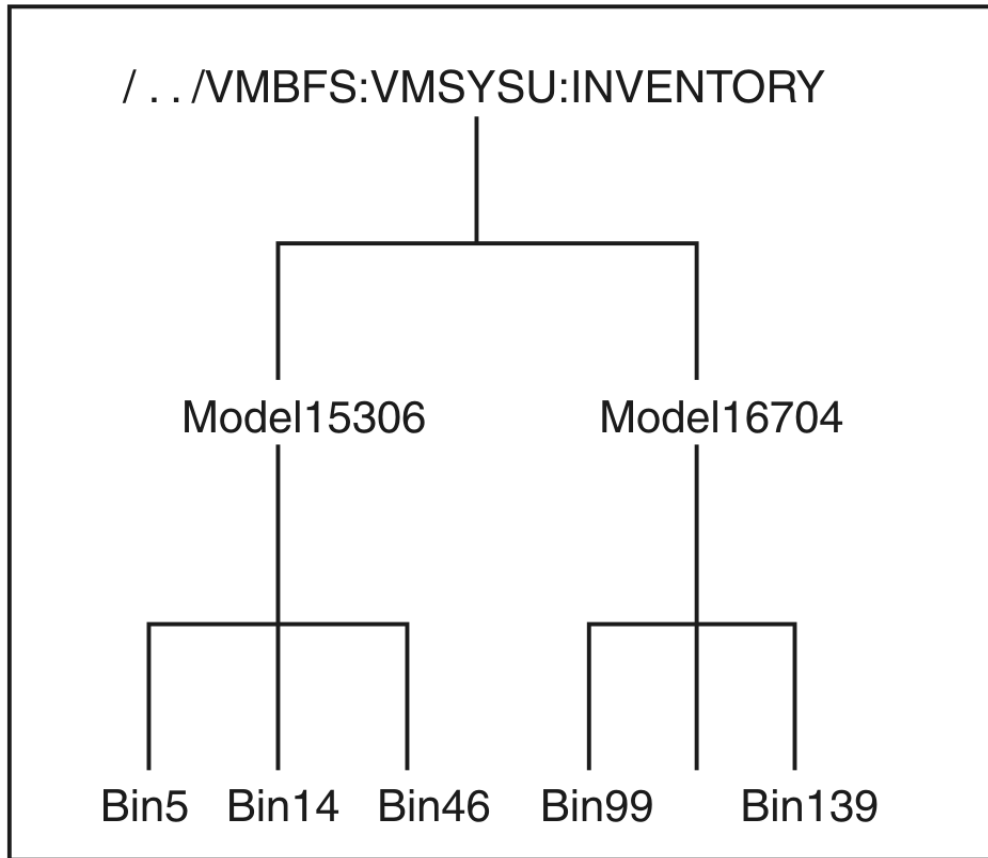


Figure 13. Example with SALES and INVENTORY

Further, assume all the members of department 16 have write permission for INVENTORY and its objects. But for the SALES application file space objects, only Sue and Brad have permission. Of course permission can be much more granular, affecting individual objects in the file spaces and involving various levels of access (such as, read, write, and search). If you intend to extend addressability by mounting additional file spaces, you need to establish a mount point in your current directory hierarchy. Usually this is done by establishing a directory that corresponds to that mount point. Then the OPENVM MOUNT command replaces that directory with the directory of the mounted file space. In [Figure 14 on page 84](#) the Inv and Sal directories are established for the SALES and INVENTORY application file spaces. The directories are established for those clients that have permission to use the INVENTORY or SALES file space objects as mount points for those file spaces.

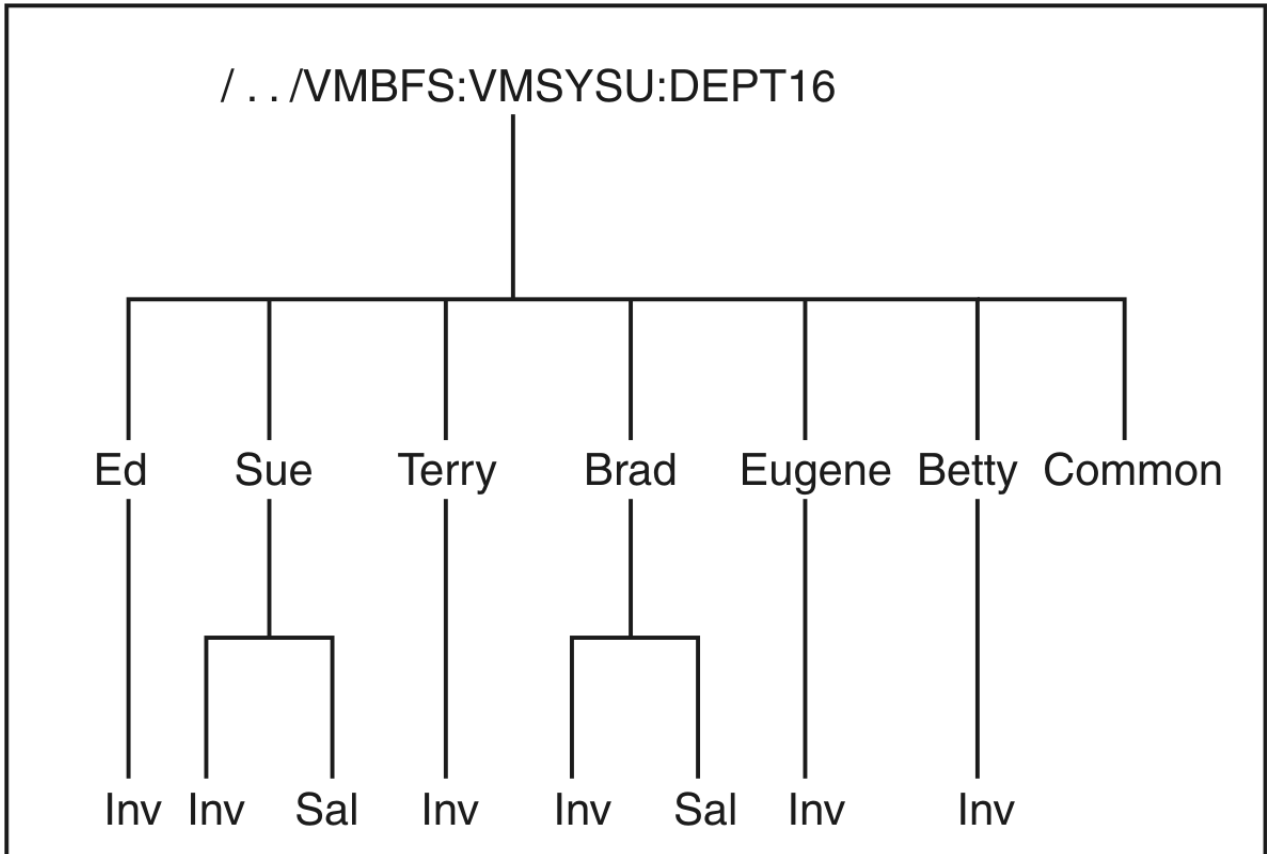


Figure 14. Example Department 16 with the Inventory and Sales File Spaces

### Example of Mounting a File System

Now let us look at some examples of mounting file systems. Let us first assume you have created directory /dept16 in the default file space, ROOT, which is defined in the default file pool, VMSYS, as [Figure 15 on page 85](#) shows:

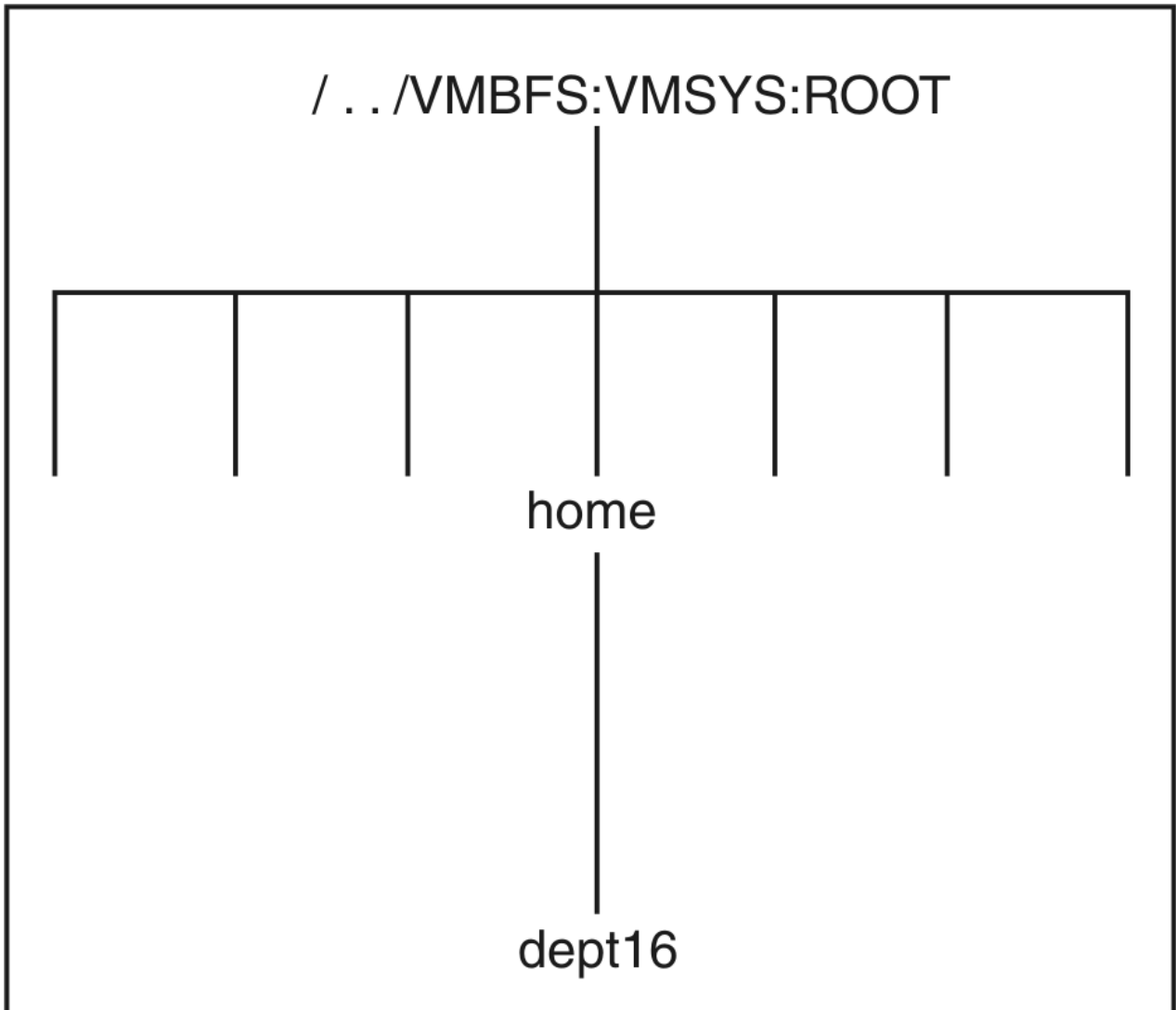


Figure 15. Example `/.. /VMBFS:VMSYS:ROOT`

See “BFS Root File Space in the VMSYS File Pool (Default)” on page 31 for more information on the contents of this ROOT file space in VMSYS.

Further assume the rest of the file spaces in the examples (DEPT16, INVENTORY, and SALES) are defined in production file pools, such as VMSYSU. With this, when client Ed issues the following mount commands:

```

OPENVM MOUNT /.. /VMBFS:ROOT/ /
OPENVM MOUNT /.. /VMBFS:VMSYSU:DEPT16/ /home/dept16
OPENVM MOUNT /.. /VMBFS:VMSYSU:INVENTORY/ /home/dept16/Ed/Inv
  
```

he will create a logical hierarchy as shown in [Figure 16 on page 86](#).

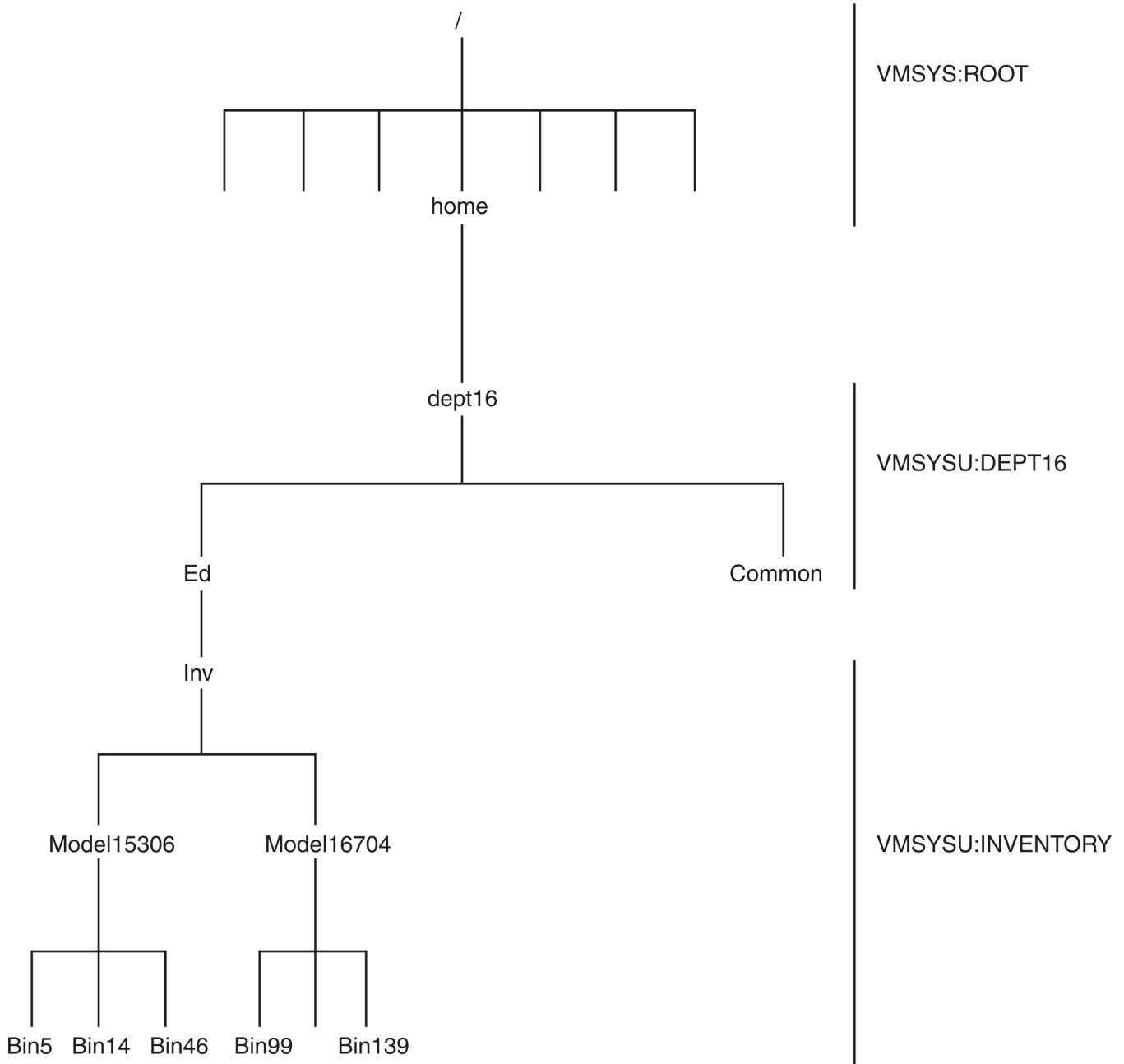


Figure 16. Example - the result of the Ed's commands

Note the root of Ed's logical BFS hierarchy is the root directory of byte file system file space ROOT in file pool VMSYS. The VMSYS file pool is intended primarily for read-only data, such as the storage of product code. It is also a LOCAL file pool so as to restrict such product code to a single system. Going down the tree, you can see directory /home/dept16 is the root directory of file space DEPT16. Also directory /home/dept16/Ed/Inv is the root of the file space INVENTORY. The two file spaces, DEPT16 and INVENTORY, are in file pool VMSYSU.

Similarly, assume client Brad issues the following mount commands:

```

OPENVM MOUNT ../../VMBFS:VMSYS:ROOT/ /
OPENVM MOUNT ../../VMBFS:VMSYSU:DEPT16/ /home/dept16
OPENVM MOUNT ../../VMBFS:VMSYSU:SALES/ /home/dept16/Brad/Sa1
  
```

We can see Brad's hierarchy is similar to Ed's, except Brad has mounted a different file space, SALES. The directory `/home/dept16/Brad/Sal` is the root of the file space SALES in this case. In Brad's case `/home/dept16/Brad/Inv` is an empty directory as [Figure 17](#) on page 88 shows.

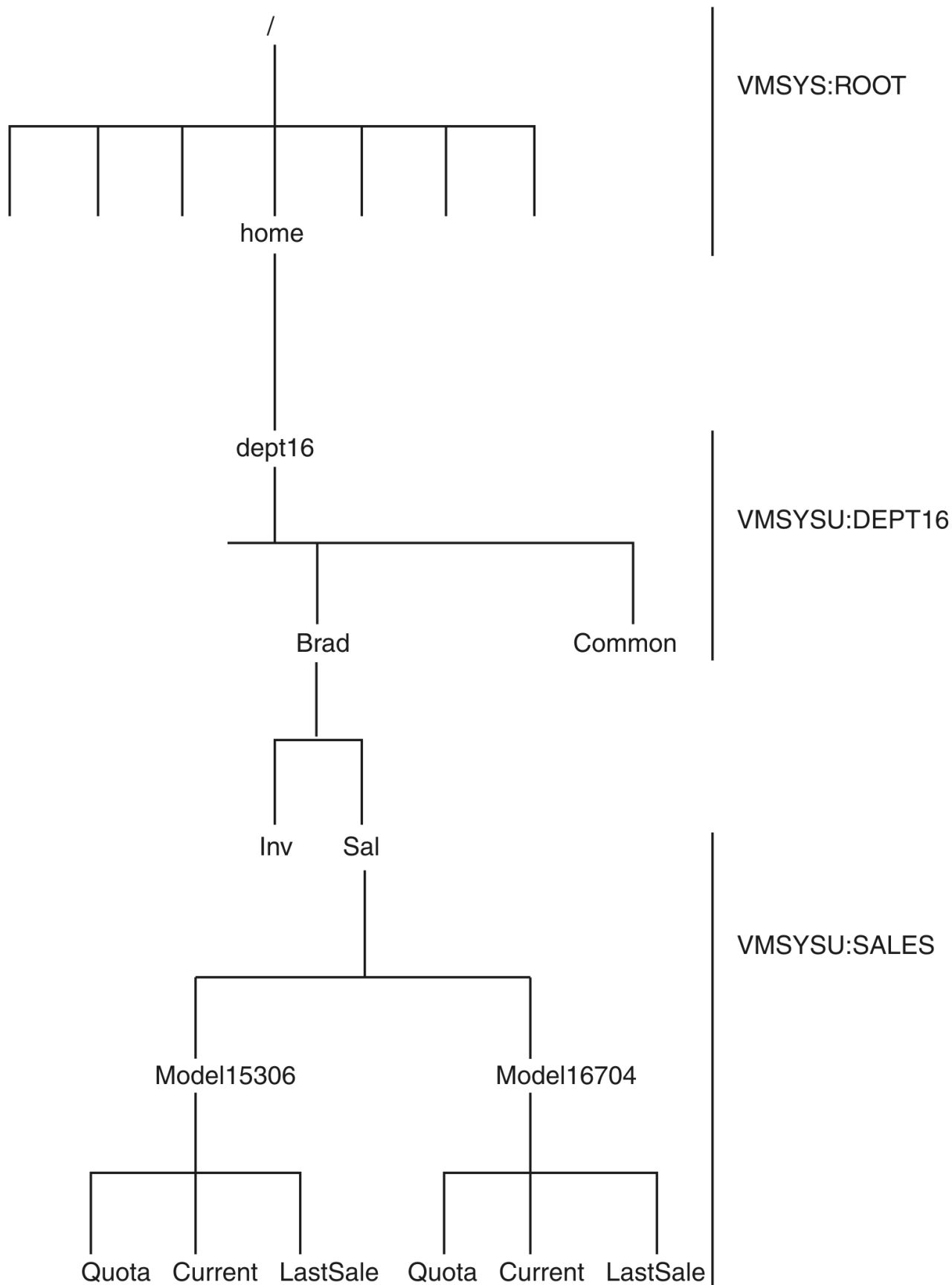


Figure 17. Example - the result of the Brad's commands



Even though Ed and Brad share the same file space, DEPT16, they may not be able to access the same objects in that file space. That is, Ed may not be permitted to access Brad's objects and Brad may not be permitted to access Ed's objects. In these examples, it is assumed they do not even see (have search permission to) each others objects. You could allow search (have the objects exist in the hierarchy), but restrict read (viewing their contents), for example. Permission can be controlled at any granularity level, from the BFS object to the BFS file space. See *z/VM: OpenExtensions User's Guide* for general information about setting up OpenExtensions. See [“Changing the Amount of Space Assigned to a File Space”](#) on page 91 for logical space management information and [Chapter 8, “Security,”](#) on page 137 for permission management.

After creating BFS file spaces (using the ENROLL USER command), you will probably want to use the OPENVM OWNER or OPENVM PERMIT commands to set up BFS clients to use the file spaces and objects you have created and set up for them.

In this illustration, mounts were explicit. You can also establish mounts through system or client profiles. The BFS root can be mounted through an entry in the z/VM system directory for a client.

## External Links of Type Mount (MEL)

Another method is to create external links of type mount (MEL) instead of establishing a directory mount point. When this is done the mount occurs implicitly when the client references the mount point (the mount external link object). Because there is a limit on how many systems you can have explicitly mounted, and to avoid having to issue a series of OPENVM MOUNT commands, the use of MEL is recommended. For example, assuming Ed had done mounts for the ROOT and DEPT16 file spaces, the following external link could be used instead of the Inv directory mount point in [Figure 16](#) on page 86.

```
OPENVM CREATE EXTLINK /home/dept16/Ed/InvExtLink MOUNT ../VMBFS:VMSYSU:INVENTORY/
```

With this we have the object InvExtLink (an external link object) in the place of the Inv directory object in Ed's hierarchy. When InvExtLink is referenced, the mount of the file space INVENTORY occurs and InvExtLink becomes the top of the hierarchy as [Figure 18](#) on page 90 shows:

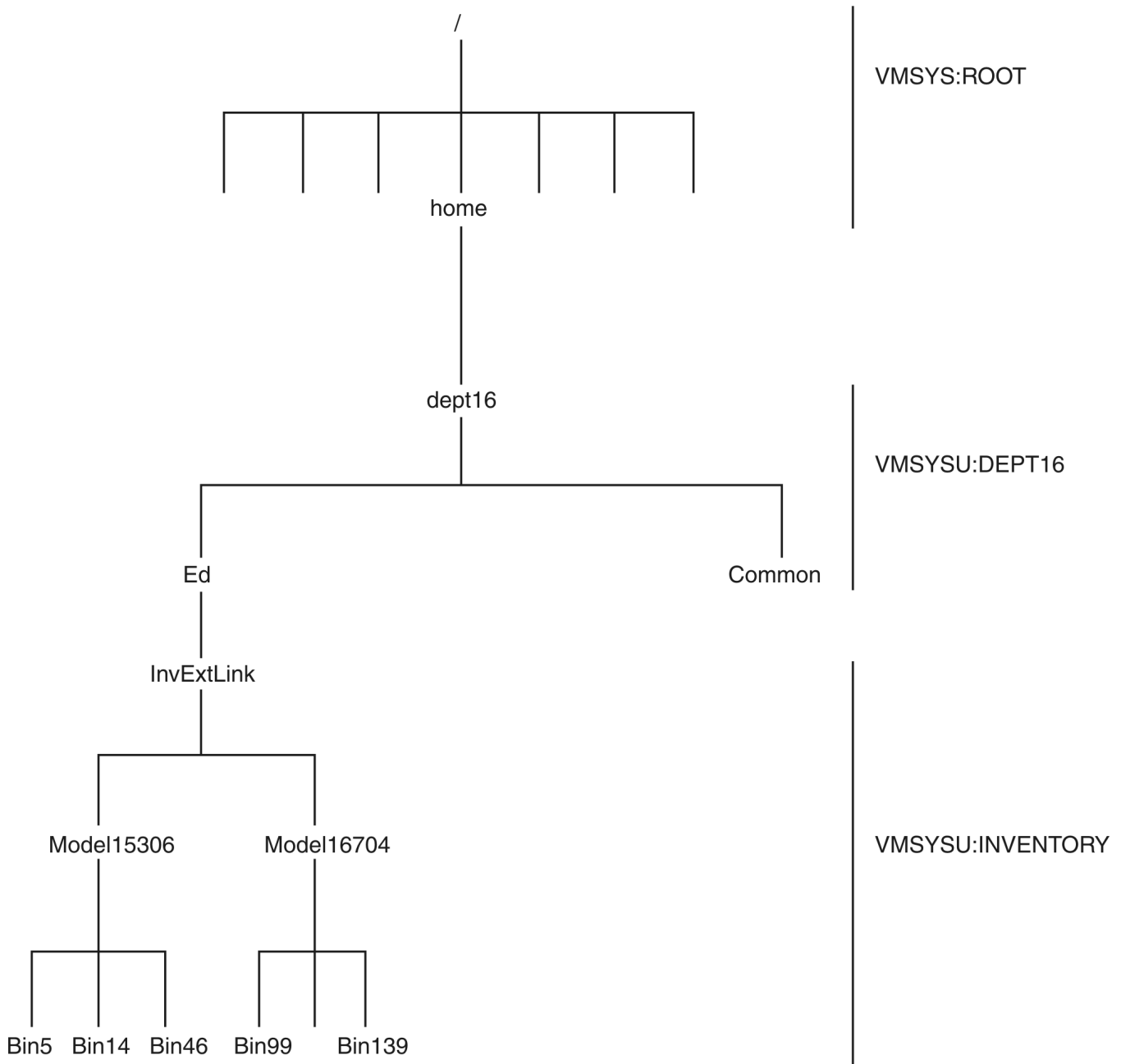


Figure 18. Example - External Link

The actual MEL should belong to the system administrator, but the directory it points to should belong to the user. Once a MEL is created, you need to override the default owner and establish permissions using the OPENVM PERMIT and OPENVM OWNER commands. See [z/VM: OpenExtensions Commands Reference](#) for more information about these commands.

To summarize, you assign space to your users by creating their own individual BFS file spaces. This is done with the ENROLL USER command. Then you join everything together by creating either a regular directory, or a MEL for each user in the /home directory of the designated system root BFS file space. These objects will be the users' home directories. If you choose to create a regular directory in /home for a given user, then whoever wants to access this user's data with that directory would first have to explicitly MOUNT the user's BFS file space onto the directory. If you choose to use MELs, you can access a user's data by simply referencing the MEL; the mounting is done under the covers. Either way, you should make sure the top directory of a user's private BFS file space has the correct UID or GID and associated permissions.

If you are using a regular directory as the home directory in `/home`, you must make sure the user has search authorization for it. Otherwise, the mount command will fail. For MELs this is not important because they work like symbolic links with forward path name resolution.

## Changing the Amount of Space Assigned to a File Space

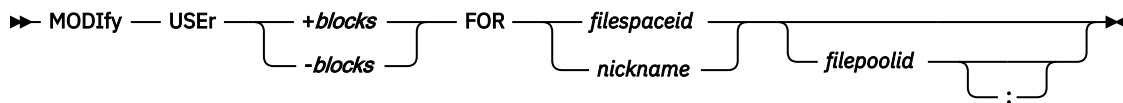
This procedure changes the maximum amount of file space that can be used by a file space in the file pool. All of the file space comes from the storage group to which the file space is currently assigned. This procedure does not allow you to change the storage group in which the file space resides (see [“Moving an SFS User or BFS File Space in Multiple User Mode”](#) on page 95).

To change the size of a SFS or BFS file space do the following:

1. Log on an administration machine if you are not already logged on to one. Ensure CMS is running in the administration machine. Also ensure the file pool is available in multiple user mode.
2. Decide how many 4KB blocks the file space is to have.
3. Check how much storage the person is currently using. Use [“QUERY LIMITS”](#) on page 655. For example, to check the amount of space user ID SUZY is using in the PROJECT1 file pool, enter:

```
query limits for suzy project1
```

4. If you are increasing the amount of space, you might want to check how much physical DASD space is still available in the storage group. Use [“QUERY FILEPOOL STORGRP”](#) on page 652.
5. Inform the user you will be changing their file space allocation. Ask the user to refrain from writing to his file space until you are done changing the space allocation. Also ask the user to prevent anyone with write authority from writing to the file space. (The user can send a message or create locks on the objects.) If those users ignore you, your MODIFY USER command can be unsuccessful.
6. Enter the MODIFY USER command. An abbreviated format of the command is:



See [“MODIFY USER”](#) on page 544 for a complete description.

To give file space Mary 1000 more blocks in the file pool named TEST1, for instance, enter:

```
modify user +1000 for mary test1
```

To reduce the allocation for file space Bob in TEST1 by 400 blocks, enter:

```
modify user -400 for bob test1
```

The file pool server deletes *unused* space from file space Bob. If file space Bob currently does not have at least 400 unused blocks, the command is unsuccessful. For example, suppose Bob had 1000 blocks originally. Over time, it used 700 of those blocks. It has 300 unused blocks. If you try to reduce the allocation for file space Bob by 400 blocks, as shown in the preceding example, the command is unsuccessful. The most you can reduce is 300 blocks.

If you need to increase the allocation of a group of SFS file spaces, try using the NAMES command to create a list of file spaces. Then enter a MODIFY USER command for the nickname. Suppose you define a nickname SPACE that identifies a group of file spaces that need a space allocation. To increase the allocation for those file spaces, each by 100 blocks, enter:

```
modify user +100 for space test1
```

If you are changing the allocation for Byte File System file spaces you may only change one file space at a time.

# Changing a User's Threshold Value (SFS Only)

An SFS administrator can change the threshold value of another user's file space using the SET THRESHOLD command with the FOR *userid* or FOR *nickname* option. For a complete command description, see “SET THRESHOLD” on page 664.

# Checking to See Who Is Enrolled (SFS) or Which File Spaces Exist

The QUERY ENROLL command shows you who is enrolled and which file spaces exist in a file pool. For instance, to see if user SMITH is enrolled in the PRODUC file pool, enter:

```
query enroll user for smith produc
```

To list all file spaces in a given file pool, and their types, enter:

```
query enroll filepace for all filepoolid
```

For more information about the QUERY ENROLL command, see [z/VM: CMS Commands and Utilities Reference](#).

# Determining Space Assigned to a File Space

To determine how much space a user or file space is allocated in a file pool, use the QUERY LIMITS command. For a complete command description, see “QUERY LIMITS” on page 655.

# Enrolling PUBLIC

You can enroll everyone on your z/VM system in a file pool. By doing so, you let any authorized z/VM user connect to the file pool. For SFS file spaces, this makes available any data in the file pool that is granted to PUBLIC. For BFS file spaces, PUBLIC enrollment does not imply any particular BFS permissions.

If your file pool is made available to a TSAF or CS collection as a global resource, then ENROLL PUBLIC has the effect of enrolling everyone in the TSAF or CS collection. If the z/VM system in which the file pool resides or the TSAF or CS collection is connected to the Systems Network Architecture (SNA) network, any authorized SNA APPC user can connect to the file pool.

To determine whether PUBLIC is enrolled in your file pool, enter the following command from any file pool user machine while the server is running in multiple user mode:

```
query enroll user for all filepoolid
```

If PUBLIC is enrolled, this will appear in the list of user IDs:

```
<PUBLIC>
```

The less than (<) and greater than (>) signs distinguish the keyword PUBLIC from a possible user ID of PUBLIC. If PUBLIC appears in the list without the less than and greater than signs, there is an enrolled user ID of PUBLIC.

To enroll PUBLIC in a particular file pool, do the following:

1. Log on an administration machine if you are not already logged on to one. Ensure CMS is running in the administration machine. The file pool must be available in multiple user mode.
2. Enter the ENROLL PUBLIC command. An abbreviated format of the command is:

```
►► ENRoll — PUBLIC —————►  
                  filepoolid  
                          :
```





pool. BRAD cannot use his file because it is locked, and DENNY cannot unlock the file because he cannot connect to the file pool. BRAD would have to call you to delete the lock.

You can delete the lock because you have administration authority. See [“DELETE LOCK” on page 403](#) for a complete description of the command.

## Overview of Moving Users and File Spaces

At times, you may want to move users to a different storage group. You might do this, for example, to balance storage group use by moving users to an under-used storage group. Or, for administrative reasons, you may want to have all the members of your department in the same storage group. You may also want to move users to a different file pool.

There are two ways to move users. You can use the FILESERV MOVEUSER command, or you can use the FILEPOOL UNLOAD command followed by the FILEPOOL RELOAD command.

FILEPOOL UNLOAD followed by FILEPOOL RELOAD will accomplish the move without the restrictions of the FILESERV MOVEUSER command, and will also allow you to move all file spaces in a storage group at the same time. For these instructions see [“Moving an SFS User or BFS File Space in Multiple User Mode” on page 95](#).

The FILESERV MOVEUSER command has the following restrictions:

- Multiple user mode processing must be stopped.
- DFSMS migrated files must be erased or recalled.
- Users may not be moved outside of the file pool.
- It can not be used for Byte File System (BFS) file spaces.

## Required Tasks Before Moving Users or File Spaces

1. Log on an administration machine if you are not already logged on to one. Ensure CMS is running in the administration machine. The file pool should also be available in multiple user mode.
2. Make sure there is enough physical storage in the *to* storage groups. Enter QUERY LIMITS commands to determine the number of 4KB blocks the users have committed. Then enter a QUERY FILEPOOL STORGRP command or use Query User Storage Group CSL request to determine whether the *to* storage groups have enough free space.

Suppose, for example, you are moving user MARY from storage group 11 to storage group 12 in the FINANCE file pool. To check user MARY's committed 4KB blocks in the FINANCE file pool, you would enter:

```
query limits for mary finance
```

Next you would enter a QUERY FILEPOOL STORGRP command to determine the amount of free physical space available in storage group 12. The 4K Blocks Free value should be greater than the number of committed 4KB blocks you plan to move to the storage group. If it is not, there is not enough space in the *to* storage group.

## Moving an SFS User or BFS File Space in Multiple User Mode

A method for moving SFS users, and the only way to move BFS file spaces, employs the FILEPOOL UNLOAD command in multiple user mode. The FILEPOOL UNLOAD command must be issued for the *filepaceid* or storage group containing the file spaces you want to move. To use the FILEPOOL UNLOAD command, you must be logged on to a machine that has administration authorization for the file pool to which you will be moving file spaces from. After the FILEPOOL UNLOAD command has successfully completed, the FILEPOOL RELOAD command must be issued. To use the FILEPOOL RELOAD command, you must be logged on to a machine that has administration authorization for the file pool to which you will be moving file spaces.

Follow these steps to move a user to a different storage group within the same file pool:



## Moving Users and File Spaces

1. Enter the FILEPOOL UNLOAD command for the *filespaceid* being moved. Be sure to use the *ALL* option on the FILEPOOL UNLOAD command, so you get all the authorizations granted to the *filespaceid* being unloaded, and the SFS aliases outside the file space being unloaded pointing to base files in the file space being unloaded.

For information, see [“FILEPOOL UNLOAD” on page 495](#).

2. Check to see if you have sufficient physical storage in the target storage group. See [“2” on page 95](#) for instructions.
3. Enter the FILEPOOL RELOAD command for the *filespaceid* being moved.

Follow these steps to move a file space to a different storage group in a different file pool:

1. Enter the FILEPOOL UNLOAD command for the *filespaceid* being moved. There is no need to use the *ALL* option when going to a different file pool. It will degrade performance, and the extra data unloaded will be ignored on the FILEPOOL RELOAD.

For information, see [“FILEPOOL UNLOAD” on page 495](#).

2. Enter the FILEPOOL RELOAD command for the *filespaceid* being moved.

For information, see [“FILEPOOL RELOAD” on page 476](#).

3. Enter a DELETE USER command to delete the user file space from the source file pool. This step is optional.

Follow these steps to move all file spaces in a storage group to a storage group in a different file pool:

1. Enter the FILEPOOL UNLOAD command for the storage group being moved. There is no need to use the *ALL* option when going to a different file pool. It will degrade performance, and the extra data unloaded will be ignored on the FILEPOOL RELOAD.

For information, see [“FILEPOOL UNLOAD” on page 495](#).

2. Enter the FILEPOOL RELOAD command for the storage group being moved.

For information, see [“FILEPOOL RELOAD” on page 476](#).

3. Enter a DELETE USER command to delete each file space from the storage group in the source file pool. This step is optional.

Follow these steps to move all file spaces in a storage group to a storage group in a the same file pool:

1. Enter the FILEPOOL UNLOAD command for the storage group being moved.
2. Enter the FILEPOOL RELOAD command for the storage group being moved.

## Moving an SFS User in Dedicated Maintenance Mode

Moving a user in dedicated maintenance mode involves stopping multiple user mode processing and making backups of the control data and affected user storage groups. Therefore, it is best to accumulate a list of SFS users that need to be moved (or will soon need to be moved) and move them all at the same time.

If you plan to move more than one user, sort the moves into clusters such that no *from* storage group ever becomes a *to* storage group in the same cluster of moves. The sorting is necessary so that backups can be made between clusters. The backups are needed to restore the file pool to a consistent state if a problem occurs while you are moving users. (Later steps tell you when backups are needed.) For an example of sorting SFS users into clusters, suppose you want to move the following users:

User ID	From	To
Cheryl	3	10
Bob	3	8
Tracy	3	11
James	4	9



User ID	From	To
Alfred	6	3
Mary	11	12

Cheryl, Bob, and Tracy, are all moving from storage group 3 to some other storage group, so they could all be in the same cluster. James can also be included because his *to* group is not group 3. Alfred must be in a separate cluster because his *to* group is 3. Mary can be in the same cluster as Alfred, but not in the cluster of Cheryl, Bob, Tracy, and James. Mary cannot be in this cluster because her *from* group is used as the *to* group in Tracy's move.

To recover from some errors while moving users, you will need to restore all the involved user storage groups. If you do not have a recent backup of one of the involved storage groups, it is recommended you backup the storage groups now, using the FILEPOOL UNLOAD or FILEPOOL BACKUP command. (Complete instructions are in [“Backing Up User Data”](#) on page 111.) Otherwise, an error could mean storage groups regress to an unacceptably earlier state.

To move a user and all of his data to a different storage group, do the following:

1. If the file pool is managed by DFSMS/VM, Mary may have files in DFSMS/VM migrated status. These are files that appear to reside in the file pool but actually reside in the DFSMS/VM storage repository. You must erase or recall these files before you can use FILESERV MOVEUSER to move her files to another storage group:
  - a. First, determine how many blocks these files are taking up by using the DFSMS REPORT command with the FILESPACE option.
  - b. Next, ensure you have enough space (free blocks) in Mary's current storage group to hold the files to be recalled. You can do this by issuing the QUERY FILEPOOL MINIDISK command and looking at the "Currently Defined Minidisk Information" values for the minidisks in Mary's storage group.
  - c. Use the ISMF file application (a DFSMS/VM facility) to list the files in migrated status you want to recall. (You must have explicit access to each directory in which the files reside. You must also have DFSMS/VM administration authority to do this step.)
  - d. Type RECALL on the command line to recall all the files on the list. (If you prefer, you can enter RECALL against individual files.)

See [z/VM: DFSMS/VM Storage Administration](#) for more information.

2. Log off (or disconnect from) the administration machine.

It is necessary to log off the administration machine because the remaining steps require you to stop multiple user mode processing and enter dedicated maintenance mode FILESERV commands. These steps must be done from the file pool server console (or its secondary user console).

3. Log on or reconnect to the file pool server machine as follows.

- If the file pool server is currently processing in multiple user mode, reconnect to it and enter a STOP operator command. Do not enter a STOP IMMEDIATE command. When the BACKUP startup parameter is in effect, the server will try to back up the control data. You should let this backup complete. The backup will be needed if something goes wrong during the following steps. (For more information about stopping multiple user mode processing, see [“Stopping Server Processing”](#) on page 65.)
- Otherwise, if the server is not currently processing in multiple user mode, log on it.

If you set up the server machine for automatic starting, as instructed in [Chapter 15, “Generating a File Pool and Server,”](#) on page 247, you have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC issues a FILESERV START command, which is not desired at this time.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

Next process the SETUP EXEC to access minidisks needed by the server machine (for information, see [SETUP EXEC](#)):

```
setup
```

If you forget to inhibit the execution of the PROFILE EXEC and the file pool server successfully starts, stop the server by entering:

```
stop
```

Do not enter a STOP IMMEDIATE command. When the BACKUP startup parameter is in effect, the server will try to back up the control data. You should let this backup complete. The backup will be needed if something goes wrong during the following steps.

4. Now move the user. If you are moving clusters of users, repeat the following for each cluster.

- Enter the FILESERV MOVEUSER command for every user in the cluster being moved. If you are moving only one user, just enter the command one time. FILESERV MOVEUSER has the following format:

```
➤ FILESERV — MOVEUSER — userid — group_number ➤
```

For *userid*, specify the user ID for the user to be moved. A nickname cannot be used.

For *group\_number*, specify the number of the storage group to which the user is being moved. User data cannot be moved to the catalog storage group (storage group 1). See [“FILESERV MOVEUSER”](#) on page 528 for a complete description of the command.

When the move is complete, all the user’s directories, base files, aliases, and authorizations have been transferred to the new storage group. Except for the storage group number, the change is transparent to the user. You should, however, inform the user what his new storage group number is.

If you are moving a cluster of users, enter the FILESERV MOVEUSER command for each user in the cluster. Continuing the example already begun, suppose you were moving Cheryl, Bob, Tracy, and James. You would enter:

```
fileserv moveuser cheryl 10
fileserv moveuser bob 8
fileserv moveuser tracy 11
fileserv moveuser james 9
```

Because Alfred and Mary are not in this cluster, you would not move them at this time. It is important to properly sort and move users.

**Note:** If you move SFS users such that a *from* group is also a *to* group in the same cluster of FILESERV MOVEUSER commands, you risk having a unrecoverable file pool.

If a FILESERV MOVEUSER command is unsuccessful, correct any indicated error condition and rerun it.

If an error occurs such that the control data needs to be restored, restore the control data and then repeat the entire sequence of FILESERV MOVEUSER commands for the cluster of users—even those that completed successfully prior to the loss of the control data.

If an error occurs such that one or more user storage groups need to be restored, restore the control data, then restore the storage groups from backups made prior to the moves. When the restores are complete, the file pool will be in the state it was in before any of the SFS users were moved.

- Back up the control data. The execution of a FILESERV MOVEUSER command makes the previous control data backup obsolete. Enter a FILESERV BACKUP command to make a new backup of the control data:

```
fileserv backup
```

See [“Backing Up The Control Data”](#) on page 104 for more information about backing up the control data.

- Back up all *from* and *to* storage groups. This will give you a set of backups that match the current state of the file pool. User storage group backups prior to the move are obsolete. If you use these obsolete backups on a subsequent restore it is possible the FILESERV MOVEUSER will be reversed or the moved user is deleted from the file pool. (See the usage note concerning "Obsolete Backup Files" listed under the description of the command "[FILEPOOL RESTORE](#)" on page 488. See "[Backing Up User Data](#)" on page 111 for instructions on backing up user storage groups.

While these backups are being made, general user access to the file pool should be restricted. See "[Restricting User Access in Multiple User Mode](#)" on page 69 for instructions.

- Repeat these steps for every cluster. Do not omit the backups between clusters.
5. After moving all the clusters of users, start multiple user mode processing by entering a FILESERV START command.

```
fileserv start
```

6. Inform SFS users what their new storage group numbers are.

## Renaming File Spaces

---

Renaming a file space transfers all the file pool objects, SFS authorizations, and locks of one file space to a new file space. For SFS, it is necessary to rename a file space when an SFS user's CP user ID changes. For example, if Mary's user ID changes from MARYLAMB to MCNTRARY, for each file pool that MARYLAMB is authorized to use, you have to rename MARYLAMB to MCNTRARY.

When MARYLAMB has a file space in the file pool, it is renamed to MCNTRARY. When MARYLAMB is not enrolled in the file pool, only explicit authorizations or current locks are transferred to MCNTRARY.

You can also rename BFS file spaces using the FILEPOOL RENAME command. However, there may not be an association with a user ID change because BFS file space names need not be associated directly with a user ID.

You rename a file space with the FILEPOOL RENAME command. For details, see "[FILEPOOL RENAME](#)" on [page 484](#).



---

## Chapter 7. Recovery Procedures

This chapter contains the procedures to use to prevent file pool data from being lost in case of an unexpected error. This is done by taking backups of the data in the file pool periodically so the data can be restored if necessary.

The procedures show the steps for backing up and restoring the two types of data in a file pool: user data and control data. Following is a list of the recovery-related topics discussed in this chapter:

- Control Data
  - Backing up control data
  - Restoring control data
- User Data
  - Backing Up User Data
  - Restoring User Data
- Restoring a file pool by generating it again—to be used as a last resort
- Replacing file pool minidisks
- Replacing both file pool log minidisks
- Using non-file pool server facilities for backing up your file pool

---

### Backing Up a File Pool - Overview

Although a repository file pool is a single logical entity, for the purposes of the backup and restore discussions, it is broken up into two parts. The two parts are:

- user data
- control data

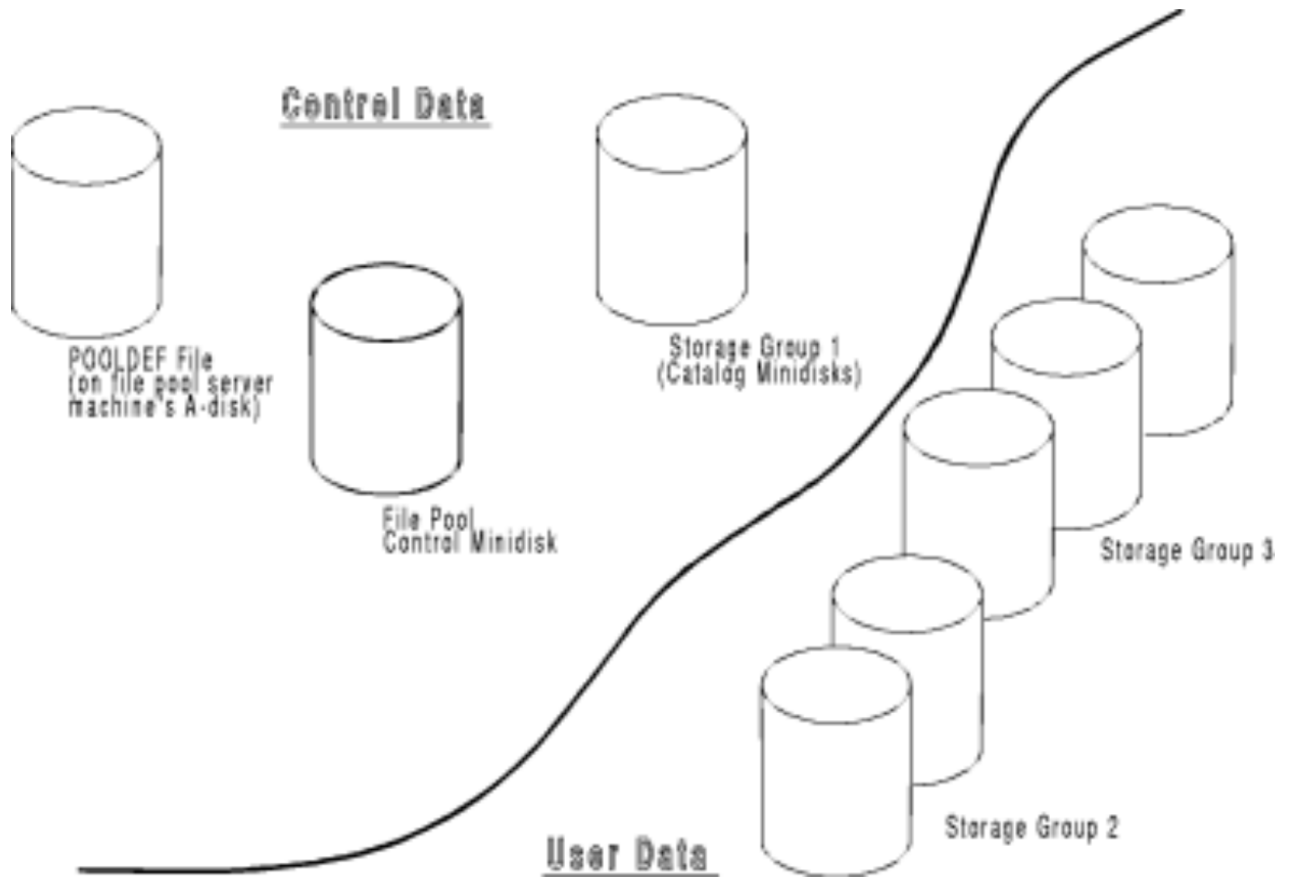


Figure 19. File Pool Control Data and User Data

The “repository” or “user data” is the data stored in storage groups 2 through  $n$ . Each of these storage groups is backed up individually.

The control data is all the other data in the file pool: the control disk, the catalogs (storage group 1), the logs, and the POOLDEF file. The repository logs are not backed up but are required for restore. Because of the dependence on the repository logs, only the latest control data backup is usable for restore.

It is recommended both user data and control data backups be taken. If you decide to take user data backups and not control data backups, and your control disk or a storage group 1 disk gets corrupted, you will be forced to generate your file pool and restore every user storage group. This will regress all the file pool data, and could be very time consuming. If a control data backup was available, only the control data would need to be restored, and the file pool would be restored to the point of error.

If you decide to back up control data and not user data, you will lose your entire storage group if one minidisk in a user storage group gets corrupted.

The other problem you may run into is if you decide to use DDR or some other non-file pool server backup facility to back up repository data. Do not do this without reading the section [“Using Non-File Pool Facilities for Backing Up Your File Pool”](#) on page 134 first.

The following tables identify the differences between control data and user data in relation to backup and restore.

<i>Table 8. Differences between User Data Backup and Control Data Backup</i>	
<b>User Data Backup</b>	<b>Control Data Backup</b>
Use: <ul style="list-style-type: none"> <li>• FILEPOOL BACKUP administrator command</li> <li>• FILEPOOL UNLOAD administrator command</li> </ul>	Use one of the following control data backup commands: <ul style="list-style-type: none"> <li>• FILEPOOL CONTROL BACKUP administrator command</li> <li>• BACKUP operator command</li> <li>• STOP BACKUP operator command</li> <li>• FILESERV BACKUP command</li> </ul> Or <ul style="list-style-type: none"> <li>• Allow automatic backups</li> </ul>
Backs up user storage groups (2..N) or, if using FILEPOOL UNLOAD, may also back up files spaces.	Backs up catalog storage group (1), Control Disk, and POOLDEF file.
<ul style="list-style-type: none"> <li>• Storage group being backed up restricted to read mode only during backup.</li> <li>• File space being backed up by FILEPOOL UNLOAD restricted to read mode only during backup.</li> </ul>	No restrictions on data.
BACKUP in the DMSPARMS file not applicable.	BACKUP must be specified in DMSPARMS file.
Backup file defined using FILEDEF command.	Backup file defined in POOLDEF file, or as defined on a control data backup command.
Backup file may reside on tape, disk, or SFS file.	Backup file may reside on tape, disk, or SFS file.
Backup runs in administrator machine.	Backup runs in server machine.
FILEPOOL LIST MINIDISK command lists files that have blocks located on a specific minidisk. The listed files can be used as input to FILEPOOL FILELOAD or FILEPOOL RELOAD FILES when a single minidisk is damaged.	Not applicable.
FILEPOOL LIST BACKUP command lists contents of the storage group backup file. The listed files can be used as input to FILEPOOL FILELOAD or FILEPOOL RELOAD FILES when a single minidisk is damaged.	Not applicable.

<i>Table 9. Differences between User Data Restore and Control Data Restore</i>	
<b>User Data Restore</b>	<b>Control Data Restore</b>
Use: <ul style="list-style-type: none"> <li>• FILEPOOL RESTORE for storage group backed up by FILEPOOL BACKUP</li> <li>• FILEPOOL RELOAD for storage group or file spaces backed up by FILEPOOL UNLOAD</li> <li>• FILEPOOL FILELOAD for individual files from file created by FILEPOOL BACKUP</li> <li>• FILEPOOL RELOAD FILES for individual files from file created by FILEPOOL UNLOAD</li> </ul>	Use: <ul style="list-style-type: none"> <li>• FILESERV START command with RESTORE in DMSPARMS file</li> </ul>
Brings the data back to the level of the backup. The repository logs are not relevant.	Brings control data back to the level at the point of error. Uses the repository logs to achieve this.

User Data Restore	Control Data Restore
Frequency of backup affects how current the level is of restored directory structures and files.	Frequency of backup has no affect on the level of the restored data. It is always current.
Any valid backup file may be used for restore.	Only the last successful backup file is useful for restore.
Frequency of backup does not affect time taken for restore.	Time taken for restore increases with amount of log data written between backups.

## Backing Up The Control Data

In order to back up the control data, you first need to enable control data backup processing, and then decide which method of control data backup to use. The following sections will help with this.

### Enabling Control Data Backup Processing

If you are currently running with NOBACKUP in the DMSPARMS file and you want to switch to BACKUP so you can start using facilities for backup of the control data, follow these steps:

1. Consider increasing the size of the repository log minidisks. See [“Step 3B: Determine the File Pool Repository & CRR Log Minidisk Allocations”](#) on page 251 for more information.
2. Log on the file pool server machine.
3. Stop server processing by entering the STOP operator command.
4. Enter the FILESERV DEFBACKUP command to define an output file for subsequent control data backups. See [“Defining the Control Data Backup File with the FILESERV DEFBACKUP Command”](#) on page 108 for instructions.
5. Edit the *serverid* DMSPARMS file. Delete the NOBACKUP startup parameter. (BACKUP is the default, so there is no need to specify it unless you choose to do so.)
6. Enter the FILESERV BACKUP command to back up the control data. See [“FILESERV BACKUP”](#) on page 500 for specific directions.
7. Enter the FILESERV START command to resume server processing.

Now the control data may be backed up using any of the methods provided. See the sections following for details.

### Control Data Backups at Shutdown

After your file pool is enabled for control data backups, the server will automatically back up the control data whenever you enter the STOP operator command. The advantage of taking a backup at shutdown is the control data is inspected prior to backing it up. Control data damage would be exposed, and a STOP NOBACKUP could be performed so the current backup would not get destroyed.

See [“STOP”](#) on page 666 for more detailed information on how to start control data backup using the STOP command.

### Control Data Backups Using FILESERV BACKUP

You can also back up the control data by entering the FILESERV BACKUP dedicated maintenance mode command. It is often necessary to use the FILESERV BACKUP command after you enter some other dedicated maintenance mode command that significantly changes the control data (such as FILESERV MINIDISK). This document will tell you when such backups are required, and you will receive messages indicating you must enter the FILESERV BACKUP command before issuing FILESERV START. See [“FILESERV BACKUP”](#) on page 500 for more detailed information on how to start control data backup using the FILESERV BACKUP command.



## Control Data Backups Using the BACKUP Operator Command

When you enter a BACKUP operator command, the file pool server starts backing up the control data while it continues processing user requests. See [“BACKUP” on page 366](#) for more detailed information on how to start control data backup using the BACKUP command.

## Control Data Backups Using FILEPOOL CONTROL BACKUP

The FILEPOOL CONTROL BACKUP command allows the administrator to perform control data backups without having to log onto the file pool server machine. This command causes a backup to be started. The file pool server starts backing up the control data while it continues processing user requests. The user will get notified on return from the command whether a backup was successfully started. The user will receive notification in a reader file named \$\$\$SFS \$MSG\$ stating whether the backup was successfully completed or not. See [“FILEPOOL CONTROL BACKUP” on page 439](#) for more detailed information on how to start control data backup using the FILEPOOL CONTROL BACKUP command.

## Control Data Backups Started Automatically

File pool server processing automatically backs up the control data in multiple user mode when the repository log minidisks are 80% full. The backup process frees log space. During the backup, the server continues to process user requests, and, consequently, continues to fill the log. If the log is 95% full and the server has not yet completed the backup, the server suspends the processing of user requests to avoid filling the logs. In rare cases, however, the server may not be able to suspend all user requests in enough time to keep the logs from filling. If the log does fill, the server stops. In that case, you should enter a FILESERV BACKUP command and then resume multiple user mode processing.

Although automatic backups allow the server to continue processing, they have the disadvantage of being unscheduled. They may start during times of peak activity. If a problem occurs such as a tape not mounted, or the minidisk the backup being written to becomes full, an error message will be issued. Another backup will be scheduled after every 1% of log space is used. If the problem is not resolved by the time the logs become full, the server will stop.

### LUW Rollback During Automatic Backup

Immediately before starting an automatic backup, the server may roll back some long-running logical units of work. The logical units of work the server rolls back are those that were active at the time the *last* backup was made. That is, the logical unit of work was active during the entire time it took to fill 80% of the log and is still active. The server rolls back these long-running logical units of work because they prevent log space from being freed. The application whose logical unit of work is rolled back will receive an error return code. Given a reasonably sized log, logical units of work that span backups often indicate improperly coded applications.

## DASD Space Needed for Control Data Backup

For control data backups, you will need enough space to hold the POOLDEF file, the entire file pool control minidisk, and the used blocks in the storage group 1 minidisks. (You can use the QUERY FILEPOOL MINIDISK command to gather this information.) If you are overwriting the previous backup file, twice as much space is needed because the current control data backup file is preserved until the new one is completely written.

## Special Considerations for Control Data Backups to File Pool

Directing control data backups to an SFS directory is advantageous because it is less disruptive to other users of the file pool. Also, you would not need to worry that the file would exceed a physical device limitation. You do need to ensure there is enough unused DASD assigned to the storage group in which the backup file is to reside. You also need to ensure there is an adequate number of uncommitted 4KB blocks in the file space of the backup file.

## Backing Up the Control Data

To verify there is a sufficient number of free 4KB blocks in the file space that will hold the control data backup file, use the QUERY LIMITS command for the target file space.

To check how much physical DASD space is available in the storage group that is to hold the backup file, enter a QUERY FILEPOOL STORGRP command for that file pool. Note that you would periodically need to re-check the physical storage. The amount of the control data being backed up could increase or other storage group users could consume the free DASD space in that storage group.

## Special Considerations for Control Data Backups To Tape

When your control data backups are directed to a tape file, you should ensure the most current backup tape(s) are not overwritten when you make a new backup. You should use a separate set of tape volumes for the new backup. If you reuse the same tape volumes, you will get message DMS3222E, and backup processing will stop.

Only after the next backup completes successfully can you reuse the tape volumes containing the old backup. You might, for example, maintain two sets of tape volumes and alternate their use for control data backups.

You should also ensure someone is available to mount tapes unless you have a tape management system set up to mount tapes. When multiple volumes are needed, the server will be in a wait state until the second volume is mounted.

For control data backup to tape, the file pool server always uses IBM standard labels. IBM standard labels are required for multivolume tape support.

## Special Considerations for Control Data Backups to CMS File

Because only the most current backup file is valid, the server is designed to overwrite any existing backup file (identified by DDNAME=BACKUP) when the backup is directed to DASD. Suppose, for example, your backup file is CONTROL BACKUP A. During server processing, you make three online backups by entering BACKUP operator commands. As each backup completes successfully, the “old” (and now obsolete) backup file is overwritten with new data. To ensure there is always a valid control data backup, the server must be certain the new backup completes successfully before it destroys the data from the old backup. The server does not immediately write over the old backup file. If it did and an error occurred before the backup completed successfully, there would be no valid control data backup. To avoid the chance of having an incorrect minidisk backup file, the server writes the backup to a temporary file named \$\$TEMP \$BACKUP. It writes \$\$TEMP \$BACKUP to the same file mode on which the file identified by DDNAME=BACKUP resides. When the backup completes successfully, the server erases the file identified by DDNAME=BACKUP and renames the \$\$TEMP \$BACKUP file to that name. If a error were to occur, a valid backup would be in either the file identified by DDNAME=BACKUP or in the \$\$TEMP \$BACKUP file.

If a backup does not complete successfully, you may need to take some recovery action, as follows:

- If the error occurs **before** message DMS3294I is displayed, you do not need to do anything. The valid backup file is the file identified by DDNAME=BACKUP. The backup in \$\$TEMP \$BACKUP is incomplete. You can erase \$\$TEMP \$BACKUP if you wish. An example of message DMS3294I is:

```
mm/dd/yy 16:32:03 File pool control data BACKUP complete
```

- If a system error occurs immediately **after** message DMS3294I is displayed, determine whether the file \$\$TEMP \$BACKUP exists on the same file mode on which the DDNAME=BACKUP file resides. If it does not exist, the backup file identified by DDNAME=BACKUP is valid. You do not need to do anything else.

If \$\$TEMP \$BACKUP does exist, it contains the valid backup. In this case, you should erase the file identified by DDNAME=BACKUP if there is one, and rename \$\$TEMP \$BACKUP to that name.

## Frequency of Control Data Backup

There are three factors to consider in determining the frequency of control data backups:

1. Control data restore time increases with the amount of work done since the last control data backup

2. Control data backup causes user response time degradation
3. File pool log minidisk size must be increased if frequent automatic backups are to be avoided.

Use the QUERY FILEPOOL LOG command to monitor file pool log consumption. When the log becomes 80% full, the file pool server initiates a backup. You should start a backup before the log becomes 80% full to avoid this automatic backup.

## Disabling Control Data Backup Processing

If you are running with BACKUP in the DMSPARMS file, and you do not want to use SFS facilities for backing up the control data, follow these steps:

1. Log on (or reconnect to) the file pool server machine.
2. Force the file pool server to apply the changes recorded in the logs.

**Do not skip this step.** To apply all changes in the logs to the file pool, you must allow multiple user mode processing to stop normally:

- If the file pool server is now running in multiple user mode, enter a STOP NOBACKUP operator command. (STOP NOBACKUP is recommended because any backup made now would never be used for restoring the control data.) Do **not** use a STOP IMMEDIATE command.
- If the file pool server is not running in multiple user mode, enter a FILESERV START command followed by a STOP NOBACKUP command.

3. Update the DMSPARMS file:

- Change the BACKUP startup parameter in the DMSPARMS file to NOBACKUP.
- Specify the FORMAT startup parameter unless your log disks are already formatted and reserved as SFS log minidisks.

4. Optionally, remove the backup file definition from the POOLDEF file by entering:

```
fileserv defbackup delete
```

5. If you are permanently disabling control data backup processing, consider reducing the size of your logs to save DASD space. See [“File Pool Log Size When File Pool Backup Is Not Used”](#) on page 254 for details.
6. Determine the virtual device numbers of the file pool log minidisks.

To find the numbers, edit the *filepoolid* POOLDEF file and look at the DDNAME=LOG1 and DDNAME=LOG2 records. In this example, the virtual device number of LOG1 is 302, while the number of LOG2 is 303:

```
⋮
DDNAME=LOG1      VDEV=302
DDNAME=LOG2      VDEV=303
⋮
```

7. Enter the FILESERV LOG command for the virtual device numbers determined in the last step:

```
fileserv log 302 303
```

See [“FILESERV LOG”](#) on page 523 for more on the FILESERV LOG command.

8. Enter the FILESERV START command to resume usual operations (for detailed instructions, see [“FILESERV START”](#) on page 537).

## Defining or Changing the Default Control Data Backup File

The default destination of the control data backups can be defined or changed in one of the following two ways:

- If the server is not currently processing in multiple user mode, the FILESERV DEFBACKUP command can be used to establish a new, or change the existing, permanent control data backup file definition

in the POOLDEF file. This definition will be used by the server to establish the control data backup environment every time the server is started in multiple user mode.

- If the server is currently processing in multiple user mode, the DEFBACKUP server operator command can be used to establish a new, or change the existing, control data backup file destination. This definition will be in effect until multiple user mode ends, or another DEFBACKUP command is issued. Note that the POOLDEF file is not changed.

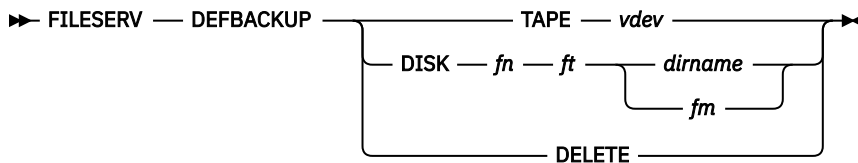
In addition, you can temporarily redirect the backup file by explicitly specifying the destination on the BACKUP operator command, the FILESERV BACKUP command, the STOP BACKUP operator command, or the FILEPOOL CONTROL BACKUP administrator command. You may want to do this if, for example:

- Your default backup destination is tape and an operator is not available to mount the tape.
- Your default backup destination is another file pool, but the file pool is not available for some reason.

### Defining the Control Data Backup File with the FILESERV DEFBACKUP Command

If you want to define or change the permanent default control data backup file destination in the POOLDEF file, follow these steps when the file pool server is not running.

1. Decide where you want to direct the backups.
2. Enter the FILESERV DEFBACKUP dedicated maintenance mode command. When you enter FILESERV DEFBACKUP, the file pool server updates the POOLDEF file to indicate the new backup file. The format of the command is:



See “[FILESERV DEFBACKUP](#)” on page 506 for a complete description of the command.

If, for example, you want to back up the control data to the tape device at virtual address 182, you would enter:

```
fileserv defbackup tape 182
```

If you wanted to direct the backups to a CMS file named BACKUP DATA B, you would enter:

```
fileserv defbackup disk backup data b
```

Here, b represents a standard CMS minidisk or an SFS directory that will have to be accessed in read/write mode before the server is started. If you are directing the backups to another file pool, you can also enter:

```
fileserv defbackup disk backup data fpname:username.dir
```

When the directory is explicitly specified, it does not have to be accessed.

If you are discontinuing the use of file pool control data backup facilities, you can use the DELETE option to remove any reference to a backup file:

```
fileserv defbackup delete
```

3. Enter the FILESERV START command to resume multiple user mode processing:

```
fileserv start
```

**Note:** The DELETE parameter on the FILESERV DEFBACKUP command causes backup file related entries in the POOLDEF file to be deleted.

When running in multiple user mode, the validity of the default backup file destination is checked when you start the server and after each backup. If something is wrong, you may see warning messages telling you what the problem is so you can correct the problem or change the destination. If you do not take action, the next backup will not work.

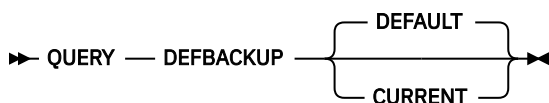
## Defining the Control Data Backup File with the DEFBACKUP Operator Command

When running in multiple user mode, you can change the default destination without stopping the server by issuing the DEFBACKUP operator command. The DEFBACKUP operator command will change the default destination for all subsequent control data backups while the server is running, but it will not change the POOLDEF file. The next time you start the server, the old destination that was recorded in the POOLDEF file with the FILESERV DEFBACKUP command will become the default destination again. You may want to use the DEFBACKUP operator command if, for example:

- You discover the minidisk you originally planned to use for backups is not large enough.
- You have specified a file mode representing an accessed directory in another file pool, but because communications to that other file pool were disrupted, that directory is not accessed anymore.

For a complete description of this operator command, see [“DEFBACKUP” on page 398](#).

If you need to determine either the default destination for the control data backup file or where the last successful backup file was created, use the QUERY DEFBACKUP operator command: The format of the command is:



If backup was started automatically (because the log was over 80% full) and it did not complete, the default control data backup file destination is invalidated and you will have to specify the backup file destination explicitly on the BACKUP or STOP BACKUP operator command. Depending on what the problem is, you could either correct it and specify the same destination, or you could try to redirect the control data file (for example, from minidisk to tape). In any event, if the problem is not corrected by the time the logs become 95% full, the server will stop. If this happens, execute FILESERV BACKUP and start the server again.

If the default control data backup file destination was invalidated, you can establish a new default destination with the help of the DEFBACKUP operator command.

## Recovery from Control Minidisk Verification Errors

Prior to making control data backups at shutdown or in dedicated maintenance mode (using FILESERV BACKUP), the file pool server verifies the contents of the control minidisk. During this process, known as *control minidisk verification*, the server looks for data inconsistencies. These inconsistencies, should they exist, would make the control data backup useless. A subsequent restore using that backup would not be successful. Therefore, when an inconsistency is detected, the server does not back up the control data. Instead, it displays messages (DMS3216E and DMS3217E) describing the inconsistency and stops.

Because of its extensive use of the control minidisk, control minidisk verification is not done for control data backups started in multiple user mode. These backups include those started automatically as well as those started by the BACKUP operator command or FILEPOOL CONTROL BACKUP administrator command.

Control minidisk inconsistencies must be corrected. If they are not and users continue to use the file pool, the results are not predictable. To recover from a control minidisk inconsistency, follow these steps:

1. Restore the control data.

When restoring the control data, ensure no users are allowed to use the file pool in multiple user mode until the restore completes.

## Restoring Control Data

In restoring the control data, the server copies the contents of the backup to the file pool, then applies the changes recorded in the log. Because the processing involved in applying the changes recorded in the log is not the same as the processing that originally caused the inconsistency, the inconsistency is frequently "fixed" by the restore. This assumes the discrepancy was introduced *after* the current control data backup was made.

If the control data backup was made using the FILESERV BACKUP command or at shutdown, you can be sure the discrepancy was introduced after the backup—file server processing would not have made the backup if the same discrepancy existed at that time. If, on the other hand, the backup was made during multiple user mode processing, the inconsistency might exist in the backup.

If the restore completes successfully, continue to the next step. Otherwise, take the recovery actions as indicated in the error messages displayed.

2. Stop server processing using the STOP BACKUP operator command.

The server again attempts to verify the control minidisk and back up the control data. If the control minidisk verification succeeds and a backup is made, all is well—restore processing removed the inconsistency. You can skip the remaining steps and restart the server for multiple user mode access.

If the discrepancy still exists, continue with the next step.

3. Generate the file pool again.

Next, try generating the file pool again and restoring all the storage groups that contain user data. When the file pool is generated again, the control data is reset to be logically *empty*. Restoring the user storage groups should remove the inconsistency. Follow the instructions in [“Restoring a File Pool by Generating It Again”](#) on page 127.

If the backup made at the end of the process succeeds, all is well. If the inconsistency still exists, call the designated support group for your installation.

## Restoring Control Data

---

If the data lost was the POOLDEF file, control minidisk, or any minidisk in the catalog storage group (storage group 1), you will have to do a restore of the file pool control data.

If your data loss affects more than one item, restore the logs first, then the control data, and, finally, the user data.

To restore the file pool control data, you must have the current backup of it. If you do not have a current backup, or if the backup itself is unusable (for example, the tape breaks), follow the instructions in [“Restoring a File Pool by Generating It Again”](#) on page 127.

Assuming you have a valid backup of the control data, do the following:

1. Log on or reconnect to the server machine. If the server is currently processing in multiple user mode, reconnect to it and enter STOP NOBACKUP. You do not want a control data backup started if you think your control data may be corrupt.
2. Add the RESTORE startup parameter to the DMSPARMS file for the server machine. (For more information about specifying startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.)

For best performance when restoring the control data, you should also specify a very large catalog buffer pool. To increase the size of the catalog buffer pool, specify the CATBUFFERS startup parameter. For example, you might enter:

```
CATBUFFERS 5000
```

3. Enter a FILEDEF command to associate the backup file with the ddname RESTORE. If, for example, your restore file is on tape, you might enter:

```
filedef restore tap1 sl valid nnnnn
```

You must use SL on the FILEDEF command to indicate IBM standard labels.

If, on the other hand, your backup file is on DASD, you will first have to access the minidisk or the directory. For example, you might enter:

```
access sysadmin:vmsysu.bkup b
filedef restore disk backup file b
```

The file pool server uses a block size of 28672. If you specify a block size (BLKSIZE) in the FILEDEF command, it is ignored. The file identified by ddname RESTORE should be the same file identified by ddname BACKUP during a previous control data back up.

4. Enter the FILESERV START command.

A message will ask whether you really want to restore the control data. If all is well, reply 1 for yes.

The file pool server will restore the control data from the backup file associated with ddname RESTORE. When the restore is complete, the server applies all the changes recorded on the repository log minidisks, which brings the file pool to the state it was in at the time of the error. Then the file pool server allows regular multiple user access.

When the restore completes, multiple user mode access begins. If you would prefer to inhibit multiple user mode access until you have had a chance to verify that all went well, see [“Restricting User Access in Multiple User Mode”](#) on page 69 before entering the FILESERV START command.

5. Remove the RESTORE startup parameter from the DMSPARMS file at the earliest opportunity. If you specified the CATBUFFERS startup parameter, you should stop multiple user mode, reset CATBUFFERS to its regular value (or allow it to default), and then resume multiple user mode.

## What If a Control Data Restore Is Unsuccessful?

During the restore, the file pool server creates a temporary file named \$\$\$AVED \$POOLDEF using the same file mode as the POOLDEF file.

If server processing ends unusually during a restore, you may need to manually restore the POOLDEF file, as follows:

1. Determine whether a file named \$\$\$AVED \$POOLDEF exists on the same file mode as the POOLDEF file.
2. If no such file exists, you can retry the restore.
3. If the file exists, erase the POOLDEF file and rename \$\$\$AVED \$POOLDEF to it, and retry the restore. If, for instance, the POOLDEF file was named VMSYSU POOLDEF, you would enter:

```
erase vmsysu pooldef a
rename $$$aved $pooldef a vmsysu pooldef a
```

## Restoring Repository Log Data

If the data loss was caused by a media error or it is otherwise necessary for you to replace SFS log minidisks, follow the instructions in [“Replacing File Pool Minidisks”](#) on page 129.

If you do not have even one good log, see [“Replacing Both File Pool Log Minidisks”](#) on page 133.

If you have one good log, and one log that is corrupted but does not need replacing, FILESERV START processing will copy the good log to the corrupted log. No other actions are necessary.

## Backing Up User Data

There are two FILEPOOL commands that may be used for backing up user data. They are FILEPOOL BACKUP and FILEPOOL UNLOAD. FILEPOOL BACKUP lets you back up a user storage group. FILEPOOL UNLOAD lets you back up either a user storage group or a file space. The main difference in the two in terms of backup and restore is that FILEPOOL RELOAD, which will restore data created by FILEPOOL UNLOAD, does not require the minidisk configuration of the storage group at the time of the FILEPOOL RELOAD be identical to the minidisk configuration at the time of the FILEPOOL UNLOAD. FILEPOOL RESTORE does have this restriction.



If you have storage groups containing Byte File System (BFS) data, it is important to note there is a restriction when using FILEPOOL FILELOAD you must identify the files to be restored by the CMS short name. (The CMS short name identifies the CMS file name and file type of an object in the backup file that is unique within a BFS to each file.) The file must exist prior to the FILEPOOL FILELOAD. When using FILEPOOL RELOAD FILES for byte file data, you may identify the files to be restored by either the CMS short name or the fully qualified pathname, and the file must exist prior to the FILEPOOL RELOAD FILES only if the CMS short name is specified.

## Using FILEPOOL UNLOAD

To back up a user storage group or file space using the FILEPOOL UNLOAD command, follow these steps:

1. Log on an administration machine.
2. Access MAINT's 193 disk.
3. Ensure the file pool is available. You should inform users of the impending back up. The data being backed up will be Read Only during the back up.
4. Enter a FILEDEF command to identify the backup file.

To identify a tape file, for instance, you might enter:

```
filedef unload tap1 sl valid volume-id
```

For *volume-id* specify a tape volume ID. Specifying SL on the FILEDEF command indicates IBM standard labels. If you use IBM standard labels, multivolume tape files are supported.

You can also direct backups to DASD, but in any case the ddname must be UNLOAD. For tape devices, if necessary, you should specify the label options and the tape device specifications (such as the number of tracks and density). A LABELDEF command may also be supplied for tape files if desired. For example:

```
labeldef unload fid file-id fseq sequence-number
```

Specify your own *file-id* and *sequence-number* as appropriate. (The FILEPOOL commands use CMS OS QSAM to process the unload files.)

The following is an example of how FILEDEF and LABELDEF commands might be used to back up multiple storage groups and a file space. In the example, storage groups 2, 3, and file space User4 in the SYSTST file pool are backed up:

```
filedef unload tap1 sl 1 valid systst
labeldef unload fid storage2 fseq 1
filepool unload group 2

filedef unload tap1 sl 2 valid systst
labeldef unload fid storage3 fseq 1
filepool unload group 3

filedef unload tap1 sl 3 valid systst
labeldef unload fid User4 fseq 1
filepool unload file space User4
```

The LABELDEF command makes it easier for you to keep track of your unload files. For more information on the FILEDEF and LABELDEF commands, see [z/VM: CMS Commands and Utilities Reference](#). For more about using tapes with CMS, see [z/VM: CMS Application Development Guide for Assembler](#).

5. Optionally enter a SET FILEWAIT ON command:

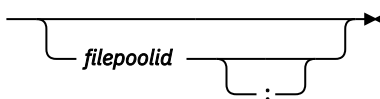
```
set filewait on
```

If you do not enter the SET FILEWAIT command, the FILEPOOL UNLOAD command will not work if anyone is writing to the storage group. If you enter the command, the FILEPOOL UNLOAD command will wait instead.



For more information on SET FILEWAIT command, see [z/VM: CMS Commands and Utilities Reference](#).

6. Enter a FILEPOOL UNLOAD command. An abbreviated format of the FILEPOOL UNLOAD command when backing up a storage group is:

► filepool — unload — group — *group\_number* 

To back up storage group 2 of the VMSYSU file pool, for example, you would enter:

```
filepool unload Group 2 vmsysu
```

An abbreviated format of the FILEPOOL UNLOAD command when backing up a file space is:

► filepool — unload — *filesystem* — *filesystemid* 

To back up file space User1 of the VMSYSU file pool, for example, you would enter:

```
filepool unload filesystem User1 VMSYSU
```

If you are backing up to tape, you will also be prompted to have the tapes mounted.

For more about the FILEPOOL UNLOAD command, see [“FILEPOOL UNLOAD” on page 495](#).

7. Enter a SET FILEWAIT OFF command:

```
set filewait off
```

You need to enter this command only if you entered SET FILEWAIT ON in a preceding step.

## Using FILEPOOL BACKUP

To back up a user storage group using the FILEPOOL BACKUP command, follow these steps:

1. Log on an administration machine.

The administrator that enters the FILEPOOL BACKUP command must be on the same processor as the file pool. The FILEPOOL BACKUP command links storage group minidisks to the administration machine, and the CP LINK command cannot link minidisks from a remote processor.

2. Access MAINT's 193 disk.
3. Ensure the file pool is available. You should inform users of the impending back up. The storage group being backed up will be Read Only during the back up.
4. Find the read passwords for the storage group minidisks.

Unless the storage group minidisks are protected by an external security manager, you should be prepared to supply the read passwords for all minidisks allocated to the storage group you plan to back up. During FILEPOOL BACKUP processing, you will be prompted to supply those passwords. If you want to avoid the prompts, you can code an exec in which you place the required passwords in the stack and then call the FILEPOOL BACKUP command. (For more information on how to use this command, see the usage notes under [“FILEPOOL BACKUP” on page 432](#).)

If the storage group minidisks are protected by an external security manager, the user ID of the virtual machine from which you are entering the FILEPOOL BACKUP command should already be appropriately authorized (For more information, see [“Enrolling an Administrator” on page 141](#)).

5. Enter a FILEDEF command to identify the backup file.

To identify a tape file, for instance, you might enter:

```
filedef backup tap1 sl valid volume-id
```

## Backing Up User Data

For *volume-id* specify a tape volume ID. Specifying SL on the FILEDEF command indicates IBM standard labels. If you use IBM standard labels, multivolume tape files are supported.

You can also direct backups to DASD, but in any case the ddname must be BACKUP. For tape devices, if necessary, you should specify the label options and the tape device specifications (such as the number of tracks and density). A LABELDEF command may also be supplied for tape files if desired. For example:

```
labeldef backup fid file-id fseq sequence-number
```

Specify your own *file-id* and *sequence-number* as appropriate. (The FILEPOOL commands use CMS OS QSAM to process the backup files.)

The following is an example of how FILEDEF and LABELDEF commands might be used to back up multiple storage groups. In the example, storage groups 2, 3, and 4 in the SYSTST file pool are backed up:

```
filedef backup tap1 sl 1 volid systst
labeldef backup fid storage2 fseq 1
filepool backup 2

filedef backup tap1 sl 2 volid systst
labeldef backup fid storage3 fseq 1
filepool backup 3

filedef backup tap1 sl 3 volid systst
labeldef backup fid storage4 fseq 1
filepool backup 4
```

The LABELDEF command makes it easier for you to keep track of your backups. For more information on the FILEDEF and LABELDEF commands, see [z/VM: CMS Commands and Utilities Reference](#). For more information about using tapes with CMS, see [z/VM: CMS Application Development Guide for Assembler](#).


6. Optionally enter a SET FILEWAIT ON command:

```
set filewait on
```

If you do not enter the SET FILEWAIT command, the FILEPOOL BACKUP command will not work if anyone is writing to the storage group. If you enter the command, the FILEPOOL BACKUP command will wait instead.

For more information on SET FILEWAIT command, see [z/VM: CMS Commands and Utilities Reference](#).

7. Ensure file mode A is accessed as read/write and is not in the storage group being backed up. This is needed because the FILEPOOL BACKUP processing uses the CMS GLOBALV facility to keep track of certain information needed for recovery from *catastrophic error*. Catastrophic error is any error that causes the usual stopping logic to be avoided.
8. Enter a FILEPOOL BACKUP command. An abbreviated format of the FILEPOOL BACKUP command is:

► filepool — backup — *group\_number* —  ◄

To back up storage group 2 of the VMSYSU file pool, for example, you would enter:

```
filepool backup 2 vmsysu
```

You will be prompted to enter the read passwords for all minidisks allocated to the storage group. If you are backing up the storage group to tape, you will also be prompted to have the tapes mounted. The FILEPOOL BACKUP command displays acknowledgment messages that let you know how far it has progressed.

For more information, see “FILEPOOL BACKUP” on page 432.

9. Enter a SET FILEWAIT OFF command:

```
set filewait off
```

You need to enter this command only if you entered SET FILEWAIT ON in a preceding step.

## Back Up Considerations if Storage Group Managed by DFSMS/VM

If the file pool being backed up is managed by DFSMS/VM, it may contain files that appear to be in the file pool, but whose data actually resides in the DFSMS/VM storage repository. Such files are often referred to as being in DFSMS/VM *migrated status*. The FILEPOOL BACKUP and FILEPOOL UNLOAD commands automatically back up both the primary and the migrated data for the storage group or file space being backed up. You should ensure you have the proper additional authorizations, and that the DFSMS/VM storage repository and file pool VMSYS are available. For detailed information on the additional requirements for backing up storage groups that are managed by DFSMS/VM, see the usage notes under “FILEPOOL BACKUP” on page 432 and “FILEPOOL UNLOAD” on page 495.

For detailed information about DFSMS/VM, migrated SFS files and managing storage groups, see [z/VM: DFSMS/VM Storage Administration](#) and [z/VM: DFSMS/VM Planning Guide](#).

## DASD Needed for Backing Up User Data

If you are backing up a storage group, it is worthwhile to enter the QUERY FILEPOOL STORGRP command to find the number of 4K Blocks In-Use in the storage group. If you are backing up a file space, it is worthwhile to enter the QUERY LIMITS command to find the number of 4K Blocks In-Use in the file space. That is the number of blocks the file pool server would back up for the storage group or file space. In addition to those blocks, the file pool server will back up control and catalog information related to the storage group. It also adds header information to each record. You should increase the number of 4KB blocks by about 15% to 20% to account for the control, catalog, and header information. The actual number of additional 4KB blocks needed varies depending on the number of files created, the number of aliases and authorizations related to those files, and so on.

## Concurrent User Data Backups

You can back up multiple storage groups or file spaces concurrently by entering FILEPOOL BACKUP or FILEPOOL UNLOAD commands from more than one administration machine. If, for instance, there are three administration machines, they can each issue a FILEPOOL BACKUP or FILEPOOL UNLOAD command for a different storage group or file space in the same file pool. The backups would occur concurrently.

Although the file pool server machine must do the same amount of processing as it would if the commands were issued sequentially by a single machine, there is more opportunity for I/O overlapping, which might reduce the net amount of time it would otherwise take.

## Staging User Data Backups

You should consider “staging” your backups. That is, you direct your backups to DASD and then copy them to tape at some later time. While this might seem like a useless step, it does allow speedier recovery, because the most recent backups of all user storage groups would be available immediately on DASD. If it is necessary to restore data, there is no time lost in finding and mounting the appropriate tape volumes.

When you are staging backups to tape, you copy the backup files to tape just before the next scheduled backup. After the backup completes successfully, you store the tapes somewhere outside your computer center. You have the most current backup on DASD and the next most recent backup on tape.

## Restoring User Data

If you want to restore one or more files from the storage group or file space, read “[Restoring Individual Files](#)” on page 123.

To restore a storage group that contains user data, you must have a backup of that storage group made by FILEPOOL BACKUP or FILEPOOL UNLOAD processing.



During the restore, the storage group will not be available to other users. After the restore completes, users can use the storage group immediately. You should send users a message indicating the restore is complete. If you would prefer to inhibit multiple user mode access until you have had a chance to verify all went well, see [“Restricting User Access in Multiple User Mode”](#) on page 69.

At the start and end of its processing for each file space in the storage group, the FILEPOOL RELOAD command displays the following message:

```
DMS3455I The reload of file space filespaceid is starting
DMS3455I The reload of file space filespaceid is complete
```

At the end of its processing for the storage group, the FILEPOOL RELOAD command displays the following message:

```
DMS3438I FILEPOOL RELOAD command successful
```

If the FILEPOOL RELOAD command is unsuccessful before the stopping message is displayed, consider running the FILEPOOL RELOAD command to restore just the file spaces that were not completely restored yet. This could save a lot of time. All of the data in the file spaces that were stated as complete should be ready to go, although you may need to manually enable the storage group.

The only possibility for damaged data in this scenario would be in the file space that was being reloaded at the time of the error, as indicated by message DMS3455I.

For more information, see [“FILEPOOL RELOAD”](#) on page 476.

7. Enter a SET FILEWAIT OFF command:

```
set filewait off
```

You need to enter this command only if you entered SET FILEWAIT ON in a preceding step.

8. Determine whether aliases are intact. See [“Concurrent User Data Restores”](#) on page 120 for details.

## From FILEPOOL UNLOAD for File Space

The file space owner needs to be aware of restore because the data will regress to some earlier state. Furthermore, the file space will not be available for use during the restore.

Follow these steps to restore a file space that was backed up by the FILEPOOL UNLOAD command:

1. Ensure the file pool is available in multiple user mode.
2. Log on a virtual machine that has file pool administration authority.
3. Enter a FILEDEF command (and, optionally, a LABELDEF command) for ddname RELOAD to identify the backup file.

Suppose, for example, you want to replace the contents of file space Jones which was backed up as a part of storage group 3. The backup file is the second file on tape volume SYSTST, which uses IBM standard labels. When making the backup, you used a LABELDEF command that specified STORAGE3 for the FID operand and 2 for the FSEQ operand. Assuming the tape is mounted at virtual device number 181, you would enter:

```
filedef reload tap1 sl 2 volid systst
labeldef reload fid storage3 fseq 2
```

When you enter the FILEPOOL RELOAD command in a later step, it will use the LABELDEF information to verify the tape label.

For more on the FILEDEF and LABELDEF commands, see [z/VM: CMS Commands and Utilities Reference](#). For more about using tapes with CMS, see [z/VM: CMS Application Development Guide for Assembler](#).

If the backup file is a CMS file named SG3 DATA that resides on a minidisk or SFS directory accessed as file mode B, you would enter:

## Restoring User Data

```
filedef reload disk sg3 data b
```

4. Optionally enter a SET FILEWAIT ON command:

```
set filewait on
```

If you do not enter the SET FILEWAIT command, the FILEPOOL RELOAD command will not work if anyone is reading from or writing to the file space. If you enter the command, the FILEPOOL RELOAD command will wait instead.

For more information on the SET FILEWAIT command, see [z/VM: CMS Commands and Utilities Reference](#).

5. Enter the FILEPOOL RELOAD command. An abbreviated format of the command is:

▶ **filepool** — **reload** — **filespace** — **filepaceid** — **filepoolid** — **:**

To restore file space JONES of the VMSYSU file pool, for example, you would enter:

```
filepool reload filespace JONES vmsysu
```

During the restore, the file space will not be available to other users.

At the end of its processing for the file space, the FILEPOOL RELOAD command displays the following message:

```
DMS3438I FILEPOOL RELOAD command successful
```

For more information, see “FILEPOOL RELOAD” on page 476.

6. Enter a SET FILEWAIT OFF command.

```
set filewait off
```

You need to enter this command only if you entered SET FILEWAIT ON in a preceding step.

7. Determine whether aliases are intact. See section “Concurrent User Data Restores” on page 120 for details.

## From FILEPOOL BACKUP

Follow these steps to restore a user storage group that was backed up by the FILEPOOL BACKUP command:

1. Inform file pool users of the impending restore. Users need to be aware of restore because the storage group data will regress to some earlier state. Furthermore, the storage group will not be available for read or write access during the restore. You should advise file pool users against attempting to reference the storage group until the restore completes. (If they ignore your advice, no harm will be done to the file pool. Their applications or commands, however, may get damaged return codes.)

Users enrolled in the storage group being restored will not be able to connect to the file pool until the restore completes successfully.

2. If you intend to stop the server before restoring user storage groups, consider increasing the CATBUFFERS startup parameter value to 5000 in the DMSPARMS file. Increasing CATBUFFERS prior to restoring a user storage group should reduce the amount of time it takes to do the restore.
3. Ensure the file pool is available in multiple user mode.
4. Log on a virtual machine that has file pool administration authority.

The administrator that enters the FILEPOOL BACKUP command must be on the same processor as the file pool. The FILEPOOL BACKUP command links storage group minidisks to the administration machine, and the CP LINK command cannot link minidisks from a remote processor.

For the VMSYS, VMPSFS, and VMSYSU file pools, the MAINT and MAINT $_{vrm}$  virtual machines have administration authority.

5. Enter a FILEDEF command (and, optionally, a LABELDEF command) for ddname RESTORE to identify the backup file.

Suppose, for example, you want to replace the contents of storage group 3. The backup file is the second file on tape volume SYSTST, which uses IBM standard labels. When making the backup, you used a LABELDEF command that specified STORAGE3 for the FID operand and 2 for the FSEQ operand. Assuming the tape is mounted at virtual device number 181, you would enter:

```
filedef restore tap1 sl 2 volid systst
labeldef restore fid storage3 fseq 2
```

When you enter the FILEPOOL RESTORE command in a later step, it will use the LABELDEF information to verify the tape label.

For more on the FILEDEF and LABELDEF commands, see [z/VM: CMS Commands and Utilities Reference](#). For more about using tapes with CMS, see [z/VM: CMS Application Development Guide for Assembler](#).

If the backup file is a CMS file named SG3 DATA that resides on a minidisk or SFS directory accessed as file mode B, you would enter:

```
filedef restore disk sg3 data b
```

The file you identify should be the most current backup. If, for some reason, you are not using the most current backup, you should read the usage note concerning Obsolete Backup Files listed under the description of “FILEPOOL RESTORE” on page 488.

6. Optionally enter a SET FILEWAIT ON command:

```
set filewait on
```

If you do not enter the SET FILEWAIT command, the FILEPOOL RESTORE command will not work if anyone is reading from or writing to the storage group. If you enter the command, the FILEPOOL RESTORE command will wait instead.

For more information on the SET FILEWAIT command, see [z/VM: CMS Commands and Utilities Reference](#).

7. Ensure file mode A is accessed as read/write and that it is not in the storage group being restored. This is needed because the FILEPOOL RESTORE processing uses the CMS GLOBALV facility to keep track of certain information needed for recovery from *catastrophic error*. Catastrophic error is any error that causes the usual stopping logic to be avoided.
8. Enter the FILEPOOL RESTORE command. An abbreviated format of the command is:

► filepool — restore — *group\_number* — *filepoolid* — *:*

To restore storage group 3 of the VMSYSU file pool, for example, you would enter:

```
filepool restore 3 vmsysu
```

During the restore, the storage group will not be available to other users. After the restore completes, users can use the storage group immediately. You should send the storage group users a message indicating the restore is complete. If you would prefer to inhibit multiple user mode access until you have had a chance to verify all went well, see [“Restricting User Access in Multiple User Mode”](#) on page 69.



At the end of its processing, the FILEPOOL RESTORE command displays the following message:

```
DMS3500I Restore of storage group nn  
in file pool filepoolid successfully completed
```

**If the FILEPOOL RESTORE command is unsuccessful before the stopping message is displayed, you should immediately rerun it.** Otherwise, certain resources may be left in an unusable state. The storage group itself is disabled early in FILEPOOL RESTORE processing. (That is, FILEPOOL RESTORE processing gets an *exclusive disable lock* on the storage group.) You should never, in this case, manually enable the storage group. The catalog data for the storage group may be corrupt. Re-enabling the storage group could lead to unpredictable results. Instead, run the FILEPOOL RESTORE command again.

If the rerun of the FILEPOOL RESTORE command does not succeed, you have to enter the FILEPOOL CLEANUP command to make available as many resources as possible.

For more information, see [“FILEPOOL RESTORE” on page 488](#) and [“FILEPOOL CLEANUP” on page 437](#).

**Note:** If you are restoring more than one storage group in the same file pool, do not try to restore them at the same time. Because the FILEPOOL RESTORE command updates the catalog data, multiple FILEPOOL RESTORE commands could interfere with each other, causing them to not complete. Always restore storage groups one at a time.

9. Enter a SET FILEWAIT OFF command.

```
set filewait off
```

You need to enter this command only if you entered SET FILEWAIT ON in a preceding step.

10. If you increased the CATBUFFERS parameter in the DMSPARMS file, you should reset the CATBUFFERS value at the earliest opportunity.
11. Determine whether aliases are intact. See [“Considerations for SFS Aliases After FILEPOOL FILELOAD” on page 127](#) for details.

## Concurrent User Data Restores

When using the FILEPOOL RELOAD command, you can restore multiple storage groups or file spaces concurrently by issuing FILEPOOL RELOAD commands from more than one administration machine. If, for instance, there are three administration machines, they can each issue a FILEPOOL RELOAD command for a different storage group or file space in the same file pool. The restores would occur concurrently. Although the file pool server machine must do the same amount of processing as it would if the commands were issued sequentially by a single machine, there is more opportunity for I/O overlapping, which might reduce the net amount of time it would otherwise take.

FILEPOOL RESTORE commands may not be entered concurrently. Because of the way the FILEPOOL RESTORE command updates the catalog data, multiple FILEPOOL RESTORE commands could interfere with each other, causing them to not complete.

## SFS Alias Consideration After Restoring User Storage Group (SFS only)

When a user storage group or file space is backed up, the base files, aliases, and directory structures are backed up as well as the authorizations related to them.

When FILEPOOL UNLOAD is used for backup, the ALL option must be specified or authorizations granted to each *filespaceid* being backed up will not be part of the backup file, and will therefore not be recreated by FILEPOOL RELOAD processing.

When you restore a storage group or file space, the base files, SFS aliases, directory structure and authorizations regress to the state they were in when the storage group or file space was backed up. Because SFS aliases can refer to files in other storage groups or file spaces, they are given special



attention during a restore. How the alias is restored varies depending on where the alias resides and where its base file resides. There are three cases:

1. The SFS aliases and the base file to which they refer reside in the same storage group or file space.

This is the simplest case. Because the entire storage group or file space is backed up and restored as a unit, the alias and base file regress to the state they were in at the time the backup was made.

2. SFS aliases in the restored storage group or file space refer to base files outside the storage group or file space.

These aliases are restored if the base file still exists in the other storage group or file space and the user is still authorized for it. Suppose, for instance, a user in storage group 2 creates an alias to a base file in storage group 3 on Tuesday morning. On Tuesday evening, storage group 2 is backed up. On Wednesday, the user erases the alias, but the base file still exists and the user is still authorized to read it. On Thursday, storage group 2 is restored from the Tuesday backup. Because the file still exists and the user is still authorized to see it, the alias will be restored, even though the user had erased the alias on Wednesday.

If the base file exists, but the user is no longer authorized to use it, the alias is restored as a *revoked* alias. For example, suppose a storage group 2 user has an active alias to a base file in storage group 3. Storage group 2 is backed up on Tuesday. On Wednesday, the base file owner revokes authority from the user for the base file. The creator of the alias sees the alias is revoked and erases it. On Thursday, storage group 2 is restored from the Tuesday backup. In this case, the alias is restored as a revoked alias, even though the alias was erased on Wednesday.

If the base file does not exist, the alias is restored as a revoked alias. The alias is not restored as an erased alias, as you might think it would be. The reason it is revoked and not erased is that several storage groups or file spaces may be scheduled to be restored. In one of the later restoration, the base file might be put back, in which case the alias's status is converted from revoked to active. FILEPOOL RESTORE and FILEPOOL RELOAD command processing does not know whether you later intend to restore other user storage groups or file spaces. So, instead of restoring the alias as erased, which may tempt users to erase the alias before you restore the storage group containing the base file, FILEPOOL RESTORE and FILEPOOL RELOAD processing restores the alias as revoked. Users are less inclined to erase a revoked alias until they find out why it was revoked.

To avoid any confusion when you intend to restore more than one user storage groups or file spaces, you should warn users that during the restoration they should not erase revoked aliases. After all storage groups and file spaces are restored, you should inform the users the restoration is complete and that they may erase revoked aliases if they wish. During the restoration, users may see their aliases switch from revoked to active status.

Note that users need to be notified only if you allow multiple user mode access during the restores. If you prevent users from accessing the file pool by restoring the storage groups in a secret multiple user mode session, users will not see the switch from revoked to active. See [“Changing the File Pool ID” on page 67](#) for more information.

3. Aliases *outside* the storage group or file space being restored refer to base files within that storage group or file space.

For restores done using FILEPOOL RELOAD, if the *ALL* option was specified on the FILEPOOL UNLOAD command, these aliases will be restored to the state they were in at the time of the FILEPOOL UNLOAD. If the *ALL* option was not specified, these aliases will not be restored. They will be deleted by FILEPOOL RELOAD processing.

In the case of restore with FILEPOOL RESTORE, the alias is not deleted by FILEPOOL RESTORE processing, but remains in effect if the base file is restored and the user is authorized to see the base file. If the base file is restored, but the user is not authorized for it, the alias is marked revoked. This could happen if, at the time of the backup, the user is not authorized for the file. After the backup is made, the user is granted authority for the file and creates an alias to it. If the backup is restored, the alias is marked revoked because the user was not authorized at the time the backup was made. If the base file is not restored (no longer exists in the storage group), the alias is marked erased.

Because of this processing, the order in which you restore storage groups or file spaces does not matter. When multiple storage groups or file spaces need to be restored, you do not need to restore the groups in the order the backups were made. No matter which order you use, the results are the same.

Finally, if the file pool is available to users during a restore, users will encounter implicit lock conflicts if they try to use aliases that refer to storage groups or file spaces that are being restored.

## Incremental Availability During Multiple FILEPOOL RESTORES

If you are restoring more than one storage group or file space, each becomes available as it is restored. This recovery characteristic is often referred to as *incremental availability*. As each storage group or file space is restored, the users of that group can become productive.

If you are restoring more than one storage group, and any of these storage groups contain DFSMS/VM Migration Level 1 repositories, restore that repository first.

It might be reasonable to permit incremental availability as follows:

1. Warn file pool users about the impending restore.
2. Enter DISABLE operator commands to disable all storage groups you intend to restore.

When you make the file pool available in increments, users should be made aware of the following:

- References to storage groups that are still disabled should be kept to a minimum. The other storage group may not yet be restored, so the user or application will encounter the disable exclusive lock. Users would receive messages such as:

```
DMS1137E Object is locked
DMS1138E File sharing conflict
```

- Administrators should be aware they should not enable storage groups or file spaces that are yet to be restored.
  - Users should expect to see revoked SFS aliases changed to usual active aliases as more of the storage groups and file spaces are restored. Users should also be told when the restore is complete so they can erase any revoked aliases they no longer want to keep. (See [“Concurrent User Data Restores”](#) on page 120.)
3. Users can access data in all storage groups and file spaces except those you have disabled.
  4. Because the storage groups being restored are disabled, there is no need to enter SET FILEWAIT ON commands for them. (No one can be using the storage group, so there will not be any implicit locks on it.) After restoring the data in a storage group, FILEPOOL RESTORE processing automatically enables the storage group for general use.

This technique is particularly useful if you must restore the file pool by generating it again and restore all the user storage groups.

## Using SFS Unresolved Aliases (SFS only)

SFS unresolved aliases can improve the performance of user applications that restore file pools and storage groups by using CMS commands and routines to overwrite file pool objects from a backup medium. Ordinarily, a user cannot have an alias for a base file unless the file exists and the user is authorized to access it. Unresolved aliases can be created when these conditions are not met, so it is not necessary to process aliases separately.

To use SFS unresolved aliases, an application that is restoring file pool objects must:

- Restore an alias by using the DMSCRALI CSL routine with the UNRESOLVED option:
  - If the base file does not exist, an unresolved alias is created.
  - If the necessary authorizations do not exist, an unresolved alias is created. In this case, the alias can be resolved after the authorizations are restored by opening the base file with DMSOPBLK RESOLVE and creating, replacing, or changing the file.
  - If the base file and the authorizations exist, an alias is created.

- Restore a base file by using the DMSOPBLK CSL routine with the RESOLVE option:
  - If necessary authorizations exist, any unresolved aliases to the file are resolved.
  - If authorizations do not exist, any unresolved aliases to the file are converted to revoked aliases.

If the restoration process must be interrupted, lock the file pool or storage group so unresolved aliases cannot be overwritten.

For details, see the description of DMSCRALI and DMSOPBLK in *z/VM: CMS Callable Services Reference*.

If the alias is not resolved, it remains in the file pool catalog; FILELIST and LISTFILE show it as a revoked alias. When unresolved aliases remain after the restoration operation:

1. Get a list of unresolved aliases:

- Enter FILESERV LIST and search for OBJECTCAT rows with a status of J.
- Or enter DMSOPCAT and search for OBJECTCAT rows with a status of J.

2. Remove the unresolved alias:

- Erase it.
- Attempt to resolve it: Using DMSOPBLK RESOLVE, open the base file and change it. If the attempt does not resolve the alias, it is converted to a revoked alias.
- Overwrite it with other types of objects with the same name, for example, by opening a file with the same name, copying a file or external object over the alias, or creating an alias with the same name.

## Finding Corrupted Files When a Minidisk is Corrupted

When one minidisk in a user storage group is known to be corrupted, running the FILEPOOL LIST MINIDISK command for the corrupted minidisk produces a file containing a list of all the files that have at least one 4KB block of data in the corrupted minidisk. This file can then be used as input to the FILEPOOL FILELOAD or the FILEPOOL RELOAD command with the FILES option to restore only these corrupted files. See [“FILEPOOL LIST MINIDISK” on page 467](#) and [“FILEPOOL RELOAD” on page 476](#) for details.

## Considerations if Storage Group is Managed by DFSMS/VM

If the file pool is managed by DFSMS/VM, it may contain migrated files. The FILEPOOL RESTORE and FILEPOOL RELOAD commands automatically restore both the primary and the migrated data for the storage group. You should ensure you have the proper additional authorizations, and the DFSMS/VM storage repository and file pool VMSYS are available.

If you are restoring several storage groups, one or more of which contain DFSMS/VM Migration Level 1 repositories, you should restore those groups first.

For detailed information on the additional requirements for restoring storage groups that are managed by DFSMS/VM, see the usage notes for [“FILEPOOL RESTORE” on page 488](#). For detailed information about DFSMS/VM, migrated SFS files and managing storage groups, see *z/VM: DFSMS/VM Storage Administration* and *z/VM: DFSMS/VM Planning Guide*.

## Restoring Individual Files

If you want to restore one or more files from a copy of the storage group data created by FILEPOOL BACKUP processing, use the FILEPOOL FILELOAD command. This lets you restore only the files you want to restore, by specifying them in a control file.

If you want to restore one or more files from a copy of the storage group data or file space data created by FILEPOOL UNLOAD processing, use the FILEPOOL RELOAD FILES command. This lets you restore only the files you want to restore, by specifying them in a control file.

This is useful when you have made changes to the data after it was backed up, but you need to restore one or more base files. Restoring the entire storage group or file space would not be productive because you would erase all of the changes. You can restore only the file(s) you need.

When selecting the files to be restored, you may want to use the output from the FILEPOOL LIST BACKUP command or the FILEPOOL LIST MINIDISK command. FILEPOOL LIST BACKUP lists the contents of a backup file which you can use to select the files to be restored. This is useful when determining what the state of the storage group or file space when the backup was taken. FILEPOOL LIST MINIDISK lists the base files with blocks located on a specific minidisk which can also be used to select the files for restore. This is useful if a single minidisk is damaged and needs to be restored. See [“FILEPOOL LIST BACKUP” on page 459](#) and [“FILEPOOL LIST MINIDISK” on page 467](#) for details.

### From FILEPOOL BACKUP

Follow these steps to restore individual file(s) from a storage group that was backed up using FILEPOOL BACKUP. If the storage group contains Byte File System (BFS) data, the CMS short name must be specified to identify the file. In this case, the BFS file data can only be replaced. The catalog data for the file must be intact. Note that if a minidisk in a user storage group is corrupted, the catalog data for the corrupted file data is still intact.

1. Log on a virtual machine that has file pool administration authority.

For the VMSYS, VMPSFS, and VMSYSU file pools, the MAINT and MAINTvrm virtual machines have administration authority.

2. Create a CMS file on your A-disk, named CONTROL FILELOAD, that contains the list of files to be restored. The records in the file CONTROL FILELOAD A must have the following format:

```
filename filetype dirname
```

where:

**filename**

is the name of the file to be restored.

**filetype**

is the type of the file to be restored.

**dirname**

is the fully qualified directory name where the file to be restored resided at the time the backup was made. You can omit the name of the file pool in the *dirname*. In this case the file pool ID specified on the FILEPOOL FILELOAD command will be used. If the file pool ID is not specified on the command either, the default file pool ID currently in effect will be used.

For example, the control file might look something like:

```
TEXTBOOK SCRIPT VMSYSU: JONES .NEWBOOK
PROGA ASSEMBLE VMSYSU: FISHER .DEPARTMENT45 .TESTS
JOHNSON NAMES JOHNSON .
2 0 VMSYSU: SMITH
3 0 SMITH
```

The length of the records in the control file is limited to 256 bytes.

3. Enter a FILEDEF command (and optionally a LABELDEF command) for ddname RESTORE to identify the backup file.

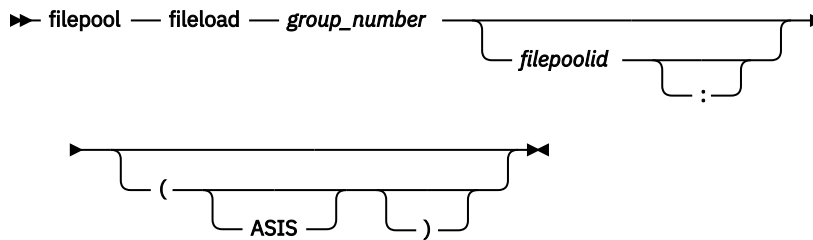
Suppose, for example, you want to replace one or more files in storage group 3. The backup file for storage group 3 is the second file on tape volume SYSTST, which uses IBM standard labels. When making the backup, you used a LABELDEF command that specified STORAGE3 for the FID operand and 2 for the FSEQ operand. Assuming the tape is mounted at virtual device number 181, you would enter:

```
filedef restore tap1 sl 2 volid systst
labeldef restore fid storage3 fseq 2
```

When you enter the FILEPOOL FILELOAD command in a later step, it will use the LABELDEF information to verify the tape label.

For more on the FILEDEF and LABELDEF commands, see [z/VM: CMS Commands and Utilities Reference](#). For more about using tapes with CMS, see [z/VM: CMS Application Development Guide for Assembler](#).

4. Enter the FILEPOOL FILELOAD command. The format of the command is:



To restore files from storage group 2 of the VMSYSU file pool, you would enter:

```
filepool fileload 2 vmsysu
```

Use the ASIS option if you do not want the file ID specifications in the CONTROL FILELOAD file to be translated to upper case characters.

If the file pool ID was changed since the backup file was created, you will be asked if execution should continue. Verify you are restoring files from the correct file pool before continuing.

During the restore, a message is displayed if an error occurs that prevents a file from being restored. For example, if the specified *dirname* is not found, the file will not be restored. The restore process does not end at this point. The process continues with the rest of the files. If you run into a virtual storage constraint, increase your storage size. If syntax errors are found in the file CONTROL FILELOAD, a message will be displayed telling you the file was not restored, and the restore processing continues with the remaining files. If the number of syntax errors reaches 20, the FILEPOOL FILELOAD command will be canceled and you will receive a message.

At the end of processing, the FILEPOOL FILELOAD command displays the following message:

```
DMS3620I n files restored
```

5. Determine whether aliases are intact. See [“ Considerations for SFS Aliases After FILEPOOL FILELOAD”](#) on [page 127](#) for details.

## From FILEPOOL UNLOAD

Follow these steps to restore individual file(s) from a storage group or file space that was backed up using FILEPOOL UNLOAD.

1. Log on a virtual machine that has file pool administration authority.

For the VMSYS, VMPSFS, and VMSYSU file pools, the MAINT and MAINT $_{vrm}$  virtual machines have administration authority.

2. Create a CMS file on your A-disk, named CONTROL RELOAD, that contains the list of the files to be restored.

There are 2 control record formats:

a. *filename filetype dirname*

*filename filetype* is the name of an SFS file or the CMS short name of a Byte File System (BFS) file.

*dirname* is the fully qualified directory name for SFS files, or the *bfsid* for BFS files.

If you omit the ID of the file pool in *dirname*, it defaults to the file pool ID used by the FILEPOOL RELOAD FILES command.

b. *fully qualified pathname*

*fully qualified pathname* is in the format: `././VMBFS:filepoolid:filespaceid/pathname`

If a blank is the last character of the pathname, a '/' or X'00' must be appended to delimit the end of the pathname. A '/' or X'00' are valid as ending delimiters in any case, and will not be considered to be part of the pathname.





RELOAD, a message will be displayed telling you the file was not restored, and the restore processing continues with the remaining files. If the number of syntax errors reaches 20, the FILEPOOL RELOAD FILES command will be canceled and you will receive a message.

At the end of processing, the FILEPOOL RELOAD FILES command displays the following message:

```
DMS3620I n files restored
```

7. Determine whether aliases are intact. See [“Considerations for SFS Aliases After FILEPOOL FILELOAD” on page 127](#) for details.

## Restoring Individual Files in DFSMS/VM Migrated Status

If the file you are restoring was in DFSMS/VM *migrated status* at the time of the backup, the file will be restored in the appropriate DFSMS/VM migration repository. Therefore, if you are restoring migrated files, you should ensure you have the proper additional authorizations, and that the DFSMS/VM storage repository and file pool VMSYS are available.

For detailed information on the additional requirements for restoring storage groups managed by DFSMS/VM, see the usage notes for [“FILEPOOL FILELOAD” on page 451](#) or [“FILEPOOL RELOAD” on page 476](#), (with the FILES option). For detailed information about DFSMS/VM, migrated files, and managing storage groups, see [z/VM: DFSMS/VM Storage Administration](#) and [z/VM: DFSMS/VM Planning Guide](#).

## Considerations for SFS Aliases After FILEPOOL FILELOAD

In contrast with restoring an entire storage group, the FILEPOOL FILELOAD command does not restore SFS aliases and authorizations. The following three cases are possible:

- If the restore process results in replacing an existing file with the same *filename filetype dirname*, the authorizations and aliases of the existing file are preserved. Note that they may not necessarily be the same as the authorizations and aliases of the file at the time the backup file was created.
- If the restore process results in the creation of a new file, the newly created file has no aliases. The newly created file has only those authorizations created as a result of current directory level authorizations (NEWREAD or NEWWRITE authorizations for file control directories, or DIRREAD or DIRWRITE authorizations for directory control directories). Therefore, the aliases and any other authorizations that existed at the time the backup file was created are not restored.
- If *filename filetype dirname* represents an alias to a base file, the alias is not restored. Use the CREATE ALIAS command to recreate it. For more information, see [z/VM: CMS Commands and Utilities Reference](#).

## Considerations for SFS Aliases After FILEPOOL RELOAD FILES

The FILEPOOL RELOAD command with the FILES option restores SFS aliases and authorizations as follows: The following three cases are possible:

- If the restore process results in replacing an existing file with the same *filename filetype dirname*, the authorizations and aliases of the existing file are preserved. If there were other authorizations and aliases at the time of the backup, they will also be restored.
- If the restore process results in the creation of a new file, the authorizations and aliases that existed at the time of the backup will be restored.
- If *filename filetype dirname* represents an alias to a base file, the alias will be recreated.

## Restoring a File Pool by Generating It Again

This is the recovery procedure to use when everything else does not work. Read all the instructions before starting.

To restore a file pool by generating it again, you should have a backup of every user storage group. Storage groups for which you have no backups are permanently lost. Follow these steps:

## Restoring a File Pool

1. Log on or reconnect to the server machine. Stop multiple user mode processing if it is not already stopped.
2. Add the NOFORMAT startup parameter to the DMSPARMS file. (For more information, see [NOFORMAT](#).)
3. Change the BACKUP parameter in the DMSPARMS file to NOBACKUP if BACKUP is specified. This will avoid the file pool logs filling during the restore process, and the time involved in taking automatic control data backups will be eliminated.
4. Enter the FILESERV GENERATE command, specifying the POOLDEF file as input. If, for example, the POOLDEF file is named VMSYSU POOLDEF A, you would enter:

```
fileserv generate vmsysu pooldef a
```

If you do not have a POOLDEF file, you need to reconstruct it. The formats of the statements in a POOLDEF file are identical to the formats of the FILESERV GENERATE control statements. See [“FILESERV GENERATE” on page 513](#).

Make sure the values of the BLOCKS parameters on the 'DDNAME=MDKnnnnn' control statements remain unchanged. If you cannot reconstruct the BLOCKS values, and you never changed the physical DASD for any of your storage group minidisks, you can omit the BLOCKS parameters.

During FILESERV GENERATE processing, for every minidisk in the filepool that is FBA and not aligned on a 4K boundary, you will receive informational and warning messages. Processing continues, but you will not be able to use data spaces until the FBA minidisks are realigned.

When the FILESERV GENERATE command completes successfully, what you have, essentially, is an empty file pool.

5. Remove the NOFORMAT startup parameter from the DMSPARMS file.
6. Start multiple user mode processing. Enter:

```
fileserv start
```

**Attention:** Do not do anything to the storage group between the FILESERV GENERATE and the FILEPOOL RESTORE or FILEPOOL RELOAD. Unpredictable results could occur.

7. Restore all the user storage groups. Follow the instructions in [“Restoring User Data” on page 115](#).

You may use either FILEPOOL RESTORE or FILEPOOL RELOAD to restore your storage groups, depending on how you created your backup files. If you want to restore more than one storage group at a time from multiple administrator machines, you may do so if you are using FILEPOOL RELOAD, but you may not do so if you are using FILEPOOL RESTORE.

**Attention:** If you do not have a backup for all the storage groups, you have to permanently prevent access to the missing storage groups. Re-using the storage group could yield unpredictable results. To prevent access, enter a DISABLE GROUP operator command or FILEPOOL DISABLE GROUP administrator command for the storage group. Specify both the EXCLUSIVE and DETACH options. You should then abandon the use of that storage group forever. A good way to prevent an administrator from inadvertently re-enabling a storage group is to specify a nonexistent user ID on the DISABLE command or FILEPOOL DISABLE command. Because the user ID is nonexistent, someone would have to explicitly specify it on an ENABLE operator command or FILEPOOL ENABLE administrator command to re-enable the storage group. See [“DISABLE” on page 412](#) for a description of the DISABLE operator command and [“FILEPOOL DISABLE” on page 443](#) for a description of the FILEPOOL DISABLE administrator command.

If you want to remove the minidisks from the unusable storage group, see [“Removing Space from a File Pool” on page 206](#).

If you are restoring more than one storage group, and any of these storage groups contain DFSMS/VM Migration Level 1 repositories, restore that repository first. If it is vital for storage groups to be made available to users as soon as they are restored, see the instructions in [“Incremental Availability During Multiple FILEPOOL RESTORES” on page 122](#).



8. Optionally enroll PUBLIC.

If PUBLIC was enrolled in the file pool, that enrollment was lost when the file pool was generated. If you want PUBLIC enrolled in this file pool, enroll it now by entering:

```
enroll public vmsysu
```

Substitute your file pool ID for VMSYSU.

9. If the BACKUP startup parameter was in effect, and you switched to NOBACKUP in step “3” on page 128, stop the server using the STOP command, change the NOBACKUP parameter to BACKUP, and back up the control data using a FILESERV BACKUP command. This backup reflects all the changes made to the control data by the restores of the user storage groups.

10. Contact the administrators of any NFS servers that might have SFS directories NFS-mounted. (You can identify NFS server machines because they must be administrators for the file pool being generated. Typically the user ID will be VMNFS.)

All NFS-mounted SFS directories must be unmounted before the file pool is restarted. This is a requirement because NFS file handles kept by NFS clients contain encoded pointers to objects in the file pool. Following a FILESERV GENERATE, it is likely a pointer for a file now points to a different file, or even to a directory.

11. Start multiple user mode processing by entering FILESERV START command.

You have now restored the file pool server by generating it again.

## Replacing File Pool Minidisks

This topic describes how to replace any minidisk in an SFS or CRR file pool.

It also describes how to replace the server machine's work minidisk, which contains the POOLDEF file. You may want to replace a file pool minidisk because:

- The minidisk is ‘damaged’, which could have been caused by an irrecoverable DASD error.
- You want to balance your DASD workload. If you have DFSMS/VM installed, see “Moving File Pool Minidisk Data Using DFSMS/VM” on page 134.
- You want to move your minidisks to a different device type. If you have DFSMS/VM installed, see “Moving File Pool Minidisk Data Using DFSMS/VM” on page 134.

To replace a file pool minidisk, you need a backup of the data on the minidisk being replaced. Table 10 on page 129 describes what form of backup you will need (if any) to restore any file pool minidisks.

To replace this kind of minidisk:	You need:
Work minidisk on which POOLDEF file resides.	The current backup of the control data. You should also have some other backup of the other files on the minidisk.
File pool control minidisk or storage group 1 minidisk in a dedicated CRR or FIFO file pool.	No backups required.
File pool control minidisk or storage group 1 minidisk in a repository file pool.	The current backup of the control data and the current log minidisks.
Storage group 2 through <i>n</i> minidisk in a repository file pool.	A current backup of the storage group in which the minidisk resides.
Storage group 2 through <i>n</i> minidisk in a dedicated CRR or FIFO file pool.	No backups required.

<i>Table 10. Forms of Backups Needed to Replace File Pool Minidisks (continued)</i>	
<b>To replace this kind of minidisk:</b>	<b>You need:</b>
One file pool log minidisk.	The other file pool log minidisk.
Both file pool log minidisks.	Follow the instructions under “ <a href="#">Replacing Both File Pool Log Minidisks</a> ” on page 133. Do not follow the instructions in this section.
One CRR log minidisk.	Follow the instructions under “ <a href="#">Replace One CRR Log Minidisk</a> ” on page 328. Do not follow the instructions in this section.
Both CRR log minidisks.	Follow the instructions under “ <a href="#">Replace Both CRR Log Minidisks</a> ” on page 328. Do not follow the instructions in this section.

The following steps need to be taken to replace any file pool minidisks except if it is both file pool logs or one or both CRR logs. Specific instructions for each type of file pool minidisk replacement are in the following sections.

1. If possible, take a backup as listed under the “You need:” column in [Table 10 on page 129](#).
2. Update the MDISK control statements for the affected minidisks in the z/VM directory entries for the server machine. [Figure 35 on page 257](#) shows example MDISK control statements for the file pool minidisks as well as the server machine work minidisk. **You must not change the virtual device numbers when updating the MDISK statements.**

For file pools, specify MINIOPT NOMDC for the control and log minidisks, but not for the rest of the file pool minidisks. For CRR file pools, specify MINIOPT NOMDC for all the dedicated CRR or FIFO file pool minidisks. Note the virtual numbers of the minidisks being replaced (for use later).

**Care must be taken to ensure the number of 4KB blocks in the new minidisk is greater than or equal to the BLOCKS parameter on the 'DDNAME=MDKnnnnn' control statement for this minidisk in the POOLDEF file. It is critical the BLOCKS parameter remains the same as it was before.** For more information on the BLOCKS parameters, see “[FILESERV GENERATE](#)” on page 513.

If you are replacing the minidisk with one on the same device type, define the same number of cylinders or blocks. (Where both the devices are FB-512 devices, you can consider them to be the same device type, even if they are different DASD models. Make sure FB-512 devices start and end on a 4KB boundary.)

If you are replacing a file pool minidisk with one on a *different* device type, you should define the new minidisk to be slightly larger than the old. Because of rounding that occurs in the space allocation algorithms, it is nearly impossible to define minidisks on two different device types that (to a server) are equal. (Where both the old and the new minidisks are FB-512 devices, you can consider them to be the same device type, even if they are different DASD. You, therefore, do not have rounding considerations.) To determine approximately equivalent sizes of minidisks on various devices, see your device documentation.

3. Log on the server machine. If the server is currently processing in multiple user mode, reconnect to it and enter STOP NOBACKUP.
4. Do the following for each replaced **file pool** minidisk (skip this step for the work minidisk):
  - a. Enter the following CP command to make the minidisk write-accessible to the server machine:

```
link * vdev1 vdev2 w
```

where:

\*

indicates the minidisk is defined in the z/VM directory entry for this virtual machine.

**vdev1 vdev2**

specifies the virtual device number used on the MDISK statement (step “2” on page 130) for both *vdev1* and *vdev2*.

**w**

indicates write mode access is required for this minidisk.

- b. Enter the following CMS command to initialize and label the minidisk:

```
format vdev fm (blksize size
```

where:

**vdev**

specifies the virtual device number you used on the CP LINK command previously.

**fm**

an unused file mode

**size**

is 512 if the minidisk is the file pool control minidisk. Otherwise, specify 4096.

The FORMAT command will prompt you for the minidisk volume label (disk label). You may enter any valid volume label value. File pools use the letter **T** followed by the virtual device number of the minidisk. If, for example, the minidisk is defined at virtual device number 301, file pools use T301 as the minidisk volume label.

The FORMAT command will prompt you whether to erase all files on the minidisk. Reply 1 for YES to this message (being sure you have specified the correct minidisk).

- c. Enter the following CMS command to allocate the minidisk as a block I/O file:

```
reserve filename filetype filemode
```

where:

**filename**

uses the file pool ID as the file name

**filetype**

uses the file ddname as the file type. The ddnames are:

**CONTROL**

for the control minidisk.

**LOG1**

for the first file pool log minidisk.

**LOG2**

for the second file pool log minidisk.

**MDKnnnnn**

for minidisks allocated to storage groups (for example, MDK00005).

**filemode**

must be the *access-letter* you entered earlier for the CMS FORMAT command.

The RESERVE command will prompt you whether to erase all files on the minidisk. Reply 1 for YES to this message (being sure you have specified the file mode letter that matches the CMS FORMAT command).

Read the section for replacing the type of minidisk you are replacing for any further instructions.

## Replacing One File Pool Log Minidisk

Follow the general steps for replacing a minidisk in “[Replacing File Pool Minidisks](#)” on page 129. Enter FILESERV START as usual. After you enter FILESERV START, both SFS logs will be functional.

### Replacing a Control Minidisk or Storage Group 1 Minidisk

Is the minidisk in a dedicated CRR or FIFO file pool or is it in a repository file pool?

#### In a CRR or FIFO File Pool

Follow the general steps for replacing a minidisk in [“Replacing File Pool Minidisks” on page 129](#). Then follow these additional steps:

1. In a CRR recovery server's DMSPARMS file, change the CRR startup parameter to NOCRR.

**Note:** For CRR this step is very important. Changing the CRR startup parameter to NOCRR prevents FILESERV GENERATE from formatting the CRR log minidisks, which would destroy active sync point data.

2. For CRR or FIFO servers enter the FILESERV GENERATE command, specifying the POOLDEF file as input. If, for example, the POOLDEF file is named VMSYSU POOLDEF A, you would enter:

```
fileserv generate vmsysu pooldef a
```

3. For CRR, change the NOCRR parameter in the recovery server's DMSPARMS file back to CRR.
4. For CRR or FIFO servers, enter a FILESERV START command to resume multiple user mode processing.

#### In a Repository File Pool

Follow the general steps for replacing a minidisk in [“Replacing File Pool Minidisks” on page 129](#). Then restore the control data using instructions in [“Restoring Control Data” on page 110](#).

### Replacing Storage Group 2 through n Minidisks

Is the minidisk in a dedicated CRR or FIFO file pool or is it in a repository file pool?

#### In a CRR or FIFO File Pool

Follow the general steps for replacing a minidisk in [“Replacing File Pool Minidisks” on page 129](#). Then enter FILESERV START.

#### In a Repository File Pool

If you are replacing all the minidisks in a storage group, follow the general steps for replacing a minidisk in [“Replacing File Pool Minidisks” on page 129](#), and then follow the steps under [“Restoring User Data” on page 115](#) to restore all the objects in the storage group. If you are replacing one or more minidisks in a storage group, but not all of the minidisks in a storage group, you may want to restore a set of individual files instead. Follow the general steps for replacing a minidisk in [“Replacing File Pool Minidisks” on page 129](#), and then follow the steps to restore the individual files that have been lost in [“Restoring Individual Files” on page 123](#).

### Replacing the Work Minidisk Containing POOLDEF File

If the work minidisk containing the POOLDEF file is being replaced, you should restore the files on that minidisk using your restore mechanisms for non-repository file pool minidisks. The POOLDEF file will be regressed if a minidisk has been added to the file pool since the backup, or the file pool or CRR log minidisks have been moved using FILESERV LOG or FILESERV CRRLOG, or the AUDIT or BACKUP file definitions have been changed using FILESERV DEFAUDIT or FILESERV DEFBACKUP. If the POOLDEF file is regressed, you will either need to restore the control data to restore the POOLDEF file (see [“Restoring Control Data” on page 110](#)), or you may manually reconstruct POOLDEF file if you are aware of all of the changes to the POOLDEF file since the minidisk was backed up.

## Replacing Both File Pool Log Minidisks

These directions apply whether the file pool log minidisks are in a repository file pool or a CRR file pool. This section describes how to replace both file pool log minidisks. You might want to replace both file pool log minidisks because:

- You want to increase or decrease the space allocated.
- You need to replace both log minidisks lost because of media errors.
- You want to move both logs to another physical device.
- You need to reformat both logs because they have been corrupted (perhaps the DASD areas were accidentally overwritten).

To replace both log minidisks, follow these steps:

1. If possible, force the file pool server to apply the changes recorded in the logs.

**Do not skip this step.** To apply all changes in the logs to the file pool, you must allow multiple user mode processing to stop usually:

- If the file pool server is now running in multiple user mode, enter a STOP NOBACKUP operator command. Do **not** use a STOP IMMEDIATE command. Also, do not enter STOP BACKUP, as the backup created would be useless.

2. Verify the FORMAT startup parameter is in effect.

Check the DMSPARMS file for the NOFORMAT startup parameter. If it is specified, delete it. You do not need to explicitly specify FORMAT, as it is the default.

3. If you need to change the MDISK control statements for the log minidisks:

- Log off the server machine.
- Using standard operating procedures for your installation, update the z/VM system directory entry for the file pool server machine. Change the MDISK control statements for the log minidisks to indicate the new size and location. The log minidisks must be of identical size, so they should be defined on the same DASD device type. (Where both the devices are FB-512 devices, you can consider them to be the same device type, even if they are different DASD models.) Specify MINIOPT NOMDC for the log minidisks.
- Log on the server machine.

If you are changing the physical location of the log minidisks, you should review the log placement considerations in [“Where to Place the Logs”](#) on page 253.

In addition to changing the log sizes and, perhaps, their placement, you can also change the virtual device numbers. If you change the virtual device numbers, remember the new numbers. They are needed in a later step.

4. Log on the server machine and enter a STOP NOBACKUP operator command.
5. Enter the FILESERV LOG command. The FILESERV LOG command must be entered from the server machine console.

When entering the FILESERV LOG command, specify the virtual device numbers of the new log minidisks. If, for instance, you defined your log minidisks at virtual device numbers 291 and 292, you would enter:

```
fileserv log 291 292
```

If the command does not complete successfully, enter it again after correcting the problem. For a complete description, see [“FILESERV LOG”](#) on page 523.

**Attention:** If you were unable to successfully apply the log changes as directed in step 1, you will get message DMS3007R. To restore your file pool to a consistent state, follow the instructions in [“Restoring a File Pool by Generating It Again”](#) on page 127. Skip the remaining steps.

6. The reformatting of the log minidisks makes previous control data backups obsolete. If the BACKUP startup parameter is in effect, enter a FILESERV BACKUP command:

```
fileserv backup
```

For a complete description, see [“FILESERV BACKUP” on page 500](#).

7. Enter the FILESERV START command to resume multiple user mode processing:

```
fileserv start
```

## Moving File Pool Minidisk Data Using DFSMS/VM

If you have DFSMS/VM installed, you can save time by using the DFSMS/VM high performance data mover to replace an existing minidisk in a file pool with a minidisk of a different or same device type. Follow these steps:

1. Update the MDISK control statements for the affected minidisks in the z/VM directory entries for the server machine. [Figure 35 on page 257](#) shows example MDISK control statements for the file pool minidisks as well as the server machine work minidisk. **You must not change the virtual device numbers when updating the MDISK statements.** For example, to replace the 305 minidisk, change the virtual device number on the MDISK statement for that minidisk in the server machine's directory entry to a free virtual number (such as 555). Then, add an MDISK statement to the server machine's directory entry for the new minidisk specifying 305 as the virtual number.

For repository file pools, specify MINIOPT NOMDC for the control and file pool log minidisks, but not for the rest of the file pool minidisks. For dedicated CRR or FIFO file pools, specify MINIOPT NOMDC for all the file pool minidisks. Note the virtual numbers of the minidisks being replaced (for use later).

**Care must be taken to ensure the number of 4KB blocks in the new minidisk is greater than or equal to the BLOCKS parameter on the 'DDNAME=MDKnnnnn' control statement for this minidisk in the POOLDEF file. It is critical the BLOCKS parameter remains the same as it was before.** For more information on the BLOCKS parameters, see [“FILESERV GENERATE” on page 513](#).

If you are replacing the minidisk with one on the same device type, define the same number of cylinders or blocks. (Where both the devices are FB-512 devices, you can consider them to be the same device type, even if they are different DASD models. Make sure FB-512 devices start and end on a 4KB boundary.)

If you are replacing a file pool minidisk with one on a *different* device type, you should define the new minidisk to be slightly larger than the old. Because of rounding that occurs in the space allocation algorithms, it is nearly impossible to define minidisks on two different device types that (to a server) are equal. (Where both the old and the new minidisks are FB-512 devices, you can consider them to be the same device type, even if they are different DASD. You, therefore, do not have rounding considerations.) To determine approximately equivalent sizes of minidisks on various devices, see your device documentation.

2. Log on the server machine. If the server is currently processing in multiple user mode, reconnect to it and enter STOP NOBACKUP.
3. Enter the DFSMS COPY command for each minidisk being replaced. See [z/VM: DFSMS/VM Storage Administration](#) for information on how to use this command.
4. Enter FILESERV START.
5. Delete the old minidisk from the server machine's directory entry.

## Using Non-File Pool Facilities for Backing Up Your File Pool

If you are considering using a non-file pool utility for backing up and restoring your file pools, you should determine whether the facility is aware of the interrelationships of file pool repository data. See the documentation for the facility to determine how to back up file pool data.

Other facilities, such as the DASD Dump Restore Service Program, back up physical DASD space without regard for the meaning of the data. The facility is not aware the data it is backing up or restoring is a part of some larger entity. For these kinds of facilities, there are two points to keep in mind:

1. The file pool must be backed up when the file pool server is shut down.
2. **All** minidisks (except the CRR log minidisks) on which the file pool resides must be backed up during that shutdown. You cannot back up half the minidisks today, start the file pool server for usual operation, and then back up the remaining minidisks tomorrow. Backups made in that way are not valid. They cannot be used to restore the file pool when there is a data loss. If such a backup is used, subsequent file pool server processing is unpredictable.

A file pool, as large as it may be, is really a single entity. The data on the minidisks that make up the file pool is logically intertwined. Without using file pool backup facilities or equivalent facilities that are aware of the interrelationships of the data, it is not possible to restore only a portion of the file pool. The whole file pool must be backed up or restored as a unit.

When file pool processing has stopped, use the non-file pool facility to make a copy of *all* minidisks in the file pool, including the file pool log and control minidisks. If the server is a combined SFS and CRR server, do not back up the CRR log minidisks. Also, make a copy of the minidisk containing the POOLDEF file. When you are done making the copy, you have a complete image of the file pool at a given point in time.

To restore a file pool, you would copy the data from the backup to the file pool minidisks. The entire file pool must be restored as a unit. If the entire file pool is not restored, subsequent file pool server processing is unpredictable. When the restore is complete, the file pool will be in the state it was in at the time the backup was made. All changes made to any data in the file pool since the time the backup was made are permanently lost.

When backing up and restoring data to storage groups that are managed by DFSMS/VM, you should keep in mind the data which resides in the DFSMS/VM repositories is an integral part of the primary file pool storage group that you are backing up or restoring. Therefore, when using non-file pool facilities, provisions should be made to keep the non-migrated and the migrated data in the storage group in sync. For more information on migrated data and DFSMS/VM, see *z/VM: DFSMS/VM Storage Administration*.





## Chapter 8. Security

Repository file pool servers, CRR recovery servers, and FIFO servers provide authorization mechanisms that allow you to control the security of the data they manage. In addition to this basic support, repository file pool servers allow you to use an external security manager (ESM). An external security manager is not applicable to dedicated CRR recovery servers or FIFO servers. The authorizations an external security manager provides may coexist with the authorizations of the repository file pool server or replace them.

### Note:

For a description of the possible combinations and types of file pool servers, see [Chapter 1, “File Pool Administration Overview,”](#) on page 3.

This chapter is divided into three parts:

- The first part describes the authorization mechanisms supplied with file pool servers.
- The second part discusses using an external security manager to augment or replace the file pool authorization mechanisms. (This discussion is applicable to only repository file pool servers, not CRR recovery servers or FIFO servers.) File pool servers do not include an external security manager. Rather, they include facilities to allow you to route certain requests to an external security manager for authorization checking. Before trying to use an external security manager, you should understand the authorization mechanisms supplied with the repository file pool servers.
- The third part discusses the programming interface a file pool server provides to an ESM. (This discussion is applicable to only repository file pool servers, not dedicated CRR recovery servers or FIFO servers.) This information is provided to assist the programmer who is developing or modifying an ESM to work with the repository file pool server.

## File Pool Authorizations and Permissions

For the set of user functions that require services from a file pool, the appropriate file pool is selected through path resolution or explicit user specification. There are two types of connections: SFS and BFS. The type of connection is dictated by the type of service that is called for or the type of object that is operated upon (or both). Connections are established implicitly. At any point in time there can be multiple connections to a file pool from a particular user virtual machine and each of these connections can be of type SFS or type BFS. Once connections are made, they are generally retained and reused for subsequent services of the associated type, as long as there is a user process to use them and the file pool server is able to perform the service.

Although SFS connections are used primarily for operations on SFS objects in SFS file spaces, there are some uses of SFS connections that go beyond this to include operations on the file pool in general or operations on BFS objects, as follows:

- File pool administration services that apply to the file pool in general, are routed on SFS connections to the appropriate file pool. These include ENROLL USER, (which can be used to create a BFS file space), RENAME USER (which can be used to rename a BFS file space), FILEPOOL BACKUP, FILEPOOL DISABLE, for example.
- File pool administrators can operate on BFS objects using a restricted subset of the normal SFS operations. These include OPEN, READ, WRITE, LOCK, and CLOSE of files, for example.
- Users who are not file pool administrators have a small set of read-only (query) SFS functions they can use against BFS objects. For example, QUERY LIMITS to get usage information on a BFS file space.

The authority or permission checking that takes place in the file pool server for user object access depends on the type of connection and the type of function used. For example, BFS permission checking with permission bits, UIDs, and GIDs are **NOT** used when SFS functions are used to operate on

## Authorizations and Permissions

BFS objects (involving SFS connections to the file pool server). The following table summarizes these authorizations/permissions.

*Table 11. Authorization or Permission Depending on Connection Type*

<b>Type of Function or Connection Used</b>	<b>Authorization or Permission</b>	<b>Object Scope</b>
SFS	Individual user - object owner or receiver of an explicit authorization grant for the object (or public). Authorization grants are done by the owner.	SFS Objects owned or for which authorization is explicitly granted
	File Pool Administrator - special authority for the whole file pool - objects, functions, and permission.	File Pool and all SFS and BFS objects therein
	ESM - can override all or part of the normal authorization mechanisms.	All or some operations on SFS
BFS	Object access levels (user, group, and public) and permissions within these levels (read, write, and execute) established by the owner (creator) of the object. Owner can change these at any time, including the ownership itself. Object access is based on matching the effective user ID, group ID, and supplementary GIDs of the process doing the access with the UID and GID of the object, then, where there is a match, allowing access permission according to the permission bits and type of function.	Current BFS Object
	File Pool Administrator - special authority for the whole file pool - objects, functions, and administration, regardless of the UID/GID values.	All BFS objects in the file pool *
	Superuser (UID=0). Special permission for all BFS objects and functions in the system, but not for file pool administration or SFS functions. Includes privileges of object owners.	All BFS objects in the system (scope of the z/VM system)
	ESM - can override all or part of the normal BFS permission mechanisms.	All or some operations on BFS objects

**Note:** \* There is a restriction for file pool servers which controls the file pool's capability for setting the UID on execute on behalf of a requesting user. The file pool administrator is subject to this same restriction. This capability depends on the corresponding setting in the z/VM directory entry for the associated file pool server virtual machine. It is also affected by a similar setting in the z/VM directory for the requesting virtual machine, for example the file pool administrator's machine.

The file pool administrator has BFS permissions only for BFS objects or operations in the file pool for which that person is the administrator, and this capability applies only where that file pool server is used.

With the exception of administration authority, all SFS authorizations and BFS permission related information related to a file pool are recorded in the catalog storage group within that file pool. The authorization scheme for one file pool is completely independent from the authorization scheme in another.

File pool administration authorizations are not recorded in the file pool. They are, instead, kept in virtual storage and disappear when the server is shut down or otherwise stops.

## Controlling Access to a File Pool Server Machine

The user who has the most power in a file pool is the user (or users) who knows the CP password of the server machine managing that file pool. Anyone who knows the CP password can log on the file pool server machine and access the file pool minidisks. The user can enter any FILESERV command against the file pool. If the user enters a FILESERV START command, the user can then enter any file pool server operator command.

In addition, the user can grant administration authority to any user ID, including his or her own user ID.

Authorization is keyed on a user ID, and, in the case of BFS, UID and GIDs, established internally when there is a connection to a file pool server. These IDs are based on those set in the connector's CP Directory. See *z/VM: CP Planning and Administration* for more information about the IDs.

When you connect in a collection, these IDs are unique and work just as if they were in the same system. However, when you connect to a file pool server in another system or collection through the network, mapping to another user ID (and the associated UID/GID in the case of BFS) may be necessary. In this case, see *z/VM: Connectivity* for more information.

**Note:** The following discussions assume that if the Single Console Image Facility (SCIF) defines a secondary user for the server, the person operating the secondary user console knows the password of the server machine. Otherwise, the user would be able to do everything described previously except log on the server machine. (The user can use the CP SEND command to process FILESERV commands and operator commands.) If it were necessary to log on the server machine (perhaps because the server is not automatically started or for various dedicated maintenance mode procedures), someone else would have to do it.

## Administration Authority

Having file pool administration authority is second only to knowing the CP password to the server machine. A user with administration authority can enter any of the file pool administration commands described in Part 3 of this document. Those commands require, however, that the file pool is available in multiple user mode, and a FILESERV START command can be entered only by the user who knows the CP password to the server or by the secondary user (using the CP SEND command).

Unless a user with administration authority also knows the CP password of the server (or has access to the secondary user console), the user is not able to enter any FILESERV command or file pool server operator command.

In addition to being able to enter the file pool administration commands, administration authority gives you quasi-ownership of every object in the file pool. You can do anything to directories, files, and other objects their owners can do. You cannot do something the owner cannot do. For example, you cannot create an SFS alias in a user's directory if the user himself did not have any authorization to the base file. (There is nothing to prevent you from granting that authority, however, and then creating the alias.)

The only user machine commands that are applicable to a CRR recovery server are:

- Administration:
  - DELETE ADMINISTRATOR
  - ENROLL ADMINISTRATOR
  - QUERY FILEPOOL AGENT
  - QUERY FILEPOOL COUNTER
  - QUERY FILEPOOL CRR
  - QUERY FILEPOOL MINIDISK
  - QUERY FILEPOOL OVERVIEW
  - QUERY FILEPOOL REPORT
  - QUERY FILEPOOL STATUS
- End Use (see *z/VM: CMS Commands and Utilities Reference* for more information on this command):

- QUERY FILEPOOL CONNECT.

Some important general concepts of administration authority are:

- An administrator has implicit authority to read from and write to objects owned by every user in a particular file pool.
- Administrators also have the ability to create and delete objects in any file space.
- Administration authority is transient. It lasts only until server processing ends or until the authority is revoked.
- Administration authority lets the user connect to the file pool even if the user is not enrolled through ENROLL USER, ENROLL PUBLIC, or has no BFS UID.
- If a user is granted administrator authority, the authority takes effect when the user initiates the next logical unit of work with the file pool.
- If a user's administrator authority is revoked, the change takes effect when any in-process logical unit of work ends and before the user begins the next logical unit of work.
- When administration authority is deleted from a user, any explicit locks acquired by the specified user ID will not be automatically deleted. The user can remove them if he or she is still explicitly enrolled or if PUBLIC is enrolled. Or, another file pool administrator can delete the locks.

When administrator authority is deleted, the user may find he or she cannot use the QUERY LOCK command to see locks he or she created. This happens if the user has no explicit authority on the object and was able to create the lock only by virtue of administration authority. After administration authority is revoked, the QUERY LOCK command does not display the locks for the user because the user has no authorization on the file or directory. Although the user cannot display the lock, the user can still enter a DELETE LOCK command to delete it.

The user that owns the locked object will be able to see the locks created by the ex-administrator (using the QUERY LOCK command), but will not be able to delete them. Users without administration authority cannot delete locks created by some other user. Any user that has administrator authority, however, will be able to see the locks and will be able to delete them.

Other facts about administration authority specifically for SFS:

- If you create an SFS file in someone else's directory, your user ID is not granted any access to that file. It is as though the owner of the directory created the file himself. If you lost your administration authority, you would lose access to the file. You could, however, grant yourself authority to the file after creating it. This authorization would persist even if you lose administration authority. It is, in effect, the same as if the owner had granted you authority.
- SFS file or directory owners can delete any authorities you have granted on their objects.
- If you have administration authority, but are not enrolled as an SFS user, you can still lock SFS files or directories. This lets you XEDIT files because XEDIT will try to lock them unless you explicitly tell XEDIT not to do so.
- If you want to create an SFS alias (in one of your own FILECONTROL directories) on a base file you do not own, you or the base file owner must first grant you authority on that base file. This will give you explicit authority on the base file. You may already have authority on that base file if you were given explicit authority on the file when the file was created. (For example, your user ID has NEWREAD or NEWWRITE on the directory). The implicit authority administrators have on users' files will not work when trying to create an alias.
- Likewise, if you are creating the SFS alias in another user's FILECONTROL directory, the user must be granted authority to read from the base file if that authorization has not been already granted.
- Having administrator authority does not imply you also automatically have an SFS file space in the file pool. A user can have administrator authority without having a file space. To get a file space, the administrator would simply enter an ENROLL USER command (with the BLOCKS option) for his or her own user ID.
- Only an owner of an SFS directory or an administrator can erase a directory. Owners (even owners who have administration authority) are not allowed to erase their own SFS directory if they already have the directory opened. An administrator can, however, erase some other user's SFS directory even if that

user has the directory opened. Therefore, directory owners should not assume their directories cannot be erased while they have them opened.

- If the administrator authority is revoked and the administrator had another user's SFS directory accessed for which he has no explicit authority, the directory will not be released when the administrator authority is revoked. The ability of the ex-administrator to access the directory remains in effect until the end of the CMS session. This includes being able to release and re-access the directory.

If the administrator has not accessed another user's SFS directory in the CMS session and his administrator authority is revoked, he will not be able to access the other user's directory, unless he has been granted specific authority.

- Administrators cannot revoke SFS authorities on files and directories from the users that own those files and directories.
- An administrator can revoke an SFS authority from himself or herself only if that authority was previously granted. (That is, you, as an administrator, cannot revoke read authority from yourself if you were never explicitly granted authority.)
- The SFS QUERY AUTHORITY command, which is described in [z/VM: CMS Commands and Utilities Reference](#), displays both implicit and explicit SFS authorizations for administrators. A second line for the same file or directory in the display indicates the administrator has been granted authority explicitly on that file or directory.

## Enrolling an Administrator

Before enrolling a file pool administrator, decide whether you want the administration authority to be temporary or permanent. Then follow the instructions in one of the two following sections.

### Temporarily Enrolling an Administrator

To temporarily enroll an administrator, use the ENROLL ADMINISTRATOR command. The ENROLL ADMINISTRATOR command must be issued from an administration machine. For example, to give BLAINE administration authority in the PROG1 file pool, you would enter:

```
enroll administrator blaine prog1
```

For a complete description, see [“ENROLL ADMINISTRATOR” on page 417](#).

The administration authority lasts until it is explicitly deleted or until server processing ends. See [“Deleting an Administrator” on page 142](#) for instructions.

### Permanently Enrolling an Administrator

To permanently enroll an administrator, specify the user ID in an ADMIN startup parameter. For example, to permanently enroll user ID MAEGAN as an administrator, specify the following in the DMSPARMS file:

```
ADMIN MAEGAN
```

Whenever file pool server processing is started, user MAEGAN will automatically be granted administration authority. If you no longer want MAEGAN to be a permanent administrator, simply remove the ADMIN startup parameter from the DMSPARMS file the next time file pool server processing is stopped.

If you want to immediately revoke MAEGAN's administration authority, use [“DELETE ADMINISTRATOR” on page 400](#). In this case, you still need to remove the ADMIN startup parameter for MAEGAN the next time server processing is stopped. Otherwise, MAEGAN will be given administration authority again when multiple user mode processing is started.

For a complete description of the ADMIN startup parameter, see [“Startup Parameter Descriptions” on page 342](#).

## Other Considerations for Administrators

This section is applicable to repository file pool servers, but not dedicated CRR recovery servers or dedicated FIFO servers because IBM does not recommend doing control data or user data backups for dedicated CRR recovery servers or FIFO servers.

To use the FILEPOOL BACKUP command to back up a storage group, the administration machine may need to have its MAXCONN value increased. The MAXCONN value is specified in the OPTION control statement of the VM system directory entries for the virtual machine. MAXCONN determines the maximum number of IUCV connections for a virtual machine. The FILEPOOL BACKUP, FILEPOOL CLEANUP, and FILEPOOL RESTORE commands use the CP DASD Block I/O system service for reading and writing file pool minidisks. CP DASD Block I/O uses IUCV. The FILEPOOL FILELOAD command, when used to restore a number of files simultaneously, uses a separate IUCV connection for each file.

Existing virtual machines may already have a MAXCONN value specified. If not specified, MAXCONN defaults to 64.

### Note:

The number of IUCV or APPC/VM connections a user machine can have to file pools for SFS type operations is limited to one half of the MAXCONN value. For example, if a user machine will need to have 50 active connections at the same time, you should set the MAXCONN value for this machine to 100.

You may also want to allocate a small *work* minidisk to the administration machine if it does not already have one. This minidisk would be needed for the administrator to back up or restore the storage group to which he or she is assigned (assuming, of course, the administrator is also an enrolled file pool user). The backup and restore operations for user storage groups (FILEPOOL BACKUP and FILEPOOL RESTORE commands) write temporary data (global variables) to a read/write file mode. During the backup or restore, users are unable to write to the storage group. So, if the administrator is backing up or restoring his or her own storage group, the administrator will need to temporarily access a minidisk as file mode A. The minidisk only needs to be a cylinder or two (1000 to 2000 blocks on FB-512 devices).

## What If I Forget the ADMIN Startup Parameter?

Another way to enroll an administrator is by using the GRANT ADMIN operator command. GRANT ADMIN is useful if you forget to specify an ADMIN startup parameter. In that case, the server is started without an initial administrator. The only way to enroll the first administrator (aside from stopping the server and specifying an ADMIN startup parameter), is to enter GRANT ADMIN. GRANT ADMIN must be entered from the server machine console (or the console of the secondary user).

Suppose, for example, SUE is the secondary user ID for the PROGSERV server machine. Sue started PROGSERV in multiple user mode, disconnected from the server machine and logged on to her own machine SUE. Then she realizes she forgot to specify ADMIN SUE in the DMSPARMS file for the server. Sue would enter the following command to grant herself administrator authority:

```
#cp send progserv grant admin sue
```

For a complete description, see [“GRANT ADMIN” on page 541](#).

## Deleting an Administrator

To delete an administrator, enter the DELETE ADMINISTRATOR command. The DELETE ADMINISTRATOR command must be issued from an administration machine. You cannot delete your own administration authority. For administrator SUE to delete administration authority from BLAINE in the PROG1 file pool, SUE would enter:

```
delete administrator blaine prog1
```

For a complete description, see [“DELETE ADMINISTRATOR” on page 400](#).

The REVOKE ADMIN operator command can also be used to revoke administration authority. The REVOKE ADMIN command must be issued at the file pool server machine console (or the console of its secondary user ID). To revoke administration authority from BLAINE at the server machine console, you would enter:

```
revoke admin blaine
```

For a complete description, see [“REVOKE ADMIN” on page 663](#).

**Note:** If you delete administration authority, but the user ID is specified on an ADMIN startup parameter, that user ID will be granted administration authority again the next time file pool server processing is started.

## Deleting a User or File Space

To delete a user or file space, use the DELETE USER command. See [“Deleting Enrolled Users and File Spaces” on page 93](#).

**Note:** If you delete a user or file space using the KEEPAUTH option of DELETE USER, the SFS authorizations granted to the deleted file space ID are kept. If the deleted file space ID is enrolled again at some later date, the new file space will immediately have access to all the data for which the previous owner of the file space ID was authorized. This could be a breach of your system's security. For a complete description, see [“DELETE USER” on page 408](#).

## Auditing Security

The security audit facility audits all types of file pool servers and many of their activities. That is, it applies to CRR and FIFO servers as well as to repository servers.

For repository file pool servers and FIFO servers, this facility audits attempts to access file pool resources.

For CRR, this facility audits the use of these operator commands that intervene in CRR activity:

- CRR recovery server operator command:
  - CRR ERASE LU
  - CRR ERASE LUWID
  - CRR RESYNC
- Repository file pool server operator commands:
  - ERASE LUNAME
  - FORCE PREPARED

The security audit facility writes a record in a CMS file, which makes it possible for you to know who is:

- Attempting to access file pool resources
- Issuing operator commands that intervene in CRR activity

You can use the server's security audit facility even if you are using an external security manager to check the authorizations. When an external security manager is used (the ESECURITY startup parameter is in effect), the server generates audit records similar to those it usually generates. The server merely records in the audit records what was doing the checking: the server itself, or an external security manager.

**Note:** External security managers are not applicable to dedicated CRR recovery servers or dedicated FIFO servers.

You can audit security at two levels: partial and complete. In a *partial audit*, these kinds of events are recorded:

- Unsuccessful attempts to access objects in the repository file pool server.
- All repository file pool server access attempts by users having file pool administration authority.
- All BFS access attempts by BFS superusers.
- Operator intervention into CRR activity.





If the audit file resides in another file pool, ensure there is adequate file space. Otherwise, there is a chance the server will stop when it tries to close the audit file. Any error during the close will cause a termination, but the most likely cause is that file space is exhausted. (Errors during writes do not cause the server to stop.)

If an error occurs when the server attempts to close the audit file, the target server will roll back the audit file updates. You lose all audit information because the audit file was opened.

To minimize the chance of a server termination and loss of audit data, allocate a high number of 4KB blocks to the file space in which the audit file will reside. The number of blocks should far exceed the anticipated size of your audit file. Also consider limiting write-sharing in that file space. Otherwise, other users could consume the blocks you had intended to be used for the audit file.

You should carefully protect the audit file, whether it is on tape or disk, from unauthorized access. Information about the existence of files and file ownership is recorded in the file. For a secure environment, this information should not be available to the user community.

## Starting and Stopping Audits

You can start an audit by specifying the AUDIT startup parameter or by entering an AUDIT operator command after the file pool server is started in multiple user mode.

When specified as a startup parameter, a partial audit starts as soon as the server begins running in multiple user mode. It is not possible to start a complete audit by using a startup parameter. The parameter is ignored when the server is started in dedicated maintenance mode. For more information on how to specify startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,” on page 339](#).

To start an audit while the file pool server is running in multiple user mode, use the AUDIT operator command. For a complete description, see [“AUDIT” on page 362](#). If you want to start a complete audit from the secondary user console of the VMSESRVU server machine, you would enter:

```
#cp send vmseervu audit on all
```

The command to start a partial audit in the same server machine is:

```
#cp send vmseervu audit on partial
```

The CRR recovery server operator command and file pool operator commands, which intervene in CRR activity, are recorded in both partial and complete audits. If you are done auditing the CRR recovery server operator command and now want to stop collecting these CRR audit records, enter:

```
#cp send vmseervu audit crr off
```

If at a later time, you want to continue auditing the CRR recovery server operator command and file pool operator commands, which intervene in CRR activity, enter:

```
#cp send vmseervu audit crr on
```

When stopping an audit, you have the option of closing the audit file or leaving it open. The only time you would leave the file open is if you intend to start another audit during that same execution of the file pool server. In that case, additional audit records are added to the end of the file. (The records previously generated are not erased.) When the file pool server is shut down, it automatically closes the audit file if it is open.

This command stops the audit in the VMSESRVU machine:

```
#cp send vmseervu audit off
```

The audit file is closed by default.

**Note:** The AUDIT command can stop an audit that was begun by a startup parameter.

## Formatting the Output of an Audit

The data in an audit file is illegible. To read it, use the FILEPOOL FORMAT AUDIT command.

The FILEPOOL FORMAT AUDIT command accepts as input any file pool server audit file. It produces another file containing formatted audit records. You can edit or modify this second file just as you would any other CMS file. The formatted output file should be carefully protected from unauthorized access, as should any print-outs made from it.

The FILEPOOL FORMAT AUDIT command lets you selectively format the audit records. That is, you do not have to format every record in the audit file. There are two ways to select audit records for processing:

1. Let the FILEPOOL FORMAT AUDIT command prompt you for selection criteria, which is described in [“Using FILEPOOL FORMAT AUDIT with Prompts”](#) on page 146.
2. Specify the selection criteria in a control file and make that file available to the FILEPOOL FORMAT AUDIT command as input. See [“Using a Control File for FILEPOOL FORMAT AUDIT”](#) on page 149 for instructions.

### Using FILEPOOL FORMAT AUDIT with Prompts

To selectively format an audit trace file using FILEPOOL FORMAT AUDIT prompts, do the following:

1. Enter a CMS FILEDEF command for the file containing the audit trace data. The FILEDEF command must associate the file with ddname INPUT. FILEPOOL FORMAT AUDIT processing assumes the file has a variable blocked records with a maximum record length of 1200. It also assumes a 4096-byte block size (RECFM VB LRECL 1200 BLOCK 4096). The file produced by the AUDIT operator command has these characteristics. Any LRECL, RECFM, or BLOCK values on the FILEDEF command are ignored. For example, if your security audit data is in SECAUD DATA A, you would enter:

```
filedef input disk secaud data a
```

2. Enter a CMS FILEDEF command for a file that is to contain the formatted audit records. Associate the file with ddname OUTPUT. The output file must be defined with a record format (RECFM) of FA or FBA and a record length (LRECL) of 133. (Any FILEDEF RECFM, LRECL, or BLOCK will be ignored by the formatter.) If, for example, you want the formatted audit records to be placed in the file FORMAUD DATA A, you would enter:

```
filedef output disk formaud data a (recfm fba lrecl 133
```

The FILEPOOL FORMAT AUDIT command places carriage control characters in the output file.

3. Now enter the FILEPOOL FORMAT AUDIT command:

```
filepool format audit
```

You see the following:

```
DMS3485I FILEPOOL processing begun at hh:mm:ss on day month year.
DMS3499R Enter 1 (if DDNAME=INPUTCTL is already available),
DMS3499R 9 (to quit audit file processing), or
DMS3499R just press enter if you want to be prompted for audit processing.
```

Press Enter to use the prompts. Enter 1 only if you have already:

- a. Created your own control file
- b. Entered a CMS FILEDEF command for ddname INPUTCTL for that file

See [“Using a Control File for FILEPOOL FORMAT AUDIT”](#) on page 149 for instructions on using a control file.

Next you see:

```
DMS3487R Enter AUDIT selections: 1 (All) or 2 (Select).
```

Enter 1 if you want all of the Audit file formatted. Enter 2 if you want to be prompted for selection criteria. If you enter 2, you next see:

```
DMS3450R Enter CRR selections: '1' to Ignore, '2' for Only, or
DMS3450R just press enter to skip this selection.
```

In response to this prompt, select 1 if you do not want to format CRR audit records or select 2 if you only want CRR audit records to be formatted. Regardless of how you respond, next you will see:

```
DMS3488R Enter up to 6 userids or
DMS3488R just press enter to skip this selection.
```

In response to this prompt, enter up to six user IDs (separated by one or more blanks). The user IDs are used in selecting audit records generated because of authorization checks caused by those user IDs. Suppose, for example, you respond:

```
chuck
```

FILEPOOL FORMAT AUDIT processing will format any audit records generated because of requests that Chuck made. (Chuck might, for example, try to open another user's file, in which case his authorization would be checked and could cause an audit trace record to be created.)

If you are not interested in the requests of any particular user ID, press Enter to select all requesting user IDs. Next you see:

```
DMS3489R Enter up to 6 ownerids or
DMS3489R just press enter to skip this selection.
```

In response to this prompt you can again enter up to file space names. Use one or more blanks between them. The names can identify SFS or BFS file spaces. In the case of an SFS file space, the name can be thought of as a ownerid. Therefore, the ids you enter represent the owners of the objects for which the authorization was checked. If, for example, you wanted to format all audit records produced because of access attempts on Eugene's objects, you would respond:

```
eugene
```

If you want to format all audit records produced because of access attempts on a certain BFS file space, enter the file space name.

If you are not interested in any particular owner or file space name, press Enter to select all file space names (owners). Next, you see:

```
DMS3490R Enter up to 10 authorization types or
DMS3490R just press enter to skip this selection.
```

Enter up to 10 numbers that represent the types of authorization you wish to select or just press Enter to select all authorization types. Notice the authorization types have slightly different meanings depending on the file space type. That's because search permission and execute permission are concepts applicable only to a BFS object. So, authority type number 4 means either read or search authority was required. Also, authority type 6 means either read or execute authority was required. The numbers are as follows:

Number	Meaning
--------	---------

- |   |   |
|---|---|
| 1 | File pool administration authority or BFS superuser access. |
| 2 | Object ownership authority.                                 |
| 3 | Write authority to the directory.                           |

- 4 Read or search authority to the directory.
- 5 Write authority to the file.
- 6 Read or execute authority to the file.

Although security audit includes authorities other than READ and WRITE authority (such as DIRREAD and DIRWRITE), all authorization checks done internally are simple verifications of a user's ability to read from or write to a file or a directory. When checking authorization, server processing does not care whether the user has, for example, READ or DIRREAD authority. It just needs to know whether the user can read the file.

The next prompt you see is:

```
DMS3491R Enter up to 10 file pool server function codes or
DMS3491R just press enter to skip this selection.
```

Enter up to 10 numbers that represent the types of file pool server function codes you wish to select or just press Enter to select all the function codes. The valid values you can enter are the same as those that can be specified on the FILEREQ keyword control statement. See [Table 12 on page 151](#) for a list of the function codes that pertain to SFS connections to SFS or BFS objects. See [Table 13 on page 156](#) for a list of the function codes that pertain to BFS connections to BFS objects.

The next prompt you see is:

```
DMS3492R Enter a date range (mm/dd/yy) or
DMS3492R just press enter to skip this selection.
```

Enter a data range in the form 'mm/dd/yy mm/dd/yy' or press Enter to select all dates. If one date is entered, only audit information for that specific date will be selected. There must be two digits for each portion of the date, enter leading zeros if needed. The separator character must be a '/'.

The next prompt you see is:

```
DMS3493R Enter a time range (hh:mm:ss) or
DMS3493R just press enter to skip this selection.
```

Enter a time range in the form 'hh:mm:ss hh:mm:ss' or press Enter without typing anything to select all times. If you specify a range, there must be two digits for each portion of the time; enter leading zeros if needed. The separator character must be a ':'.

The next prompt you see is:

```
DMS3494R Enter authority check results wanted:
DMS3494R 1 (Unsuccessful),
DMS3494R 2 (Successful),
DMS3494R 3 (Successful because of
special authority), or
DMS3494R just press enter to skip this selection.
```

Enter a 1 to select only unsuccessful audit authority checks. Enter a 2 to select all successful audit authority checks. Enter a 3 to select audit authority checks that were successful because of special authority. Enter a null to select all authority check results.

If you pressed ENTER without typing anything in response to all the above messages, you see:

```
DMS3498R You have not entered any special selections. Enter 9
(quit audit file processing) or just press enter to process all the
audit records.
```

In this case, you can either press Enter again to format all the audit records or enter a "9" to quit.

- After you respond to the prompts, the FILEPOOL FORMAT AUDIT command processes the audit records that meet your search criteria. Note that any record that meets *any* of the search arguments is formatted (the selectivity arguments are 'ORed').

See [“Output Format of FILEPOOL FORMAT AUDIT” on page 160](#) for an explanation of the output produced by FILEPOOL FORMAT AUDIT.

### Using a Control File for FILEPOOL FORMAT AUDIT

If you prefer not to be prompted for the FILEPOOL FORMAT AUDIT selectivity keywords, you can place those keywords in a CMS file. Instead of responding to prompts, you would have the FILEPOOL FORMAT AUDIT command process the file instead. Here's how to do it:

- Create a file that contains keyword control statements. The keywords are described later in [“Keyword Control Statements for FILEPOOL FORMAT AUDIT” on page 149](#). The file can reside in an accessed SFS directory or an accessed minidisk. It must have a record length of 80 and a fixed record format (LRECL 80 RECFM F).
- Enter a FILEDEF command to associate the control statement file with the ddname INPUTCTL. If, for example, you have named the file AUDIT CONTROL and the file resides on file mode A, you would enter:

```
filedef inputctl disk audit control a
```

- Enter the usual FILEDEF commands for the input and output files (ddnames INPUT and OUTPUT).
- Enter the FILEPOOL FORMAT AUDIT command. The command displays a message (DMS3499R) asking whether you want to use a control file. Respond "1" to the message. The audit file will then be formatted without further prompting.

Note that any record that meets *any* of the search arguments is formatted (the selectivity arguments are "ORed").

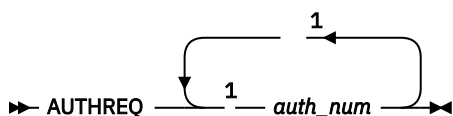
See [“Output Format of FILEPOOL FORMAT AUDIT” on page 160](#) for an explanation of the output produced by FILEPOOL FORMAT AUDIT.

### Keyword Control Statements for FILEPOOL FORMAT AUDIT

This section describes all the keyword control statements for the FILEPOOL FORMAT AUDIT command. The following general rules apply:

- Keyword control statements can be specified in any order.
- Only one control statement can be placed on a record.
- Control statements cannot span records.
- The records themselves must be fixed-format with a length of 80 (RECFM FIXED LRECL 80).
- Keywords can be preceded only by blank characters.
- You can leave any number of spaces between a keyword and its associated parameters.
- If there are duplicate control statements, the last one in the file overrides any preceding statements.
- If a control statement is omitted, the records selected for formatting are not limited by that particular category.

The keyword control statements follow:



Notes:

- <sup>1</sup> A maximum of 6 repetitions.

## Authorizations and Permissions

The AUTHREQ keyword control statement lets you format only those records that pertain to particular kinds of authorization checks. The AUTHREQ values represent authority that was needed by the requesting user.

For *auth\_num*, specify one of the following:

- 1** File pool administration authority.
- 2** Object ownership authority.
- 3** Write authority to the directory.
- 4** Read authority to the directory.
- 5** Write authority to the file.
- 6** Read authority to the file.

Separate the numbers with blanks.

Although security audit includes SFS authorities other than READ and WRITE authority (such as DIRREAD and DIRWRITE), all authorization checks done internally are simple verifications of a user's ability to read from or write to a file or a directory. When checking authorization, server processing does not care whether the user has, for example, READ or DIRREAD authority. It just needs to know whether the user can read the file.

The preceding numbers represent the authorizations that were required for successful access to the object. For example, if you wanted to format records generated for authorization checks for write authority to a file, you would select 5.

### ►► CRRIGNOR ◄◄

The CRRIGNOR keyword does not format records associated with CRR recovery server processing and SFS file pool operator commands that intervene in CRR activity. This keyword has no parameters.

### ►► CRRONLY ◄◄

The CRRONLY keyword lets you format and display only records associated with CRR recovery server processing and SFS file pool operator commands that intervene in CRR activity. This keyword has no parameters.

►► DATE — *mm/dd/yy* —————►  
  └── *mm/dd/yy* ─┘

For *mm/dd/yy*, choose the date or range of dates you want to select audit records for. If only the first date is specified, only records created on that date will be considered. If both dates are specified, only records created on the first date through records created on the second date will be considered. If DATE is omitted, the date is not considered in choosing the audit records to be formatted.

### ►► DUMPALL ◄◄

The DUMPALL keyword causes all audit records to be formatted. This keyword has no parameters. When DUMPALL is specified anywhere in the control file, all other keyword control statements are ignored.

►► FILEREQ ———— *req* —————►  
  └── 1 ─┘

Notes:

<sup>1</sup> A maximum of 10 repetitions.

The FILEREQ keyword control statement lets you format only those records that pertain to particular requests made to the file pool server. There is a family of file pool requests that originate from SFS connections and are meant to operate on SFS or BFS objects. These requests are subject to SFS authority rules. These requests are listed in [Table 12 on page 151](#). There is a second family of file pool requests that originate from BFS connections and are meant to operate on BFS. These requests are subject to BFS authority rules as dictated by the POSIX standard. These requests are listed in [Table 13 on page 156](#). Notice that the function codes are not shared between the requests in the two tables. For *req*, specify one of the codes shown in either one of the tables.

Depending on what the user was trying to do, different authorization checks may be needed for a given request. For instance, an OPEN request would require a read authority check if the user was trying to read a file. A write authority check would be required, however, if the user were trying to open an existing file for write, and did not own the directory in which the file resided.

Authorizations for DIRCONTROL directories and files that reside in them are not necessarily checked on every request. For example, if a user accesses a DIRCONTROL directory in read/write status, the server verifies the user's ability to write at access time. On subsequent file OPEN requests, the server does not check the authorization—it already determined at access time the user has DIRWRITE authority.

[Table 12 on page 151](#) shows the file pool request codes you can specify for the FILEREQ keyword control statement, and the kinds of requests they represent. For each request, the various authorizations that might be checked are listed. Note that a request for a particular file pool service can be caused in many ways. For example, “lock requests” can be caused by a CREATE LOCK command, DISABLE operator command, or the DMSCRLC Callable Services Library routine. The server does not distinguish between any of the three.

Table 12. File Pool Request Codes (SFS)

Decimal Code	Hex Code	Meaning
16	X'10'	Cache release request Read authority to the directory being cache released. <b>Note:</b> A <i>cache release request</i> is a request to the server to stop sending cache updates to a user machine. The <i>cache</i> in a user machine contains information about the user's accessed directories. The server sends updates to the user's machine as needed to keep the user's cache current. While the updating of a cache can be suspended for many reasons (the user logs off, the APPC/VM link to the user is severed), authorization is checked only for explicit requests from the user's machine. CMS decides whether to enter cache release requests to the server depending on various internal conditions in the user's machine. The issuing of the request is not directly related to CMS commands or applications the user might be running.
21	X'15'	Get directory entry request 1. Read authorization to the parent directory 2. Write authorization to object 3. Read authorization to object <b>Note:</b> A <i>get directory entry request</i> is a request for the server to provide information about a single directory entry. (This request is made when the DMSEXIST Exist CSL routine is processed in a user machine.)

Table 12. File Pool Request Codes (SFS) (continued)

Decimal Code	Hex Code	Meaning
22	X'16'	<p>Lock request</p> <ol style="list-style-type: none"> <li>1. Administrator authority</li> <li>2. Read authorization to the file/directory being explicitly locked for SHARE use</li> <li>3. Write authorization to the file/directory being explicitly locked for UPDATE or EXCLUSIVE use</li> <li>4. Administrator authority to lock storage group</li> <li>5. Administrator authority to lock file space</li> <li>6. Ownership of file space to be locked</li> </ol> <p><b>Note:</b> A <i>lock request</i> is a request for the server to lock an object in the file pool. (Note that DISABLE operator commands are considered lock requests.)</p>
23	X'17'	<p>Open file request</p> <ol style="list-style-type: none"> <li>1. Ownership of the directory in which the file resides</li> <li>2. Read authorization to the base file for 'open read'</li> <li>3. Write authorization to the base file for 'open write' and 'open replace'</li> <li>4. Write authorization to the directory of the new file for 'open new'</li> </ol> <p><b>Note:</b> An <i>open file request</i> is made for both opening a file and creating an empty file. (An empty file is created when issuing a CREATE FILE command or executing the DMSCRFIL CSL routine.)</p>
24	X'18'	<p>Open directory request</p> <ol style="list-style-type: none"> <li>1. Administrator authority for query lock</li> <li>2. Read authority to the directory/file</li> </ol>
26	X'1A'	<p>Refresh directory request</p> <p>Read authorization to the directory</p> <p><b>Note:</b> This request is made when the ACCESS command is entered.</p>
28	X'1C'	<p>Unlock request</p> <p>Administrator authority</p>
32	X'20'	<p>Connect request</p> <p>Administrator authority. If unsuccessful, implies no connect authority.</p> <p><b>Note:</b> This request is made when any command or CSL routine is entered to the SFS file pool by a user who is not enrolled in the file pool. There is usually only one per connection.</p>
33	X'21'	<p>Create directory request</p> <p>Ownership of the parent directory</p>



Table 12. File Pool Request Codes (SFS) (continued)

Decimal Code	Hex Code	Meaning
34	X'22'	Create alias request 1. Ownership of the directory in which the alias is being created 2. Read authorization to the base file 3. Read authorization to the base file for the owner of the directory in which the alias is being created 4. Write authorization to the directory in which the alias is being created
35	X'23'	Delete request 1. Write authorization to the directory which contains the file/alias 2. Write authorization to the file being deleted 3. Read authorization to the base file of the alias being deleted
36	X'24'	Delete directory Ownership of the directory being deleted
37	X'25'	File copy 1. Read authorization to source file 2. Write authorization to the target file being replaced 3. Ownership of the target file 4. Write authorization to the target directory
38	X'26'	Grant Ownership of the directory/file
39	X'27'	Relocate Ownership of the directories in which the object is being relocated
40	X'28'	Rename Ownership of the object being renamed
41	X'29'	Revoke Ownership of the object being revoked
44	X'2C'	Change attribute Write authorization to the file being changed <b>Note:</b> This request is made when the FILEATTR command is entered or the DMSCATTR CSL routine is processed.
48	X'30'	Add storage Administrator authority to add storage <b>Note:</b> This request is made when the MODIFY USER command is entered.
49	X'31'	Change threshold Administrator authority

## Authorizations and Permissions

Table 12. File Pool Request Codes (SFS) (continued)

Decimal Code	Hex Code	Meaning
50	X'32'	Delete storage Administrator authority
51	X'33'	Grant administrator authority Administrator authority
52	X'34'	Grant user Administrator authority
53	X'35'	Query Administrator Request Administrator authority
56	X'38'	Query Lock Conflicts Request Administrator authority
57	X'39'	Query file pool request Administrator authority
58	X'3A'	Query user space request Administrator authority
59	X'3B'	Revoke Administrator Authorization Request Administrator authority
60	X'3C'	Revoke user request Administrator authority
61	X'3D'	Write accounting request Administrator authority
65	X'41'	Media recovery close catalog request 1. Read authorization to object 2. Write authorization to object
66	X'42'	Media recovery open catalog request Administration authority
69	X'45'	Release blocks request Administrator authority
70	X'46'	DATASPACE command request Administrator authority
71	X'47'	Change directory attribute request (DIRATTR) 1. Administrator authority 2. Ownership of the directory

Table 12. File Pool Request Codes (SFS) (continued)

Decimal Code	Hex Code	Meaning
72	X'48'	Query accessors request 1. Administrator authority 2. Ownership of the directory 3. Write authority to the directory
73	X'48'	Query data space request Administrator authority
96	X'60'	Change DRA Administrator authority <b>Note:</b> This request is used by DFSMS/VM.
97	X'61'	Create External Object 1. Write access to the directory 2. Alter access to the new file (external object) <b>Note:</b> This request is made when the DMSCROB CSL routine is processed.
99	X'63'	Query User Storage Group request Administrator authority <b>Note:</b> This request is made when the DMSQUSG CSL routine is processed.
101	X'65'	Connect user Administrator authority <b>Note:</b> This request is made for initial server requests on work units having assigned user IDs by means of the DMSGETWU CSL routine.
102	X'66'	Add minidisk Administrator authority <b>Note:</b> This request is made when the FILEPOOL MINIDISK command is entered.
103	X'67'	Control backup Administrator authority <b>Note:</b> This request is made when the FILEPOOL CONTROL BACKUP command is entered.
104	X'68'	Rename Userid Request Administrator authority <b>Note:</b> This request is made when the FILEPOOL RENAME command is entered.

The following table summarizes the permission rules implemented by the Byte File Server. The rules are defined by the POSIX standard. Some terminology that appears in the table, needs some explanation:

- The user is said to have "appropriate privilege" if either:
  1. The process is a superuser, which is defined as UID=0.
  2. The user is an SFS administrator.

## Authorizations and Permissions

If the user is determined to have appropriate privilege, then in **most** cases the user is allowed to perform the specified function, regardless of the permission bit settings. There are a few exceptions which are noted in the following section.

- The UID and GID used in the permission testing are the effective UID and GID except in one case. Only `access()` uses the real UID and real GID.
- The term "parent directory" means the one immediately preceding the target component.

Table 13. File Pool Request Codes (BFS)

Decimal Code	Hex Code	Meaning
193	X'C1'	<p>access</p> <p>This function tests the accessibility for the permissions indicated by an inputted <code>amode</code> parameter.</p> <p>There are two aspects of the access request that relate to permission checking. The first is determining if the user is permitted to perform the access against the specified object. The permission required is:</p> <ol style="list-style-type: none"> <li>1. Search permission on path prefix. This test is performed using the real UID and GID.</li> </ol> <p>The second is determining the result of the access request. The intent is to check the accessibility indicated in the <code>amode</code> field against the specified object. The client can test for any combination of R, W, or X. Failure is indicated if any one of them is not permitted. Tests are done using the real UID and the real GID. This aspect of access is not considered an auditable action. That is, the results of the test are not audited.</p>
194	X'C2'	<p>Catalog utility</p> <p>This request is intended for use by internal tools that perform file pool catalog diagnostics and maintenance. It is not part of the POSIX standard and there is no API for it. The permission required is:</p> <ol style="list-style-type: none"> <li>1. SFS administrator authority. (Superuser is not sufficient.)</li> </ol>
195	X'C3'	<p>Chaudit - Change Audit flags</p> <p>The auditor flags may be used by the ESM. They are not used by the file pool server. The request is provided only as a tool for ESM writers.</p> <p>The permission required to change the user's audit options:</p> <ol style="list-style-type: none"> <li>1. Search permission on path prefix and</li> <li>2. File owner (Effective UID of calling process matches owner UID).</li> </ol> <p>The permission required to change the auditor's audit options:</p> <ol style="list-style-type: none"> <li>1. appropriate privilege</li> </ol>
196	X'C4'	<p>Chmod - Change File Mode</p> <p>The required permission is:</p> <ol style="list-style-type: none"> <li>1. Search permission on path prefix and</li> <li>2. File owner (Effective UID of calling process matches owner UID).</li> </ol>

Table 13. File Pool Request Codes (BFS) (continued)

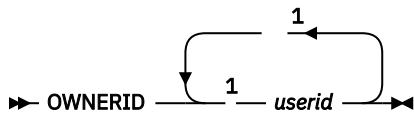
Decimal Code	Hex Code	Meaning
197	X'C5'	<p>Chown - Change Owner</p> <p>The required permission is:</p> <ol style="list-style-type: none"> <li>1. Search permission on path prefix</li> <li>2. To change owner UID, the calling process must have appropriate privileges</li> <li>3. To change owner GID, the calling process must have appropriate privileges, or all the following must be true:                             <ol style="list-style-type: none"> <li>a. Effective uid of caller equals the uid of the file AND</li> <li>b. The new uid specified equals the uid of the file AND</li> <li>c. The new gid specified equals the calling process effective GID, or equals one of its supplementary GIDs.</li> </ol> </li> </ol> <p>These rules conform to POSIX_CHOWN_RESTRICTED = on.</p>
199	X'C7'	<p>Lookup - Lookup an object in the catalogs</p> <p>Lookup is used extensively by the client to obtain file attributes in preparation for a variety of syscalls. It is also used for the stat() syscall.</p> <p>The required permission is:</p> <ol style="list-style-type: none"> <li>1. Search permission on path prefix</li> </ol>
200	X'C8'	<p>Mkcat - Make catalog entry</p> <p>The required permission is:</p> <ol style="list-style-type: none"> <li>1. SFS administrator authority. (Superuser is not sufficient.)</li> </ol>
202	X'CA'	<p>Open - Open a file</p> <p>This request is used to open all objects except directories.</p> <p>The required permission is:</p> <ol style="list-style-type: none"> <li>1. Search permission on path prefix</li> <li>2. If the file exists, permissions corresponding to the type of open must exist. That is, if the file is being opened for read, the caller must have read permission on the file. If the file is being opened write or truncate, the caller must have write permission. If the file is being opened for execute, the caller must have execute permission.</li> <li>3. If file does not exist, the caller must have write permission to the parent directory.</li> <li>4. Open_Truncate must have write permission on object.</li> </ol>
203	X'CB'	<p>Opendir - Open directory</p> <p>The required permission is:</p> <ol style="list-style-type: none"> <li>1. Search permission on path prefix and</li> <li>2. Read permission on specified directory.</li> </ol>

*Table 13. File Pool Request Codes (BFS) (continued)*

<b>Decimal Code</b>	<b>Hex Code</b>	<b>Meaning</b>
204	X'CC'	Readlink - Read contents of a soft link The required permission is: <ol style="list-style-type: none"> <li>1. Search permission on path prefix and</li> <li>2. If the object is an external link, read permission is required on the external link. If the object is a symbolic link, no permission checking is performed on the symbolic link. Symbolic links do not have permissions.</li> </ol>
205	X'CD'	Rename - Rename any object Path1 is the old name, path2 is the new name The required permission is: <ol style="list-style-type: none"> <li>1. Appropriate privilege or</li> <li>2. Search permission on path prefix of path1, and</li> <li>3. Search permission on path prefix of path2, and</li> <li>4. Write permission to the directory containing path1, and</li> <li>5. If the new name already exists, write permission to the directory containing path2.</li> </ol>
206	X'CE'	Rmdir - Remove directory The required permission is: <ol style="list-style-type: none"> <li>1. Search permission on path prefix and</li> <li>2. Write permission to the parent of the removed directory.</li> </ol>
207	X'CF'	Softlink - Create a symbolic or external link The required permission is: <ol style="list-style-type: none"> <li>1. Search permission on path prefix and</li> <li>2. Write permission to the parent directory.</li> </ol>
208	X'D0'	Unlink - Delete a hard link The required permission is: <ol style="list-style-type: none"> <li>1. Search permission on path prefix and</li> <li>2. Write permission to the parent directory.</li> </ol>
209	X'D1'	Utime - Update time stamps The required permission is: <ol style="list-style-type: none"> <li>1. Search permission on path prefix and</li> <li>2. If times parameter is null (that is, times are set to current time), the caller must be the owner of the file or have write permission.</li> <li>3. If times is not null, the caller must be the owner of the file.</li> </ol>

Table 13. File Pool Request Codes (BFS) (continued)

Decimal Code	Hex Code	Meaning
211	X'D3'	<p>Putime - pipe utime</p> <p>This request is used only by the FIFO server to communicate with the primary server. Putime is designed to update time stamps relative to the client that is performing a close of a named pipe. The close itself was directed to the FIFO server, so it is the FIFO server that is issuing the putime to the primary server. The primary server can have only one pipe server, but the userid of that pipe server is not known to the primary. Therefore, since it would be inappropriate to bypass permission checking totally for this request, the permission tested in the primary server is:</p> <ol style="list-style-type: none"> <li>1. appropriate privilege</li> </ol>



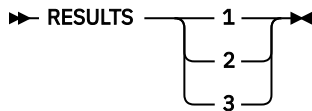
Notes:

<sup>1</sup> A maximum of 6 repetitions.

The OWNERID keyword control statement lets you format audit records only for accesses to objects owned by specific users. You could, for example, determine who attempted to access user CHUCK's objects by specifying:

```
OWNERID CHUCK
```

You can specify up to six owners. Separate each of their user IDs by blanks.



This keyword control statement causes FILEPOOL FORMAT AUDIT to format only those records that either were generated as a result of:

- 1 Unsuccessful request
- 2 Successful request
- 3 Request that was successful only because of special authority

If RESULTS is omitted, the reason why the record was generated is not considered in choosing the records to be formatted.

```
TIME — hh:mm:ss — hh:mm:ss →
```

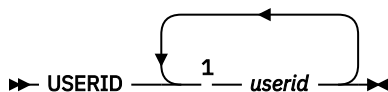
This keyword control statement causes FILEPOOL FORMAT AUDIT to format only those records that were generated during a particular time period. Both a starting time and an ending time must be specified, using 24-hour clock notation.

## Authorizations and Permissions

There must be two digits for each portion of the time. Enter leading zeros if necessary (for example, 01:01:00). Use a colon (:) to separate the hours, minutes, and seconds.

Specifying a time interval that passes through midnight must be done in two different executions of the FILEPOOL FORMAT AUDIT command.

If TIME is omitted, the time value is not considered in choosing the audit records to be formatted.



Notes:

<sup>1</sup> A maximum of 6 repetitions.

The USERID keyword control statement lets you format the audit records only for specific user IDs. You can specify up to six user IDs. Separate the user IDs by blanks.

If USERID is omitted, user ID values will not be considered in choosing the audit records to be formatted.

### Output Format of FILEPOOL FORMAT AUDIT

The output to DDNAME=OUTPUT is similar to the following. The first output record shown below (beginning with AUDITPOINT) is split onto two lines so it will fit on the page.

```
AUDITPOINT=nnnnn AUTHORITY REQUIRED=nnn FP FUNCTION CODE=nnn AUTHORITY CHECKING DONE
      BY=nnn DATE=mm/dd/yy
TIME=hh:mm:ss
RESULTS OF AUTHORITY CHECK=snnnnn USERID OF REQUESTOR=xxxxxxx OID OF RESOURCE=xxxxxx
QUALIFIED OBJECT NAME=filename filetype dirname

FORCE PREPARED      FILEPOOL:filepool LUWID:nnnnnnnn.11111111.iiiiiiiiiii.ssss
TOKEN:ttttttt
mm/dd/yy hh:mm:ss REQUESTING USERID:userid COMMIT END USERID:userid RESULT:xxxxxxx
Transaction tag:transaction_tag_field
```

Note the field names are not available in national languages other than American English. They are displayed only as shown above regardless of the language used on your system. The meanings of the fields are as follows:

#### **AUDITPOINT=nnnnn**

The value *nnnnn* in this field is an internal audit point number (of value only to service personnel).

#### **AUTHORITY REQUIRED=nnn**

*nnn* is a number representing the authorization required to perform the requested function. These numbers have the same meanings as the numbers you supplied in response to the prompt to supply authorization types or those supplied on the AUTHREQ keyword control statement.

#### **FP FUNCTION CODE=nnn**

*nnn* is a number representing the file pool server function that was being requested. These numbers have the same meanings as the numbers you supplied in response to the prompt for server function codes or those supplied on the FILEREQ keyword control statement.

#### **AUTHORITY CHECKING DONE BY=nnn**

*nnn* can be either 0 or 1. A value of 0 indicates the file pool server checked the authorization. A value of 1 indicates an external security manager did the checking.

#### **DATE=mm/dd/yy**

is the date the request was made.

#### **TIME=hh:mm:ss**

is the time the request was made.

#### **RESULTS OF AUTHORITY CHECK=snnnnn**

*snnnnn* can be either 0, -1, or 4. A value of 0 means the user was authorized to make the request. A value of -1 means the user was not authorized to make the request. A value of 4 indicates the request was made by a user having file pool administration authority.



**USERID OF REQUESTOR=xxxxxxx**

identifies the user ID making the request.

**OID OF RESOURCE=xxxxxx**

is a number used internally to identify the object the requestor was trying to access. This is intended for use by service personnel.

**QUALIFIED OBJECT NAME=filename filetype dirname**

This is the fully qualified name of the object the requestor was trying to use.

**FORCE PREPARED**

is the command entered and is being traced.

**FILEPOOL:filepool**

*filepool* identifies the file pool that contains the prepared work to be forced.

**LUWID:nnnnnnnn.lllllll.iiiiiiiiiii.ssss**

identifies the logical unit of work that is being forced. The LUWID consists of:

- *nnnnnnnn.lllllll* is the fully qualified LU name consisting of:
  - *nnnnnnnn* is the optional SNA network ID.
  - *lllllll* is the LU name.
- *iiiiiiiiiii* is the instance number.
- *ssss* is the sequence number.

**TOKEN:ttttttt**

identifies the SFS transaction recovery token and is used by SFS to identify the LUWID to the CRR recovery server. This is the same thing displayed as the "Recovery Token" field in the output of the QUERY PREPARED, CRR QUERY LU, and CRR QUERY LUWID commands.

**mm/dd/yy**

is the date when the FORCE PREPARED command was entered.

**hh:mm:ss**

is the time when the FORCE PREPARED command was entered.

**REQUESTING USERID:userid**

identifies the user ID issuing the FORCE PREPARED command.

**COMMIT**

identifies the heuristic action taken by the SFS operator, which could be COMMIT or BACKOUT.

**END USERID:userid**

identifies the user ID of the end user that was forced.

**RESULT:xxxxx**

is an SFS reason code. See *z/VM: CMS and REXX/VM Messages and Codes* for more information.

**Transaction tag:transaction\_tag\_field**

*transaction\_tag\_field* is a 1-80 byte field supplied by the application program that describes the transaction.

## Using an External Security Manager

---

The file pool server lets you augment or replace its security manager with your own. A security manager that is not part of the file server code included with z/VM is called an external security manager (ESM).

**Note:** ESMs deal only with file pool server activity, not CRR recovery server activity or FIFO server activity.

If an ESM is installed on your system, you can have a file pool server use it by specifying the ESECURITY startup parameter in the server's DMSPARMS file. The type of authority checking routed to the ESM is determined by how certain flags are set in the DMSESM PROFILE, discussed later in this chapter. For a description of the startup parameter, see [ESECURITY](#).

**Note:** ESMs are not used in dedicated maintenance mode. If the ESECURITY startup parameter is specified for a dedicated maintenance mode command, it is ignored.

## What Is a Security Manager?

A security manager is a program that controls file access based on the authorizations it knows about. The file pool server actually contains two security managers. One for SFS objects and one for BFS objects.

For SFS objects, the security manager is a program that maintains a list of files and directories along with the user IDs that are allowed to read from or write to those objects. To create and maintain that list, the security manager must let users (or administrators) grant and revoke authorities. In addition, it allows a server machine to determine whether a user ID is authorized to access an object.

The security manager included in the file pool server maintains an authorization list in the file pool catalogs. The CMS GRANT and REVOKE commands and corresponding CSL routines let users create and delete entries in the authorization list. When the server needs to check authority in the course of processing a user request, it calls its security manager to check authorization.

Although SFS supports many different authorities (READ, WRITE, NEWREAD, NEWWRITE, DIRREAD, DIRWRITE), it does not require an external security manager to support all of them. SFS file pool server processing distills its authorization checking into simple verifications of a user's ability to read from or write to a file or a directory. When checking authorization, server processing does not care whether the user has, for example, READ or DIRREAD authority. It just needs to know whether the user can read the file.

For BFS objects, the file pool server maintains permission bits, owning UID and GID for each object. The BFS security manager uses this data to determine if a given client is authorized to access a given file in a particular manner.

## What an ESM Does Not Protect

An ESM does not, by default, protect all security aspects of file pool server processing. The server does its own checking in two areas: attempted user connections and administration requests.

### Connect Verification

The server maintains a list of enrolled file pool users and the amount of blocks they are allowed to consume. When a user tries to connect to a file pool, the server allows the connection if any of the following are true:

1. The user is enrolled in the file pool.
2. PUBLIC connect authority has been granted.
3. The user is an SFS administrator.
4. The user has been assigned a UID, other than the default UID.

If none of these are true, the connection is rejected.

This verification process is always done by file pool server processing. The ESM is not used for connect verification.

### Administration Requests

Some functions that require administration authority are checked only by file pool server processing, while others are checked by the ESM (if the ESECURITY startup parameter is in effect).

An administrator's use of SFS functions can be separated into two categories:

1. Functions that explicitly require administration authority
2. All other access attempts

The SFS administration commands and the equivalent CSL routines are in the first category. Those commands and CSL routines explicitly require administration authority to process successfully. File pool server processing always verifies the user ID has administration authority for those commands and CSL routines. The ESM is used if command authorizations are being checked. (See [“DMSESM PROFILE” on page 164](#) for more information.)

In the second category, however, the ESM is always used. The functions in this category are available to general users, but the administrator may be using the function without being explicitly authorized. That is, the administrator is using the function by virtue of his or her administration authority.

For example, if a user enters a QUERY FILEPOOL REPORT command with the CATALOG option, the file pool server verifies the user is authorized to enter the command. Because the QUERY FILEPOOL REPORT command with the CATALOG option requires administration authority, the external security manager may not be used, even if ESECURITY is in effect. If, on the other hand, a user tries to read another user's file, it is the external security manager that verifies the user is authorized—not the file pool server. In this case, the external security manager is used even if the user making the request has file pool administration authority in the server.

When the server asks the external security manager to check authorization, the server accepts whatever the external security manager tells it. The external security manager can tell the server the user is authorized or not authorized. It can also tell the server it wants the server to check the authorization. In this latter case, the server will check to see whether the user is authorized. If the user has the proper authorization (read, write, or administration authority) the server processes the request.

**Note:** Because the server calls the ESM to check authorization for many access attempts, you can use an ESM to limit the scope of administration authority.

## How an ESM Affects Users

### SFS ESM

When ESECURITY is in effect, CMS commands that display the authorization status of files or directories indicate whether the object is protected by the ESM. For example, in the QUERY AUTHORITY command output, a "P" is displayed if the object is protected by an ESM.

These commands do not indicate what authority a user might hold in the ESM, however. That is, a user cannot tell whether read or write authority is held on the object. The user can only tell the object is protected. If users want to determine what authorities they hold in the ESM, they must use commands provided by the ESM.

Users may also need to learn new authorization rules. For example, when a user creates a file in another user's directory, the server automatically grants read and write authority on that file to the creator. Such is not the case with ESMs.

The server does not grant or revoke ESM authorities. A user who creates a file in another user's directory does not have any authority on that file in the ESM authorization records. Users or administrators must process some ESM function to grant authority on files.

SFS provides ESM user data support that can extend ESM function beyond rudimentary authorization checking. An ESM might use the ESM user data to support functions such as password protection of files and directories. These functions can then be exploited by application programmers who use the ESM.

The ESM user data support has two parts: a CSL interface for application programmers, and a server interface to the ESM. The CSL interface consists of a general-purpose CSL routine (DMSUDATA), optional *userdata* parameters on other key CSL routines, and an information name available through DMSERP (the extract/replace CSL routine). The server interface is a simple data structure the server builds and passes to the ESM. The ESM responds to the server by setting a return code field before returning to the server.

Because the server does not operate on the ESM user data, the ESM can define the form and content. For example, the ESM might expect a password to be in the ESM user data. If the password does not match that defined for the file, the ESM can reject the operation (that is, return a *not authorized* result to the program).

Although the ESM user data support is well-suited for password protection, it is not limited to that function. For example, the ESM might perform some secondary function not related to security, such as starting a program in another virtual machine, whenever a keyword is received as part of the user data. See the ESM documentation for information on how the user data is used.

## BFS ESM

When there is an ESM for BFS objects, the permission bits, the owning UID, and the owning GID are stored in the catalogs, the same as they are when there is no ESM. Commands that display the permission bits, do so from these values in the catalogs. That means these commands do not display any particular value to indicate the object is protected by an ESM. There is no facility provided for this.

Also, since the ESM can define its own authorization rules, users may also need to learn the new rules.

ESM user data is not provided for BFS connections to BFS file spaces. That means the application programming interface does not provide the optional *userdata* parameter as described above.

## DMSESM PROFILE

When a file pool server is started for multiple user access, if the server's DMSPARMS file contains the ESECURITY startup parameter, the server looks for a file called DMSESM PROFILE. The DMSESM PROFILE indicates the types of authorization checking to be routed to the ESM and specifies the names of the CSL routines the server calls.

The DMSESM PROFILE contains three types of records:

- Initialization and termination routine.
- Types of requests to be reviewed and the name of the CSL routine the server should call to check those requests.
- Specific file pool requests to be reviewed. This record applies to SFS connections only.

You can tailor the DMSESM PROFILE to meet the needs of your installation. The profile contains flags you can set to define the type of SFS authority checking you want the ESM to do. These flags are described in the following sections. [Figure 20 on page 164](#) shows the DMSESM PROFILE supplied with z/VM.

```
P0 DMSSECIT
A0012 DMSUAUTH B0000 DMSAAUTH C00 DMSOAUTH D1 DMSSECIT <E0 DMSPERM>
A002301 A002401
```

*Figure 20. IBM-supplied DMSESM PROFILE*

You can also customize the ESM interface by modifying or replacing the IBM-supplied CSL routines. (An ESM might supply its own replacements.) For more information, see [“Writing Your Own ESM Exit Routines” on page 176](#).

### Record 1: Initialization and Termination Routine

The first line of the DMSESM PROFILE contains the name of the ESM initialization and termination routine. There is one routine that handles both SFS connections and BFS connections. The line has the following format:

```
Pa init_routine
```

**a**

is a flag that indicates whether the ESM is required to protect all system objects (an object is a named thing that can be protected, such as a file, external object, directory, or alias):

**0**

NO, the ESM can defer some access decisions back to the usual server authorizations. This is the setting in the supplied profile.

**1**

YES, the ESM must protect all system objects. If the ESM attempts to defer a request back to the server, it is treated as a rejected authorization.

***init\_routine***

is the name of the CSL routine the server calls to initialize or stop ESM processing. The routine specified in the supplied profile is DMSSECIT.

**Note:** The IBM-supplied DMSSECIT routine calls a module called RPIUCMS to initialize the ESM. However, RPIUCMS is not supplied with z/VM. This module is either supplied by the ESM or written by the installation. If RPIUCMS is missing or cannot be called, server processing ends with an error.

**Record 2: Types of Calls to Be Reviewed**

The second line of the DMSESM PROFILE identifies the types of authorization calls the ESM should review. The first three tokens specified in this record apply only to SFS authorizations. They do not apply to BFS permissions. The fourth token, the ESM program check routine applies in all cases. The fifth token applies only to BFS permissions. Four types of calls can be directed to the ESM:

- SFS object authorization check
- SFS command authorization check
- SFS operator command authorization check
- ESM program check
- BFS permission checking

For each type of call, there is a token containing flags to indicate the extent of the checking to be done. The token is followed by the name of the CSL routine the server calls for that type of authorization checking. The line has the following format:

```
Almno obj_routine Bpqrs cmd_routine Ctu op_routine Dv prog_routine
```

***Almno***

is the token containing flags for an SFS object authorization call:

***l***

indicates whether the ESM should check new file, external object, and directory names as they are created. Usually, when you create a new file, the system checks only to see if you have write authority in the directory. The possible settings are:

**0**

NO. This is the setting in the supplied profile.

**1**

YES.

***m***

indicates whether the ESM should check alias names as they are created, changed, or deleted:

**0**

NO. This is the setting in the supplied profile.

**1**

YES.

***n***

indicates whether SFS calls the ESM for object authorization checking.

**0**

NO.

**1**

YES. This is the setting in the supplied profile.

**2**

SFS calls the ESM only for specific file pool requests, identified by tokens in the third and following lines of the profile.

***o***  
indicates whether SFS calls the ESM at commit and rollback time.

**0**

NO.

**1**

YES.

**2**

SFS calls the ESM only for specific file pool requests, identified by tokens in the third and following lines of the profile. This is the setting in the supplied profile.

### ***obj\_routine***

is the name of the CSL routine the SFS server calls for an object authorization check. The routine specified in the supplied profile is DMSUAUTH.

### ***Bpqrs***

is the token containing flags for an SFS command authorization call.

***p***

indicates whether SFS calls the ESM for command authorization checking.

**0**

NO. This is the setting in the supplied profile.

**1**

YES.

**2**

SFS calls the ESM only for specific file pool requests, identified by tokens in the third and following lines of the profile.

***q***

indicates whether SFS calls the ESM at commit and rollback time.

**0**

NO. This is the setting in the supplied profile.

**1**

YES.

**2**

SFS calls the ESM only for specific file pool requests, identified by tokens in the third and following lines of the profile.

***r***

indicates whether renaming the SFS directory requires SFS command authorization checking.

**0**

NO. This is the setting in the supplied profile.

**1**

YES.

***s***

indicates whether relocating an SFS file or directory requires SFS command authorization checking.

**0**

NO. This is the setting in the supplied profile.

**1**

YES.

**2**

Command authorization is required only when relocating a directory. Command authorization is not required when relocating a file.

**cmd\_routine**

is the name of the CSL routine the SFS server calls for a command authorization check. The routine specified in the supplied profile is DMSAAUTH.

**Ctu**

is the token containing flags for an SFS operator command authorization call:

**t**

indicates whether SFS calls the ESM for operator command authorization checking.

**0**

NO. This is the setting in the supplied profile.

**1**

YES.

**2**

SFS calls the ESM only for specific file pool requests, identified by tokens in the third and following lines of the profile.

**u**

indicates whether SFS will accept an ESM rejection of an operator command (return code 8 from the authorization exit routine).

**0**

NO. This is the setting in the supplied profile.

**1**

YES.

**op\_routine**

is the name of the CSL routine the SFS server calls for an operator command authorization check. The routine specified in the supplied profile is DMSOAUTH.

**Dv**

is the token containing a flag for an ESM program check.

**v**

indicates whether SFS should call a specified ESM routine if a program check occurs in the ESM code.

**0**

NO.

**1**

YES. This is the setting in the supplied profile.

**prog\_routine**

is the name of the CSL routine the SFS server calls to handle a program check in the ESM code. The routine specified in the supplied profile is DMSSECIT.

```
Almno obj_rtn Bpqrs cmd_rtn Ctu op_rtn Dv prog_rtn Ew perm_rtn
```

**Ew**

is the token containing a flag for BFS permission checking. This token is optional. If omitted, the filepool server assumes there is no ESM for BFS objects.

**w**

indicates whether the server should call a specified ESM routine for permission checking for BFS objects.

**0**

NO This is the setting in the supplied profile.

**1**

YES.

**perm\_routine**

is the name of the CSL routine the file pool server calls to perform permission checking for BFS objects. The routine specified in the supplied profile is DMSPERM.

**Record 3: Specific SFS File Pool Requests to Be Reviewed**

The third and following lines of the DMSESM PROFILE may contain tokens identifying specific SFS file pool requests the ESM should review. Note that these lines pertain only to requests for SFS objects from an SFS connection. Each token represents a SFS specific file pool request on a specific type of authorization call.

**Note:** The corresponding flag for the specified type of authorization call must be set to 2 in line two of the profile, or an error occurs.

The format of a file pool request token is:

wxxxxyz
---------

**w**

indicates the type of authorization call.

**A**

SFS object authorization check

**B**

SFS command authorization check

**C**

SFS operator command authorization check

**xxxx**

is the hexadecimal representation of the file pool request code. If the authorization check is an SFS object authorization check or an SFS command authorization check (flag *w* is **A** or **B**), see [Table 12 on page 151](#) for the list of file pool request codes you can use. If the authorization check is an SFS operator authorization check (flag *w* is **C**), see [Table 14 on page 168](#) for the list of the special file pool request codes for SFS operator commands.

*Table 14. Special File Pool Request Codes for SFS Operator Commands*

Command	Code
AUDIT	X'01'
BACKUP	X'02'
CRR	X'0C'
DEFBACKUP	X'0A'
DISABLE	X'03'
ENABLE	X'04'
ERASE	X'0B'
ETRACE	X'05'
FORCE	X'06'
GRANT	X'0D'
ITRACE	X'07'
QUERY	X'08'
REVOKE	X'0E'



Table 14. Special File Pool Request Codes for SFS Operator Commands (continued)

Command	Code
STOP	X'09'

**y**

indicates whether this token applies to all SFS calls of the type indicated by flag *w*.

**0**

NO. (This setting is not valid for an operator command authorization check.)

**1**

YES.

**z**

indicates whether this token applies to SFS calls of the type indicated by flag *w* only at commit and rollback time.

**0**

NO.

**1**

YES. (This setting is not valid for an operator command authorization check.)

The file pool request tokens defined in the supplied DMSESM PROFILE are A002301 (object authorization check for a *delete request* call at commit and rollback) and A002401 (object authorization check for a *delete directory* call at commit and rollback).

## Getting an External Security Manager to Interface with SFS or BFS

This section contains information about the programming interface the file pool server provides to an external security manager (ESM). The information is provided to assist the programmer who is developing or modifying an ESM to use the SFS or BFS interface. The implementation of the interface through the IBM-supplied exit routines for the SFS ESM is described in [Appendix D, “Mapping SFS Authorization Calls to RACROUTE Requests,” on page 689](#).

**Note:** ESMs deal only with file pool server activity, not CRR recovery server activity.

### General Requirements for an ESM

The file pool server should be thought of as having two native security managers. The first handles references to SFS or BFS objects when those references originate from SFS connections. Another way of saying that is when the references are from the CMS API to objects stored in an SFS file space or a BFS file space. In the following, this security manager is simply called the SFS security manager, and the programming interface is described as the interface to an SFS ESM. The second security manager handles references to BFS objects from a BFS connection. This is called the BFS security manager, and the programming interface is described as the interface to a BFS ESM.

The SFS security manager and the BFS security manager each provide the authorization rules appropriate to the semantics of their objects. These rules necessarily differ from each other and these differences will affect the ESM. An ESM designed to support SFS objects will not be sufficient for BFS objects.

### SFS ESM

The file pool server supports many different SFS authorities and does not require an ESM. The file pool server distills its SFS authorization checking into simple verifications of a user's ability to:

- Read, write, or create a file, external object, directory, or alias
- Enter a certain SFS command

When checking a user's read or write authorization to an SFS file or directory, the server SFS checks for authorization only to the specific file or directory. For example, when a user wants to read a file, the server checks only that the user can read the file; the server does not check if the user can read the directory containing the file.

If you elect to use a different authorization scheme, your ESM must provide for SFS file, external object, directory, and alias authorization. To do so, the ESM must maintain its own record of who is authorized to read, write, or create SFS files, external objects, directories, and aliases, and who is authorized to enter SFS commands. Because an ESM's list of authorizations is maintained independent of the SFS authorizations, it does not matter to the SFS server how that list is implemented. You can use whatever technique you like.

Your ESM must also provide some user or administrator interface that allows the authorizations to be granted or revoked. You might, for example, code programs that users can run to perform functions equivalent to the CMS GRANT and REVOKE commands. (The CMS GRANT and REVOKE commands affect only the authorizations SFS itself maintains.) Or, if you want to maintain strict control, you can code programs that can be used by only security administrators. The implementation is your choice, and does not concern the SFS server.

The only part of the ESM that concerns the server is the interface used to route authorization requests to the ESM. Your ESM is responsible for processing these requests, checking the authorizations, and reporting the results to the server. In some instances, your ESM might defer the authorization checking back to SFS (if SFS permits you to do so). Your ESM might do this, for example, if it had no record of the object in question. In that case, it might be better to let the server check the authorization rather than have your ESM report the user was not authorized. Your ESM might also defer authorization checking to the server if the ESM is not implemented to do the kind of authorization checking the server requests. For example, if your ESM does not protect against read requests, it should defer all requests for read authority back to the server.

## BFS ESM

According to the POSIX 1003.1 standard, implementations may provide *additional* or *alternate* access control mechanisms. The characteristics of each are:

- An additional control mechanism "can only further restrict the access permissions defined by the file permission bits". For example, if the permission bits indicate the process does not have read permission to a file, the ESM can not then allow read access to the process.
- An alternate control mechanism:
  1. Must specify permission bits for the same classes as the standard. That is, the ESM must specify an alternate setting of the permission bits in the mode field for the owner, group, and other classes. The alternate permission bits replace the original permission bits.
  2. Must be enabled by explicit user action, on a per-file basis.
  3. Must be disabled for a file after the file permission bits are changed with the chmod command.

Given these definitions, z/VM **does not allow an alternate** control mechanism. Whether it allows an **additional** control mechanism is a matter of interpretation. What it will allow is an ESM that is (strictly speaking) a combination of the two.

The following are the rules for a BFS ESM:

1. The BFS server will call an ESM exit for the purpose of allowing the ESM to determine if the process can perform the indicated function against the indicated object(s). The ESM can decide one of the following.
  - deny the access.

The BFS server will reject the request without any other consideration. The server does not check the permission bits stored with the file to determine if the ESM decision makes sense to the server.
  - allow the access.

The BFS server will allow the request without any other consideration. The server does not check the permission bits stored with the file to determine if the ESM decision makes sense to the server. So, the ESM could contradict the definition of *alternate* control mechanism above. Of course, the ESM writers will be encouraged not to do that, but the point is that it is a decision of the ESM writers and the file pool server does nothing to enforce any restrictions on them.

- defer to the BFS server rules.

This return just means the rules defined in the POSIX standard will be applied by the server. As part of the input to the ESM, the server indicates the results of these rules. That is, the server actually applies the rules before calling the ESM and tells the ESM the results (pass or fail) of those rules.

2. The mode (that is, the permission bits), the owner UID, and owner GID are managed by the file pool server. They are created and changed by rules established by the standard. The ESM can not change them or cause the server to change them. The server does not define an exit or a return from an exit that allows the ESM to change or replace these values.

## File Pool Server Interface to an ESM

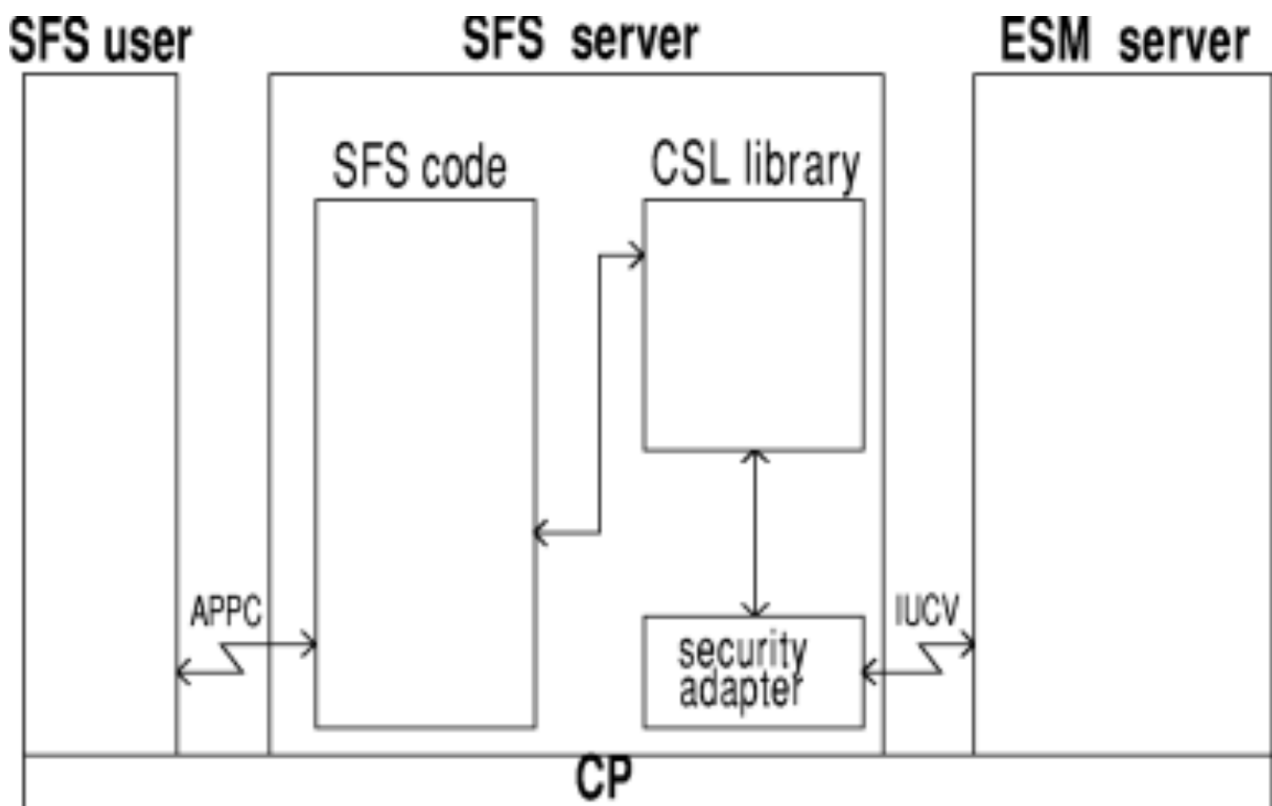


Figure 21. File Pool Server Interface to an ESM

Figure 21 on page 171 shows an overview of the interface the file pool server provides to an ESM. The figure is generic in that the basic control flow shown is the same for an SFS ESM and a BFS ESM. The DMSESM PROFILE indicates the types of file pool requests to be routed ESM and specifies the name of the CSL routine the file pool server calls for each type of request. These CSL routines are installation-wide exits. Table 15 on page 171 lists the exit functions and the IBM-supplied CSL routines.

Table 15. SFS ESM Exits

Exit Function	CSL Routine
ESM initialization and termination	DMSSECIT
SFS object authorization checking	DMSUAUTH

Table 15. SFS ESM Exits (continued)

Exit Function	CSL Routine
SFS command authorization checking	DMSAAUTH
SFS operator command authorization checking	DMSOAUTH
BFS permission checking	DMSPERM
ESM program check	DMSSECIT

## ESM Initialization

During file pool server initialization processing, if the ESECURITY startup parameter is specified in the server's DMSPARMS file, the server calls the ESM initialization CSL routine named in DMSESM PROFILE. Depending on the configuration of your ESM, the ESM initialization routine can interface directly with the ESM or through a portion of the ESM code called the security adapter that is loaded into the file pool server machine. The security adapter provides the communications interface between the file pool server and the separate security manager virtual machine. You could write your own ESM initialization CSL routine that includes the security adapter code, or you could have your initialization routine load a separate security adapter module. For information about writing a replacement routine, see [“Writing Your Own ESM Exit Routines”](#) on page 176.

**Note:** The IBM-supplied ESM initialization routine (DMSSECIT) calls a module named RPIUCMS to enable the ESM. RPIUCMS MODULE is not supplied with z/VM. If your ESM uses this interface, you must provide the RPIUCMS MODULE. If RPIUCMS is missing or cannot be called, server processing ends with an error. For information about coding RPIUCMS, see [z/VM: Security Server RACROUTE Macro Reference](#).

## Authorization Processing

When the ESECURITY startup parameter is in effect, the file pool server looks at DMSESM PROFILE for the name of the appropriate authorization checking CSL routine, builds a CSL parameter list that contains information about the authorization check, and calls the routine. A BFS ESM may be able to perform the authorization check itself without involving an ESM virtual machine. An SFS ESM probably cannot. In that case the CSL routine can route authorization requests to the ESM (through your security adapter) according to the flags set in DMSESM PROFILE.

**Note:** The IBM-supplied SFS authorization checking routines (DMSUAUTH, DMSAAUTH, and DMSOAUTH) call the RACROUTE macro to pass the authorization requests to the ESM. For information about the RACROUTE calls, see Appendix D, “Mapping SFS Authorization Calls to RACROUTE Requests,” on page 689. If you do not want to use the RPIUCMS interface or the RACROUTE macro, you can replace the IBM-supplied exit routines with your own CSL routines that interface directly with your ESM. See [“Writing Your Own ESM Exit Routines”](#) on page 176.

The IBM-supplied BFS authorization checking routine, DMSPERM, does not call the RACROUTE macro. It merely defers all authorization checks back to the file pool server. You must replace DMSPERM if you have a requirement to call an ESM virtual machine.

Your security adapter creates a control block for the authorization data and formulates an IUCV request to pass the authorization data to your security machine. As part of the IUCV request, you identify an IUCV exit routine that gets control when the security machine responds to your authorization check request. Note that the security adapter should not disable IUCV interrupts. Immediately after sending the IUCV request to the security machine, the security adapter should set a return code of 1 (asynchronous wait) on the authorization CSL routine and return to the SFS server.

The security adapter sends the address of the control block to the security manager as part of the authorization data. When the security manager responds to your request, it passes back the address of the control block so the security adapter knows which request is being satisfied. (There can be multiple outstanding authorization requests to the security manager, and the security manager does not necessarily need to respond to them in order.)

The file pool server recognizes that the user for whom it is processing is waiting for an external authorization request to finish. Because it cannot do anything else for this request until the authorization check is completed, the file pool server dispatches another user or waits for an interrupt if there is no other work to be done.

When the security machine has checked the authorization, it uses IUCV to communicate the results to the security adapter in the file pool server machine. The IUCV communication causes an interrupt to occur in the file pool server machine, at which time your IUCV exit routine gets control. The IUCV exit routine should update the original parameter list (that the file pool server passed) with an authorization for the object. (For SFS objects, see authorization routine parameter 16.) The return code indicates the authorization check was successful (the user is authorized), the authorization check was unsuccessful (the user is not authorized), or the ESM defers the authorization check back to the file pool server. In the latter case, your external security manager has decided it does not care about the object in question and wants the file pool server to check its authorizations instead.

### ***Posting the Event Control Blocks (ECBs)***

After setting the return code in the parameter list, your security adapter should post the two event control blocks (ECBs) and return to CMS. To post the ECBs, set the following bit in each fullword ECB:

#### **Byte 0**

##### **Bit 0**

Reserved

##### **Bit 1**

Set to 1 to post

##### **Bit 2-7**

Reserved

#### **Byte 1**

Reserved

#### **Byte 2**

Reserved

#### **Byte 3**

Reserved

Do not alter any of the bits that are reserved.

If file pool server processing was interrupted by the communication from the security machine, CMS now returns control to the file pool server, which finishes what it was doing before it was interrupted. (Perhaps the server was doing work for another user when the response to your authorization request arrived.)

When the server completes its work, it scans the ECB list and notices the ECBs have been posted. The posting of the ECBs tells the server the authorization request is complete, so the server dispatches the agent (user task) that was waiting for the authorization check and calls the authorization routine again.

If the server was not busy when the IUCV interrupt occurred (it was waiting for communications or for some ECB to be posted), CMS notices the ECBs are set and wakes up the file pool server. The server then dispatches the agent (user task) that was waiting for the authorization request and calls the authorization routine again.

In either case, your posting of the ECBs causes the file pool server to complete the work for the user who caused the authorization check.

### ***ECBs and Error Handling***

As an ESM coder, it is your responsibility to ensure the ECBs are posted. The filepool server does not try to find out what happened to an agent if the ECB is never posted. That is, it does not look for missing ECBs. It will do work for others while the user whose ECB is missing waits indefinitely. Assuming your code is debugged and tested, it is still possible for events to occur that would prevent ECBs from being posted. Your ESM should be coded to handle these events and post the ECBs.

The main reason your security adapter would miss posting an ECB (or ECBs) is an error in the security machine. Suppose, for example, the security machine unusually ends. Any outstanding authorization checks would not be answered. Your security adapter must be able to detect there is a problem and post outstanding ECBs. Naturally, the security adapter will have to maintain a list of outstanding requests for it to be able to do this.

The easiest way for the security adapter to know the security machine is experiencing problems is to have the security manager tell it. Security manager termination processing might sever the IUCV connection, for example, and the security adapter could be coded to post outstanding ECBs when IUCV sever occurs.

It is not always possible, however, to have the security manager inform the security adapter it is terminating or experiencing some other problems. For these situations, in which everything seems to be fine but is not, the security adapter might use a "time-out" technique that posts outstanding ECBs after a certain interval has passed. This time-out processing would also have to inform the security machine that it timed-out a user so the security machine can end its work for the user (assuming the security machine is able to receive the communication).

For any of the above errors, which are characterized by the sudden disappearance of the security machine, your security adapter should display a message on the server machine console. Then it should pass back a return code of 8 in the parameter list that the server passed to it originally. (For SFS object authorizations, it should also make each object authorization an N in parameter 16.) Specifying a return code of 8 would cause SFS user requests to be unsuccessful because the users are "not authorized", preventing users from accessing any of their data, thereby keeping the system secure. This action may prompt the users to contact the system administrator to determine the problem.

### Types of Authorization Checks (for SFS ESM only)

SFS file pool server processing calls the external security manager for authorization requests for SFS objects and SFS commands. The authorization checks your ESM must be coded to handle are:

#### Read authority check

Whenever a user wants to read from an object (file, external object, or directory), the SFS server verifies the user is authorized to do so. The SFS server verifies read authority even if the user created the object.

#### Write authority check

Whenever a user wants to write to an object (file, external object, or directory), the SFS server verifies the user is authorized to do so. The SFS server verifies write authority even if the user created the object.

#### Create authority check

If a user attempts to create a new object (file, external object, directory, or alias), the SFS server verifies the user is authorized to do so.

**Note:** This authority check is done only if the appropriate flags are set in the DMSESM PROFILE.

#### What authority query

To process a file pool request like GET DIRECTORY ENTRY, the SFS server enters this query to find out what authorities the user has on the object.

#### Generic profile query

The SFS server issues this query to determine whether a generic profile protects an object.

When provided with the above functions, the file pool server can check the authorization for any request it receives. Many requests require multiple authorization checks. For example, suppose a user enters a CMS COPYFILE command that copies an existing file to a target file that does not yet exist. To verify the user is authorized, the SFS server does two authorization checks:

1. Is the user authorized to read the file to be copied?

For this, the SFS server calls the ESM for a read authority check on the file.

2. Is the user authorized to update the directory to which the file will be copied?

For this, the SFS server calls the ESM for a write authority check on the directory.

The SFS server may also need to do a third authorization check: a create authority check on the new file identifier.

If the ESM returns a "not authorized" return code on any of these requests, the SFS server returns a "not authorized" code to the user machine. CMS in the user's machine then displays the appropriate error message.

All CMS requests to file pool servers are handled in a similar way. Except when you are processing ESM user data, your ESM does not need to analyze the user request or even be aware of what it is. The SFS server handles that aspect of processing for you. All your ESM needs to do is provide "answers" to specific authorization "questions" the SFS server asks.

When you are processing ESM user data, you might find it necessary to know what the request is. For more information, see [“Processing User Data” on page 189](#).

## **Generic Profiles**

Your ESM should include a generic profile query function that allows the SFS server to determine whether:

- All files or external objects in a directory are protected by the ESM.
- All subdirectories within a directory are protected by the ESM.

A generic profile defines a group of objects that are treated as a unit by an ESM when it protects and grants authorization to resources. For example, the ESM can give users authorized for a group of objects authorization for objects that are added to the group. If an ESM defines a group as all the files in the VMSYSU:USERX directory, then users authorized for that group (the directory) are authorized for any file added to VMSYSU:USERX.

SFS file pool servers use the generic profile query when they do operations like the CMS RELOCATE command, which moves an object, such as a directory subtree, to another directory. The server first checks whether there is a generic profile for the source directory; if there is, all subdirectories of that directory are protected. Next, the server checks whether there is a generic profile for the target directory. If there is a generic profile for the source, but not for the target, there may be a security exposure if the subtree is relocated, so the server rejects the request. (The server does a separate check for write authority to the target directory.)

The SFS server uses a generic profile query when it processes a request to erase a directory with the FILES option, in order to ensure a user authorized to erase the directory is also authorized to erase every file in the directory. If a generic profile does not cover all the files in the directory, the server rejects the request, to avoid the security exposure.

The SFS server also uses a generic profile query to make the ACCESS command more efficient. When a user accesses a directory, the SFS server checks whether a generic profile covers the directory. If it does, the SFS server knows all objects within it are protected. Otherwise, the SFS server would have to call the ESM for each file, external object, or directory to determine whether the object was protected.

## **Discrete Profiles**

A discrete profile is an ESM-maintained authorization record for a protected SFS object or command. (SFS has catalogs while ESMs have profiles.) If your ESM is going to update discrete object and command profiles, there are some things to consider. For example, when SFS calls the ESM authorization exit for an ERASE *fn ft*, the file is not erased at that point. The file is not erased until the work is committed. Your ESM may want to use a combination of calls to be sure a change to a profile is valid. (You can have SFS issue a sequence of calls for a specific file pool request code by setting certain flags and providing certain tokens in the DMSESM PROFILE.) The first authorization call from SFS for the ERASE would have a processing code (parameter 5) of B A A. This call provides the name of the file to be erased, and so on. The second call for the ERASE file pool request might have a processing code of B E A, indicating a commit (the ERASE really happened). Then your ESM can update the profile. The transaction ID provided in both calls (parameter 7) links the two requests together. Your ESM would follow the same sort of procedure for a rename or relocate of a file or directory.

## Writing Your Own ESM Exit Routines

### PI

The IBM-supplied ESM exit CSL routines are in the VMLIB callable services library. You can replace the IBM-supplied routines with your own CSL routines. For general information about writing a CSL routine, see *z/VM: CMS Application Development Guide for Assembler*. For information about the CSLENTYR, CSLGETP, and CSLEXIT macros used in writing a CSL routine, see *z/VM: CMS Macros and Functions Reference*.

**Note:** The IBM-supplied routines that interface to an SFS ESM call the RPIUCMS module and the RACROUTE macro. See [Appendix D, “Mapping SFS Authorization Calls to RACROUTE Requests,”](#) on page 689. If your ESM uses this interface, you may not need to replace the IBM-supplied routines. The IBM-supplied CSL routine for the BFS ESM, DMSPERM, merely defers all requests back to the file pool server.

When writing your own CSL routines, you must take into account future possibilities, such as:

- New optional parameters
- New authorization types
- More flags for these authorization types
- New object types
- New access requested types

Also see [Appendix C, “File Pool Server Exit Considerations,”](#) on page 687.

If you supply routines with new names, the DMSESM PROFILE must be updated with the new names.

### Initialization/Termination and ESM Program Check Routine

IBM supplies a single CSL routine (DMSSECIT) to handle initialization and termination calls to the ESM. DMSSECIT also handles ESM program checks. You can replace this routine with your own CSL routine. If you prefer, you can write a separate CSL routine for the program check function.

The IBM-supplied CSL template file for both exits is DMSJBITP TEMPLATE, shown in [Figure 22](#) on page 176. This file identifies the routine's input and output parameters. The general format of a CSL template file is described in *z/VM: CMS Application Development Guide for Assembler*.

**Note:** The initialization, termination, and program check conditions in this routine are not asynchronous.

13	13		13 parameters and all are required
SBIN	4	OUTPUT	Parm 1 - Return code
FCHR	0	OUTPUT	Parm 2 - Reason code string
FCHR	8	INPUT	Parm 3 - Server userid
FCHR	8	INPUT	Parm 4 - Filepool id
SBIN	4	INPUT	Parm 5 - Maxconn value for server
SBIN	4	OUTPUT	Parm 6 - Number of connects needed by ESM
SBIN	4	INPUT	Parm 7 - Max number of agents in server
FCHR	0	INPUT	Parm 8 - Command keyword
FCHR	8	OUTPUT	Parm 9 - ESM identity
FCHR	1	INOUT	Parm 10 - Protect_All indicator
FCHR	8	INPUT	Parm 11 - PSW content for program check
SBIN	4	INOUT	Parm 12 - Local storage area pointer
SBIN	4	INOUT	Parm 13 - Global storage area pointer

Figure 22. DMSJBITP TEMPLATE file

DMSJBITP TEMPLATE contains templates for 13 required parameters:

1. The **return code** from the routine. A return code of 0 indicates success; a nonzero return code indicates an error.



- The **reason code string** from the routine. This string is six fullwords long, in the following format:

```
count_reasoncode1_reasoncode2_reasoncode3_reasoncode4_reasoncode5
```

The first fullword is a count of the number of reason codes set by the routine. The ESM can use the reason code structure for detailed debugging codes.

- The **user ID of the file pool server** calling the routine.
- The **file pool ID**.
- The **MAXCONN value for the file pool server**, which is the maximum number of IUCV and APPC/VM connections specified for the server virtual machine in the CP directory minus any connections already used for minidisks.
- The **number of the file pool server's connections the ESM needs**. This depends on the communications protocol used between the file pool server and the ESM. For example, IUCV can use a single connection; APPC/VM can use any number. If no connections are needed, specify 0.
- The **maximum number of agents** (tasks) in the file pool server that can be active.
- A **command keyword** that indicates the type of call:

**INIT**

Initialization

**TERM**

Termination

- The **name of the ESM**.
- A **protect-all indicator** that indicates whether the ESM is protecting all resources:

**0**

No

**1**

Yes

This parameter applies only to the SFS ESM. It has ignored for the BFS ESM. SFS sets the indicator to 0 on the call. If the ESM resets the indicator to 1, SFS uses the information to increase performance on ACCESS, RENAME, or RELOCATE of a directory. All subdirectories and files in the directory are marked with a P to indicate they are protected by the ESM, which saves having to check each one individually.

**Note:** There is a flag in the DMSESM PROFILE that indicates whether SFS will allow the SFS ESM to defer authorization requests back to SFS.

- (Program check only) the **PSW content** from the CMS abend area, stored there as the result of an ESM program check.
- (Program check only) a **pointer to the local storage area** of the agent running when the program check occurred. See the description of the local storage area parameter on the authorization exits.
- A **pointer to a global storage area** in which the ESM can save information it needs for all requests, such as the address of its control blocks containing data related to the server.

## Authorization Checking Routines (SFS ESM)

You can replace the IBM-supplied SFS object authorization checking CSL routine (DMSUAUTH), the SFS command authorization checking CSL routine (DMSAAUTH), and the SFS operator command authorization checking CSL routine (DMSOAUTH) with your own CSL routines. In each routine, the number and order of the parameters is the same (although some parameters might be unused). Therefore, if you prefer, you can write one CSL routine to handle all three exits.

The IBM-supplied CSL template file for these routines is DMSJBATP TEMPLATE, shown in [Figure 23 on page 178](#). This file identifies the routines' input and output parameters. The general format of a CSL template file is described in [z/VM: CMS Application Development Guide for Assembler](#).

23	23		23 Parameters defined, 23 required
SBIN	4	OUTPUT	Parm 1 - Return Code from auth check routine
FCHR	0	OUTPUT	Parm 2 - Reason Code string
FCHR	8	INPUT	Parm 3 - Userid requesting authorization check
SBIN	4	INPUT	Parm 4 - File Pool Request Code
FCHR	0	INPUT	Parm 5 - Processing Code
SBIN	4	INPUT	Parm 6 - SFS request sequence number
SBIN	4	INPUT	Parm 7 - Transaction ID
FCHR	8	INPUT	Parm 8 - File Pool Id of the server
FCHR	4	OUTPUT	Parm 9 - Event Control Block 1
FCHR	4	OUTPUT	Parm 10 - Event Control Block 2
FCHR	0	INPUT	Parm 11 - Object List or Command String
FCHR	0	INOUT	Parm 12 - User Data Flags
FCHR	0	INPUT	Parm 13 - User Data entered by user
SBIN	4	INOUT	Parm 14 - Local Storage Pointer
SBIN	4	INPUT	Parm 15 - Global Storage Pointer
FCHR	0	OUTPUT	Parm 16 - Object authorizations
FCHR	1	INPUT	Parm 17 - Check All Objects indicator
FCHR	0	INPUT	Parm 18 - User ID list for command
FCHR	8	INPUT	Parm 19 - Source server VMID
FCHR	0	INPUT	Parm 20 - ESM Security token
SBIN	4	INOUT	Parm 21 - Buffer addr for ESM info
FCHR	1	INOUT	Parm 22 - New sectoken info indicator
FCHR	0	INPUT	Parm 23 - User connection information

Figure 23. DMSJBATP TEMPLATE File

DMSJBATP TEMPLATE contains templates for 23 required parameters.

**Note:** Some required parameters are used only for specific types of authorization checks.

In an object check or command check, only those parameters marked with a star ★ are valid at commit and rollback time.

1. The **return code** from the routine. Your routine must return one of the following values:

**0**

Normal termination.

**1**

Exit routine not finished; wait and call again when both event control blocks are posted.

The exit routine uses this return code to handle asynchronous calls to the ESM server. This allows the SFS server to continue to do work for other users.

**4**

(Command check and operator command check only) Defer to SFS, but only if all protection is disabled.

**8**

Request rejected.

**Note:** There is a flag in the DMSESM PROFILE that indicates whether SFS will accept the ESM's rejection of an operator command.

**12**

Unusual error. The SFS server abends, and CMS message 2030E is entered.

2. The **reason code string** from the routine. This string is six fullwords long, in the following format:

```
count_reasoncode1_reasoncode2_reasoncode3_reasoncode4_reasoncode5
```

The first fullword is a count of the number of reason codes set by the routine. If the routine returns with a return code of 12, the reason codes are displayed in CMS message 2030E. The ESM can use the reason code structure for detailed debugging codes.

Reason code 3 is used for the return code from processing a DMSUDATA request. See [“Processing the DMSUDATA CSL Routine”](#) on page 191.

3. The **user ID** requesting the authorization check. If the call is for an operator command check, this is the ID of the SFS server machine.
4. The **file pool request code** for the operation. The codes for SFS object and SFS command operations (except operator commands) are listed in [Table 12 on page 151](#). The special codes for SFS operator commands are listed in [Table 14 on page 168](#).

**Note:** This code is unreliable during commit and rollback processing.

5. ★ A **processing code**, in the format  $w x y$ , indicating the purpose of the call.

**w**

indicates the type of call:

**A**

SFS operator command authorization check

**B**

SFS object authorization check

**C**

SFS command authorization check

**x**

indicates when the call is being made.

**A**

Before the server processes the request.

**B**

(Object check or command check only) after the LUW (transaction ID) has been committed; the SFS request sequence number is not meaningful.

**C**

(Object check or command check only) after the LUW (transaction ID) has been rolled back; the SFS request sequence number is not meaningful.

**D**

(Object check or command check only) after the specific request identified by the combination of the file pool request code (parameter 4) and the SFS request sequence number (parameter 6) has been rolled back.

**E**

(Object check only) after the specific request identified by the combination of the file pool request code (parameter 4) and the SFS request sequence number (parameter 6) has been committed.

**F**

(Object check only) after the specific request identified by the combination of the file pool request code (parameter 4) and the SFS request sequence number (parameter 6) has been committed, but the file is still open.

**y**

(object or command check only) indicates whether the call requires special processing beyond the usual processing for this file pool request code.

**A**

No.

**B**

(Object check only) do you know query: the SFS server wants to know if the object is protected by (is known to) the ESM. The SFS server asks for R (read) access to the object. The ESM, if the answer is "No, I do not know about this object", replies with an object authorization (parameter 16) of D, to defer the access decision to SFS. Any positive object authorization (R, W, C) is considered by SFS to be an answer of "Yes, I do know about this object".

**C**

(Object check only) generic profile query: the SFS server wants to know if a generic profile is protecting a directory's subdirectories (object\_type = D) or files/external objects (object\_type = F). The SFS server asks for R (read) access to the object. If the ESM does not have a generic profile covering these objects it answers with an object authorization (parameter 16) of D. Any positive object authorization (R, W, C) means the ESM has a generic profile covering these objects.

**D**

(Object check only) what authority query: the SFS server wants to know what kinds of authority the user has on the object. The SFS server asks for RWC (read, write, and create) access to the object. The ESM replies with whatever access the user is allowed. Each authorization is unique; for example, W (write) access does not imply read access.

**E**

(Object check only) miscellaneous authority check: the SFS server wants to know if the user has read or write authority on the file or directory.

6. ★ (Object check or command check only) the **SFS request sequence number**. This number identifies the specific request within the transaction ID, because the transaction ID might contain several requests. This is simply a counter that starts from 1.
7. ★ (Object check or command check only) the **transaction ID**, a number that identifies all requests within a particular unit of work until a commit or rollback occurs.
8. ★ The **file pool ID**.
9. ★ **Event control block 1**, a fullword containing a bit posted (set) by the ESM (together with event control block 2) to "wake up" the SFS server to call the CSL routine again.
10. ★ **Event control block 2**, a fullword containing a bit posted (set) by the ESM (together with event control block 1) to "wake up" the SFS server to call the CSL routine again.
11. The **object list or command string** to be checked. For a non-operator command check, SFS rebuilds the command or CSL routine to a standard form before passing it in the authorization request. See Table 37 on page 692. Operator commands do not need to be rebuilt. For an object check, SFS passes a list of object descriptions, each description having the format:

```
object_type access_requested ownerid object_name
```

**object\_type**

indicates the type of object:

**F**

Existing file

**FN**

New file

**A**

Existing alias

**AN**

New alias

**D**

Existing directory

**DN**

New directory

**E**

Existing external object

**EN**

New external object

**access\_requested**

indicates the type of access requested:

**C**  
Create

**R**  
Read

**W**  
Write

**RW**  
Read and write

**RWC**  
Read, write, and create

All the authorization types except what authority have just one *access\_requested* type for each object. The what-authority authorization check asks for every possible *access\_requested* type the server knows about.

**ownerid**  
is the owner of the object.

**object\_name**  
is the name of the object to be checked. This is a 152-character field in the following format:

```
TOP DIRECTORY (CHAR 8)
D1          (CHAR 16)
:
D8          (CHAR 16)
FILENAME      (CHAR 8)
FILETYPE      (CHAR 8)
```

12. (Object check or command check only) a pair of **user data flags**. SFS sets the first flag to 1 if there is user data specified, or 0 if there is no user data. SFS always sets the second flag to 0.

If SFS indicates there is user data on the call, the ESM must reset the second flag to 1 on the return to SFS to indicate the ESM has processed the user data. If the ESM does not reset the second flag, the user's request is unsuccessful (the same as if an access return code of N was returned in parameter 16).

13. (Object check or command check only) a string of **user data** that SFS passes to the ESM.

14. ★ A **pointer to a local storage area**, in which the ESM can save information about this authorization request, such as the address of its control blocks. The ESM may use this storage area to hold the address of the RACROUTE PLIST. On the initial call, SFS passes a string of zeros.

This area can be very helpful in cases where several CSL calls are necessary for a single authorization request. For example, to check a COPYFILE command, separate calls are made to the ESM to check the user's read access to the source, write access to the target, and alter access to the new file.

15. ★ A **pointer to the global storage area** established during initialization.

16. (Object check only) a string of **object authorizations** from the ESM, indicating the type of access to be allowed for each object in the object list:

**C**  
Create

**R**  
Read

**W**  
Write

**RW**  
Read and write

**RWC**  
Read, write, and create

**N**

No access allowed

**D**

Access decision deferred to SFS

For example, the string RW R N indicates the user can have read/write access to the first object, read access to the second object, and no access to the third object.

17. (Object check only) a **check all objects indicator** that indicates whether the ESM must check all objects and obtain an authorization for each one before returning control to SFS.

**Y**

Yes

**N**

No

In the case of some errors, checks that follow can be skipped.

18. (Optional, command check only) a blank-delimited **list of the user IDs** in the group, if the command being checked affects a group of users.
19. (Optional, object or command check only) the **ID of a server** initiating the authorization check on behalf of the user.

See the explanation of the *userid* and *sectoken* parameters of the DMSGETWU CSL routine in [z/VM: CMS Callable Services Reference](#).

20. (Optional, object or command check only) a 0–64 byte **security token** from the user machine. The ESM uses this parameter in conjunction with the following two parameters.
21. (Optional, object or command check only) a pointer to a **storage buffer** the ESM may use for new security information about the user. This information is then passed back to the ESM on subsequent requests.

SFS passes a zero on the initial call. The authorization routine allocates the buffer and passes the address to the ESM.

22. (Optional, object or command check only) a **new security token information indicator** that indicates whether the ESM is to generate new security information about the user.

**Y**

Yes

**N**

No

If used, SFS sets the indicator to Y on the call; the ESM resets the indicator to N on the return after generating the new information successfully.

23. (Object or command check only) the **user connection information** which indicates if there is more information following the userdata content. The information consists of:

```
CONNECTION TYPE      (FIXED 32)
PATH ID NUMBER      (FIXED 32)
UNIQUE_TOKEN        (FIXED 32)
VALID_FLAG          (CHAR 1)
RESERVED BY IBM     (CHAR 3)
CPEVPLKL INFORMATION (CHAR 16)
CPEVRLUL INFORMATION (FIXED 32)
CPEVRLUN INFORMATION (CHAR *)
```

For details on this connection information, see [Table 38 on page 695](#).

### Authorization Checking Routines (BFS ESM)

The IBM-supplied BFS object authorization checking CSL routine is DMSPERM. It defers all requests back to the file pool server. The IBM-supplied CSL template file for this routine is DMSJBPTP TEMPLATE, shown in [Figure 24 on page 183](#). This file identifies the routines' input and output parameters. The general format of a CSL template file is described in [z/VM: CMS Application Development Guide for Assembler](#).

```

14 14          14 parameters defined, 14 required
SBIN  4  OUTPUT Parm  1 - Return Code from the BFS ESM exit
FCHR  0  OUTPUT Parm  2 - Reason Code string
SBIN  1  INPUT  Parm  3 - Request type
FCHR  0  INPUT  Parm  4 - Processing Code
SBIN  4  INPUT  Parm  5 - BFS request sequence number
FCHR  8  INPUT  Parm  6 - File Pool Id of the server
FCHR  4  OUTPUT Parm  7 - Event Control Block 1
FCHR  4  OUTPUT Parm  8 - Event Control Block 2
FCHR  4  OUTPUT Parm  9 - Request dependent output
SBIN  4  INPUT  Parm 10 - Pointer to the CRED structure
SBIN  4  INPUT  Parm 11 - Pointer to the FSP structure
SBIN  4  INPUT  Parm 12 - Pointer to the CLIENT structure
SBIN  4  INPUT  Parm 13 - ESM Local Storage Pointer
SBIN  4  INPUT  Parm 14 - ESM Global Storage Pointer

```

Figure 24. DMSJBPTP TEMPLATE File

1. The **return code** from the routine. The routine must return one of the following values:

**0**

The ESM allows the access. The file pool server should continue with the operation.

**1**

Exit routine not finished; wait and call again when both event control blocks are posted.

The exit routine uses this return code to handle asynchronous calls to the ESM server. This allows the file pool server to continue to do work for other users.

**4**

Defer to the permission checks done by the file pool server. These checks are described in [Table 13 on page 156](#).

**8**

Permission is denied.

**12**

Unusual error. The file pool server abends, and CMS message 2030E is displayed.

2. The **reason code string** from the routine. This string is six fullwords long, in the following format:

```
count_reasoncode1_reasoncode2_reasoncode3_reasoncode4_reasoncode5
```

The first fullword is a count of the number of reason codes set by the routine. If the routine returns with a return code of 12, the reason codes are displayed in CMS message 2030E. The ESM can use the reason code structure for detailed debugging codes.

3. The **request type** for the operation. This value in conjunction with the **processing code** (parm 4) define what operation the BFS server is performing and thus enable the ESM to determine what permission rules it should apply. The values for request type are shown in [Table 16 on page 183](#).

Table 16. BRM requests.

Request name	Type	Description
Access	x'01'	Access
Catalu	x'02'	Catalog utility
Chaudit	x'03'	Change audit options
Chmod	x'04'	Change permission
Chown	x'05'	Change owner
Link	x'06'	Create a hard link
Lookup	x'07'	Lookup

Table 16. BRM requests. (continued)

Request name	Type	Description
MkCat	x'08'	Make Catalog
Mkobj	x'09'	Make object
Open	x'0A'	Open a regular file
Opendir	x'0B'	Open directory
Readlink	x'0C'	Read soft link
Rename	x'0D'	Rename
Rmdir	x'0E'	Remove directory
SoftLink	x'0F'	Create a soft link
Unlink	x'10'	Delete a link
Utime	x'11'	Update time
Paccess	x'12'	Pipe access
Putime	x'13'	Pipe utime

4. **Processing code.** This is a variable length character string in the format *w x*. That is, the second character is actually a blank and is not used. The first character in the string indicates what processing the Byte File Server is performing when it makes the call to the exit.

**w**

indicates the type of call.

**A**

Performing a permission check for a given file pool request type. The request type is identified in parameter 3, **request type**.

**B**

Performing pathname resolution. The file pool server is traversing the directory in the path prefix. This is done for every request that allows the specification of a path name. The file pool server considers that search permission is required, but the ESM may have any rules it likes. The request that is causing the pathname resolution is identified in parameter 3, **request type**.

**x**

indicates when the call is being made.

**A**

After the server has applied its permission checking rules and before the request is done. This is the only value used by the file pool server. Other values are reserved for a future release.

The results of the server permission check are passed to the ESM exit in QT7FSPresult (see Table 18 on page 186).

This parameter is described in more detail in the example in the description of parm 5.

5. **BFS request sequence number**, a number that identifies all ESM exit calls within a given file pool request. One file pool request may result in multiple calls to the ESM exit. The processing flow is best described with an example. Suppose the client request is to open file A/B/C/y. The following occurs:
- A function of z/VM known as the Logical File System (LFS) executing in the client virtual machine would first call the server to lookup A/B. The ESM exit is called with **Processing code** of B A and the **request type** of x'07'. The request sequence number would be, for example, 136.
  - Next, the client machine calls the server to lookup B/C. The ESM is called again with **Processing code** of B A and the **request type** of x'07'. The request sequence number would be, for example, 291.



c. Next, the client machine calls the server to open C/y. The ESM exit would be called during the pathname resolution step of the open. **Processing code** would be B A and the **request type** would be x'0A' (open). Assuming the ESM allows the request, the ESM exit would be called a second time, identifying file y. **Processing code** would be A A and the **request type** would be x'0A' (open). The **BFS request sequence number** would be the same for both of these calls, allowing the ESM to relate them if necessary.

If the client was to later open the same file again, since A,B and C have already been looked up and therefore cached in the client machine, the client logic would call the server only once - to open file y. There would be no call to the ESM to perform search permission checks, just a call to perform open permission checks.

6. The **file pool ID**. This is simply the file pool id of the file pool server.
7. **Event control block 1**, a fullword containing a bit posted (set) by the ESM (together with event control block 2) to "wake up" the file pool server to call the CSL routine again.
8. **Event control block 2**, a fullword containing a bit posted (set) by the ESM (together with event control block 1) to "wake up" the file pool server to call the CSL routine again.
9. **Request dependent output**. This is output from the ESM exit for cases where a specific request dependent action is to be taken by the file pool server. Currently there is only one such condition.
  - a. From the permission check (processing code A A) for chmod. The POSIX standard states that under some conditions, chmod must result in the set\_gid\_onx bit being cleared. To indicate this condition, the ESM should set the first byte of the **Request dependent output** to a value of x'01'. The remaining three bytes are ignored.
10. **Pointer to the CRED structure**. The CRED is a structure used to pass path name information to the ESM exit. The structure is mapped by DMS7QT. It basically contains the pathnames as they are known in the client machine. These pathnames are passed to the ESM exit for the purpose of auditing.

Table 17. CRED - DMSPERM parm 10, length is 64 bytes.

Name	DISP	LEN	Description
QT7Credid	0	4	Eye catcher, contains 'CRED'.
	4	1	Reserved, contains 0.
QT7Credlen	5	3	Length of this CRED (X'40').
QT7Credver	8	1	Version number (see constants).
	9	3	Reserved, contains 0.
QT7Crednameflg	12	1	Indicates which (1 or 2) name is being checked.
	13	3	Reserved, contains 0.
QT7CredPN1	16	12	Path name 1. It is the name specified by the client. For link and rename, it is the existing file.
	16	2	Reserved, contains 0.
QT7CredPN1len	18	2	Length of path name 1.
	20	4	Reserved, contains 0.
QT7CredPN1addr	24	4	Address of path name 1.
QT7CredFN1	28	12	File name 1. This is the part of path name 1 being checked.
	28	2	Reserved, contains 0.
QT7CredFN1len	30	2	Length of file name 1.

Table 17. CRED - DMSPERM parm 10, length is 64 bytes. (continued)

Name	DISP	LEN	Description
	32	4	Reserved, contains 0.
QT7CredFN1addr	36	4	Address of file name 1.
QT7CredPN2	40	12	Path name 2. It is the name specified by the client. For link and rename, it is the new file. When creating a soft link, this is the contents of the link.
	40	2	Reserved, contains 0.
QT7CredPN2len	42	2	Length of path name 2.
	44	4	Reserved, contains 0.
QT7CredPN2addr	48	4	Address of path name 2.
QT7CredFN2	52	12	File name 2. This is the part of path name 2 that is being checked. When path name 2 is the contents of a soft link, this field is not used.
	52	2	Reserved, contains 0.
QT7CredFN2len	54	2	Length of file name 2.
*	56	4	Reserved, contains 0.
QT7CredFN2addr	60	4	Address of file name 2.

Constants defined for fields in the CRED.

```

Content of QT7Credid
  QT7Credeye      Char(4)  Constant('CRED')

Contents of QT7CredVer
  QT7Credver1    Fixed(8)  Constant('01'x)   Version 1
  QT7CredLver    Fixed(8)  Constant('01'x)   Latest version

Contents of QT7CredCrednameflg
  QT7Credname1   Fixed(8)  Constant('01'x)
  QT7Credname2   Fixed(8)  Constant('02'x)

```

11. **Pointer to the FSP structure.** The FSP is a structure that describes the object(s) involved in the request. It is mapped by DMS7QT.

Table 18. FSP- DMSPERM parm 11, length is 208 bytes.

Name	DISP	LEN	Description
QT7FSPid	0	4	Eye catcher, contains 'FSP'.
QT7FSPver	4	1	Version number (see constants).
QT7FSPresult	5	1	Results of server test. 0 if allowed.
QT7FSPlen	6	2	Length of the FSP.
QT7FSParray	8	-	4 entry array, each entry is 40 bytes.
QT7FSPflags	0	4	Flags, defining the array.
QT7FSPused	-	b	On if this entry contains data.
QT7FSProot	-	b	On if this entry is for root dir.

Table 18. FSP- DMSPERM parm 11, length is 208 bytes. (continued)

Name	DISP	LEN	Description
QT7FSPNew	-	b	This bit is on when the array entry pertains to a new object or name. That is possible for link, softlink, open, rename, and mkobj. For these entries the INO, SCID, and audit flags are 0. Except for rename, the remaining fields are filled in. For rename, only the filespace name is filled in.
QT7FSPOwnuid	4	4	Owning UID of the file.
QT7FSPOwngid	8	4	Owning GID of the file.
QT7FSPmode	12	4	Mode of file.
QT7FSPUaudfl	16	4	User audit flags.
QT7FSPAaudfl	20	4	Auditor audit flags.
QT7FSPFQID	24	16	File identification.
QT7FSPfspath	24	8	Filespace name.
QT7FSPino	32	4	INO - file serial number.
QT7FSPscido	36	4	SCID of object.
QT7FSPAuditopt	174	1	For chaudit, the audit option. x'00' indicates the user audit flags are being changed. X'01' indicates the auditor audit flags are being changed.
*	175	1	Reserved, contains 0.
QT7FSPAuditflg	176	4	For chaudit, the new audit flags.
QT7FSPnewmode	180	4	The new mode value for chmod, mkobj and open of a new file.
QT7FSPnewuid	184	4	For chown, the new uid value.
QT7FSPnewgid	188	4	For chown, the new gid value.
QT7FSPutimes	192	8	For utime, the new times. The first 4 bytes is the access time. The second 4 bytes is the modification time. If either value is X'ffffff' (-1), it means the times will be set to the current time.
	200	8	Reserved, contains 0.

Content of QT7FSPid  
 QT7FSPeye Char(4) Constant('FSP ')

Contents of QT7FSPVer  
 QT7FSPver1 Fixed(8) Constant('01'x) Version 1  
 QT7FSPLver Fixed(8) Constant('01'x) Latest version

Index into QT7FSParray  
 QT7FSPtarg1 Fixed(8) Constant(1) target file, path 1  
 QT7FSPptarg1 Fixed(8) Constant(2) parent of entry one  
 QT7FSPtarg2 Fixed(8) Constant(3) target file, path 2  
 QT7FSPptarg2 Fixed(8) Constant(4) parent of entry three

Values for open type(when request is open)

QT70penR	Char(1)	Constant('R')	Open read
QT70penW	Char(1)	Constant('W')	Open write
QT70penX	Char(1)	Constant('X')	Open execute

12. **Pointer to the CLI structure.** The CLI (for CLIENT) structure contains information about the user making the request.

Table 19. CLIENT Structure - DMSPERM parm 12, length is 88 bytes.

Name	DISP	LEN	Description
QT7CLId	0	4	Eyecatcher, contains 'CLI'.
QT7CLIVER	4	1	Version number (see constants).
*	5	1	Reserved, contains 0.
QT7CLILEN	6	2	Length of this CLI.
QT7CLIFLAGS	8	4	Miscellaneous flags.
QT7CLIAADMIN	-	b	User is a file pool administrator.
QT7CLIUUSERID	12	8	VM userid of requestor.
QT7CLIPROCID	20	8	Process ID.
QT7CLIRUID	28	4	Process real uid.
QT7CLIRGID	32	4	Process real gid.
QT7CLIEUID	36	4	Process effective UID.
QT7CLIEGID	40	4	Process effective GID.
QT7CLISUPCNT	44	4	Number of supplementary GIDS.
QT7CLISUPPTR	48	4	Pointer to supplementary GIDS.
*	52	36	Reserved, contains 0.

13. **Pointer to a local storage area.** This is a pointer to a 32 byte area that can be used by the ESM. For the first ESM exit call within a BFS sequence number, the area contains all zeros. On subsequent calls, within that BFS sequence number, it will contain whatever the ESM placed there. It's considered local since its use is intended for just the exit calls for a single file pool server request.
14. **Pointer to the global storage area.** This is a pointer to a 64 byte area the BFS server establishes during file pool initialization. The ESM can use this for any purpose it likes. Once it is established, it remains until the server is stopped.

## Making the Replacement Exit Routines Available

The file pool server must call the replacement exit routines instead of the IBM-supplied modules in VMLIB. One way to do this is to put the replacement routines in a separate callable services library:

1. Assemble the routines to build TEXT files.
2. If any routine has a different name than the IBM-supplied version, update the DMSESM PROFILE with the new name.
3. Create a CSL control file for the library. Although any file ID for the control file is permitted, the convention is to use the name of the library and the file type CSLCNTRL. For example, if the library is called ESMLIB, name the control file ESMLIB CSLCNTRL.

In the control file, create a ROUTINE record for each exit routine that identifies the CSL routine name (the name to be called by the file pool server), the TEXT file name, and the CSL template file name. For example, the control file might look like this:

```
ROUTINE DMSSECIT DMSSECIT DMSJBITP
ROUTINE DMSUAUTH DMSUAUTH DMSJBATP
```

```
ROUTINE DMSAAUTH DMSAAUTH DMSJBATP
ROUTINE DMSOAUTH DMSOAUTH DMSJBATP
ROUTINE DMSPERM DMSPERM DMSJBPTP
```

The complete format of a ROUTINE record is described in [z/VM: CMS Application Development Guide for Assembler](#).

4. Use the CSLGEN command to build the callable service library. The library can be stored on a minidisk or in a saved segment. For example, the following command builds ESMLIB on a minidisk accessed as file mode A:

```
cslgen dasd esmlib from esmlib
```

Information about the CSLGEN command is in [z/VM: CMS Commands and Utilities Reference](#).

5. Update the file pool server's PROFILE EXEC.
  - a. Add a command to access the minidisk on which the library resides. (This is not necessary if the library is in a saved segment.)
  - b. Add an RTNLOAD command to load each new routine. For example, the following command loads the DMSSECIT routine from the ESMLIB library;

```
rtnload dmssecit (from esmlib system
```

Adding a GLOBAL command is not necessary because the library is specified in the RTNLOAD command. An RTNDROP command is not required either; the new DMSECIT overrides the IBM-supplied version when the RTNLOAD command is processed. Information about the RTNLOAD command is in [z/VM: CMS Commands and Utilities Reference](#).

- c. If applicable, release the minidisk on which the library resides.

## ESM User Data (SFS ESM)

The CMS CSL application programming interface provides ESM user data support that lets you extend your ESM function beyond rudimentary authorization checking. Note that ESM user data is not provided in the application programming interface to BFS files. Using ESM user data, you can code ESMs that support functions such as password protection of files, external objects, and directories. These functions can then be exploited by application developers who use your ESM.

The ESM user data support has two parts: a CSL interface for application programmers and a server interface to your ESM. The CSL interface consists of a general-purpose CSL routine (DMSUDATA - Send User Data), an optional *userdata* parameter on other key CSL routines, and an information name available through the extract/replace CSL routine (DMSERP). The programmer can pass up to 80 bytes of user data to the ESM. The server interface is simply an authorization call to the ESM, passing the user data in one of the parameters.

**Note:** For SFS command and SFS object authorization calls, the IBM-supplied authorization CSL routine also builds a user data structure and passes the address of this control block to the ESM on the RACROUTE request. For more information, see [“User Data Structure” on page 695](#).

If your ESM is not going to process user data, it can ignore any user data the server passes. Your ESM documentation should, in this case, tell the programmers to omit the *userdata* parameter from all CSL routines. Your documentation should also warn them not to use the DMSUDATA routine.

## Processing User Data

Table 20 on page 190 shows the CSL routines that have a *userdata* parameter and indicates the type of authorization checking the SFS server does for each routine. All of these CSL routines are documented in [z/VM: CMS Callable Services Reference](#). Your ESM might require users to specify *userdata* on only some of these routines. Some server functions require multiple authorization checks. If there are multiple checks for a given server function, the same ESM user data is passed on each call. Your ESM must decide when the user data is relevant and when it is not.

Table 20. CSL Routines That Allow ESM User Data

CSL Routine	Authorization Checks
DMSCATTR - Change Attributes	See <a href="#">“Miscellaneous Authority Check”</a> on page 703. This function is not supported for external objects.
DMSCRALI - Create Alias	See <a href="#">“CREATE ALIAS”</a> on page 699. This function is not supported for external objects.
DMSCRDIR - Create Directory	See <a href="#">“CREATE DIRECTORY”</a> on page 700.
DMSCRFIL - Create File	See <a href="#">“DMSOPEN (New or Existing File)”</a> on page 702.
DMSCRLOC - Create Lock	See <a href="#">“CREATE LOCK”</a> on page 700. This function is not supported for external objects.
DMSCROB - Create Object	See <a href="#">“Miscellaneous Authority Check”</a> on page 703. For external objects, authorization is write access to target directory.
DMSDELOC - Delete Lock	See <a href="#">“DELETE LOCK”</a> on page 403. This function is not supported for external objects.
DMSERASE - Erase	See <a href="#">“DISCARD/ERASE”</a> on page 701. For external objects, authorization is write access to parent directory.
DMSEXIDI - Exist (Directory)	See <a href="#">“‘What Authority?’ Query”</a> on page 705.
DMSEXIFI - Exist (File)	See <a href="#">“‘What Authority?’ Query”</a> on page 705. For external objects, authorization is read access to parent directory.
DMSEXIST - Exist	See <a href="#">“‘What Authority?’ Query”</a> on page 705. For external objects, authorization is read access to parent directory.
DMSFILEC - Filecopy	See <a href="#">“COPYFILE NEWFILE/REPLACE”</a> on page 699.
DMSGGRANT - Grant Authority	See <a href="#">“‘Do You Know?’ Query”</a> on page 702. Also see <a href="#">“Special Considerations for GRANT and REVOKE”</a> on page 191. This function is not supported for external objects.
DMSOPBLK - Open Blocks	See <a href="#">“DMSOPEN (New or Existing File)”</a> on page 702.
DMSOPDBK - Open Data Blocks	See <a href="#">“DMSOPEN (New or Existing File)”</a> on page 702.
DMSOPDIR - Open Directory	See <a href="#">“Miscellaneous Authority Check”</a> on page 703.
DMSOPEN - Open	See <a href="#">“DMSOPEN (New or Existing File)”</a> on page 702.
DMSQOBJ - Query External Object	See <a href="#">“Miscellaneous Authority Check”</a> on page 703. Authorization is read access to parent directory.
DMSRELOC - Relocate	See <a href="#">“RELOCATE (File, Alias, External Object, or Directory)”</a> on page 704. For external objects, authorization is same as relocate of file.
DMSRENAM - Rename	See <a href="#">“RENAME (File, Alias, External Object, or Directory)”</a> on page 705. For external objects, authorization is same as rename of file.
DMSREVOK - Revoke Authority	See <a href="#">“‘Do You Know?’ Query”</a> on page 702. Also see <a href="#">“Special Considerations for GRANT and REVOKE”</a> on page 191. This function is not supported for external objects.
DMSUDATA - Send User Data	Special case read check (file pool request code X'2B'). See <a href="#">“Processing the DMSUDATA CSL Routine”</a> on page 191.

When a CSL routine containing the *userdata* parameter is called, CMS in the user machine passes the user data to the server machine. The server machine does not, itself, process the user data. The server calls

the authorization CSL routine, setting the first flag in parameter 12 to 1 to indicate there is user data and placing the user data in parameter 13.

To reject a request because of incorrect or missing ESM user data, specify a return code of 8 for the authorization checking routine, the way you reject any read or write authorization check. If your ESM checks user data, but does not keep a catalog of authorized users, you can use return code 4 to have SFS check the authorizations. For example, suppose the user opens a file and specifies the *userdata* parameter (such as a password). Your ESM can determine whether the user data (password) is valid. Then it can return to the server with a return code of 4 to have the server check the authorizations. In any case, set the second flag in parameter 12 of the authorization routine to 1 on the return to the server to indicate your ESM has processed the user data.

If SFS sets the first flag in parameter 12 to 1 on the authorization call, but the second flag is still 0 on the return from the ESM, the server knows something is wrong. Someone specified the *userdata* parameter on a CSL routine, but your ESM does not support user data. In this case, CMS returns an error code to the program with an appropriate reason code that indicates "not found or not authorized".

### ***Special Considerations for GRANT and REVOKE***

If you want your ESM to process ESM user data for grant and revoke functions (request codes 38 and 41), do not use generic profiles. That is, when the server calls your ESM with a "generic query", you should always respond with a return code of 4, which indicates a generic profile does not exist. If you indicate a profile does exist, the server does not allow the grant and revoke functions for the protected objects.

To grant authority for an object (DMSGRAV) or revoke authority for an object (DMSREVOV), the user must own the object. The server already knows from its catalogs whether the user owns the object. However, the server still calls the ESM for a read check to see if the object is protected by the ESM and to give the ESM the opportunity to reject the grant or revoke request. A grant request is not allowed if the object is protected by the ESM. If the ESM defers authorization checking to the server, the grant is allowed.

For DMSREVOV, the server processing varies depending on whether *userdata* is specified on the call. When *userdata* is not specified on DMSREVOV, the server does not let the ESM reject the request. Instead, if the ESM returns a 0 or 8 on the read authorization check, the server revokes the authority, but returns a warning reason code to the application. The warning indicates the object is protected by the ESM.

If *userdata* is specified, a return code of 8 causes the server to reject the revoke request. This lets your ESM either permit or prevent revocations of SFS authority based on what is specified for *userdata*. A return code of 0 causes the server to revoke the authority and return a warning code indicating the object is protected.

In all cases, a return code of 4 causes the server to revoke the authority without returning a warning to the application.

### **Processing the DMSUDATA CSL Routine**

Processing for the DMSUDATA CSL routine differs slightly from the general ESM user data processing. The purpose of the DMSUDATA routine is to pass data to your ESM along with a syntactically correct file, external object, directory, or alias name. Note that although CMS checks the syntax of the file, external object, directory, or alias name, it does not verify the existence of the object before calling your ESM. In other words, your ESM should not assume the object exists.

Rather than create a new call to the ESM, the server uses a read authorization check to pass the DMSUDATA data to the ESM. Your ESM should process the user data, set the second flag in parameter 12 of the authorization CSL routine to 1, and set one of the following return codes in the *reason\_code\_3* field of parameter 2:

#### **Value**

#### **Meaning**

**0**

Processing is successful. DMSUDATA returns to the user's program with a 0 return code (assuming no other problems, such as a server error, occur).

**2**

Processing is successful and the server should notify user machines the file, external object, or alias is no longer ESM-protected. DMSUDATA returns to the user's program with a 0 return code (assuming no other problems, such as a server error, occur).

**3**

Processing is successful and the server should notify user machines the file, external object, or alias is now protected by the ESM. DMSUDATA returns to the user's program with a 0 return code (assuming no other problems, such as a server error, occur).

**8 or other**

Processing is unsuccessful. DMSUDATA returns to the user's program with an 8 return code and a 44400 reason code.

## Password Protection for Objects

Because the server does not operate on the ESM user data, but passes it on to the ESM, it can be of a form and content defined by the ESM. You might use the data to implement password protection for files, external objects, directories, and aliases. That is, your ESM can be coded to expect a password to be in the 80-byte parameter. If the password does not match the one defined for the object, your ESM can reject the operation (that is, return a "not authorized" result to the program).

In this scenario, you could use the DMSUDATA routine to associate password information with an object. (Conveniently, the DMSUDATA routine lets you specify a file or directory as part of its parameter list, as well as 80 bytes of ESM user data.) Or, you could use DMSUDATA to change existing passwords.

Although the ESM user data support is well-suited for password protection, it is not limited to that function. You might, for example, use it to trigger other operations not related to security. That is, your ESM might perform some secondary function such as starting a program in another virtual machine whenever a keyword is received as part of the user data.

Whatever you choose to implement, keep in mind it is your responsibility to communicate the content and meaning of the data to application programmers so they can properly use your ESM. When describing the *userdata* parameter, *z/VM* books refer the reader to the ESM documentation.

## Determining Protection Status

Application programmers can tell whether a file is protected by using the Extract/Replace facility (CSL routine DMSERP). The information name to specify is FILE\_ESM\_PROTECT.

Programmers can also use these CSL routines to determine protection status:

DMSEXIFI - SFS Exist - File

DMSEXIST - SFS Exist

DMSGETDF - SFS Get Directory - File (after calling DMSOPDIR)

DMSGETDI - SFS Get Directory (after calling DMSOPDIR)

DMSGETDS - SFS Get Directory - Searchauth (after calling DMSOPDIR)

DMSGETDT - SFS Get Directory - Auth (after calling DMSOPDIR)

DMSGETDX - SFS Get Directory - File Extended (after calling DMSOPDIR)

Although programmers can use these CSL routines, using DMSERP is faster. It gets the protection status from the user's virtual storage, while the other CSL routines request the data from the server. All these CSL routines are documented in *z/VM: CMS Callable Services Reference*.

Note that the protection status for a file is maintained in virtual storage only when its parent directory is accessed. If your application does not access the directory, it cannot use the Extract/Replace facility (CSL routine DMSERP) to determine the protection status of a file.

To tell whether a directory is protected, use the appropriate CSL routine from the preceding list (not DMSERP). There is no extract/replace information name for the protection status of directories.



### ***Maintaining Protection Status***

If you want your application programmers to use the extract/replace facility to determine the protection status of files, your ESM needs to help keep the users' virtual storage up to date. That is, when the protection status of a file changes, your ESM must tell the server to notify users the status has changed.

Your ESM can tell the server of a protection status change by setting the return code values when processing the DMSUDATA routine. One return code means the routine was successful and the file is now protected. Another return code means the routine was successful and the file is no longer protected.

When your ESM returns to the server, the server examines the return code and makes a note of the protection status. The server knows which users have information about the file in their virtual storage, so it knows which users need to be notified.

When one of those users next requests *any* function from the server, the server (as part of its response) passes the status change to the user machine. The user or application program is not aware of additional data—CMS intercepts it. On any subsequent extract of the file protection status, the user will see the new status.

Note that the server does not immediately broadcast the change in status. It waits until the user's next request. Consequently, the protection status of a file may be momentarily out-of-date.

**PI end**



---

## Chapter 9. Managing Storage

This chapter describes how to manage a file pool's physical DASD storage. Storage management includes:

- Rearranging existing storage groups (applicable only to repository file pool servers)
- Adding DASD space to a file pool (applicable only to repository file pool servers)
- Getting information about space in a file pool (applicable only to repository file pool servers)
- Moving file pool minidisks to different physical devices (applicable to all file pool servers)
- Moving an entire repository file pool to different physical devices (applicable only to SFS file pool servers)
- Removing space from a file pool (applicable only to repository file pool servers)
- Setting the space warning threshold (applicable only to repository file pool servers)
- Using DFSMS/VM for storage management (applicable only to repository file pool servers)
- Activating the SFS storage use exits

---

### Rearranging Existing Storage Groups

Before adding DASD space to a file pool, determine whether that space is really needed. It might be possible to rearrange existing user storage groups or catalog storage groups.

#### User Storage Groups

If you think you need to add DASD space to a file pool, you may be able to move users to under-utilized storage groups instead. If, for example, you received a warning message storage group 4 is short on storage, you may be able to move a few storage group 4 users elsewhere. You can use the QUERY FILEPOOL STORGRP command to determine whether sufficient free space exists in other storage groups. For a description, see [“QUERY FILEPOOL STORGRP” on page 652](#). See [“Moving an SFS User or BFS File Space in Multiple User Mode” on page 95](#) if you decide to move users.

#### Catalog Storage Groups

If you receive a warning message the catalog storage group (storage group 1) is short on storage and you have not reorganized the catalogs in a long time, consider reorganizing the catalogs before adding additional DASD space. The catalog indexes could be using DASD space inefficiently because they are fragmented. When the catalogs are reorganized, enough DASD space may be freed so you no longer need to add any. See [Chapter 10, “Reorganizing the File Pool Repository Catalogs,” on page 213](#) to determine whether a reorganization would be beneficial.

---

### Overview of Adding DASD Space to a File Pool

<b>Attention:</b>
When adding DASD space to any of the storage groups in the file pool, you must add one or more new minidisks; you cannot enlarge an existing minidisk.

If you determine you do need additional DASD space, you should next consider whether you should add it to an existing storage group or whether you should define a new storage group. In the latter case, you would move users from the storage groups that are short on storage into the new storage group. The primary reason for defining a new storage group is to keep existing storage groups from getting so large

they are inconvenient to back up or restore. If you are going to define a new storage group, you must also define one or more minidisks in the z/VM directory for the file pool server machine that owns the file pool.

You can add DASD space to any of the storage groups when the file pool is in dedicated maintenance mode or multiple user mode. If you can wait to add the minidisks until the file pool can be stopped, use the dedicated maintenance mode command, FILESERV MINIDISK, to add minidisks to the file pool. Because the FILESERV MINIDISK command is a dedicated maintenance mode command, you can determine the least disruptive time to the file pool users to stop the file pool and add minidisks.

But, there may be times when you need to add minidisks to the file pool immediately and cannot wait until the most convenient time to stop the file pool. You can add one or more minidisks while the file pool is in multiple user mode by using the FILEPOOL MINIDISK command. Because the FILEPOOL MINIDISK command is processed while the server is in multiple user mode, you can add minidisks with minimal disruption to the file pool users because you do not have to stop the file pool.

Adding minidisks to a file pool significantly changes the control data in the file pool. Therefore, it is recommended you back up the control data. To be able to back up the control data, you should have the BACKUP startup parameter defined in the *serverid* DMSPARMS file.

## Required Tasks Before Adding Minidisks

New minidisks are added to storage groups of your choice. Minidisks can be added to the catalog storage group (storage group 1) as well as any user storage group. You need to 1) determine the new allocations, and 2) update the z/VM directory to add minidisks to the file pool, regardless of which mode the file pool is in.

After this, follow separate instructions depending upon whether you want to add minidisks in dedicated maintenance mode using the FILESERV MINIDISK command, or add minidisks in multiple user mode using the FILEPOOL MINIDISK command.

This section describes the common tasks for adding minidisks.

### Determine the New Allocations

Whenever space is added to the file pool, you must determine

1. Which storage groups will be given the space (new or existing)
2. How much space each storage group needs
3. Which physical devices should be used.

You usually have the answer to the first question before you get to this chapter. If you use a "leave it alone" approach to operating the server, you have probably received a warning message that a particular storage group was running out of storage space. If, instead, you occasionally monitor storage consumption, you may have decided to increase the space of a few storage groups based on usage trends.

In any case, you should consider issuing the QUERY FILEPOOL STORGRP command again to make sure you know the numbers of the storage groups that need space. (See [“Getting Information about Space in a File Pool”](#) on page 204 if you are not sure how to check storage.) Write down those numbers for a later procedure.

Enter a QUERY FILEPOOL OVERVIEW command and note the Undefined Addressable 4K Blocks value for the file pool. This tells you how many more blocks you can still add before having to regenerate the file pool. See [“Limit Monitoring”](#) on page 58 for more information about monitoring Total physical file pool space. The file pool must be available for multiple user access when you enter the QUERY FILEPOOL STORGRP or QUERY FILEPOOL OVERVIEW commands.

Next, decide how much space to give each of the storage groups. If you have specific space requirements, you should calculate them in units of 4KB blocks (to later be converted to cylinders or FB-512 blocks on specific devices). For example, suppose you plan to move data previously stored on user-owned minidisks to the file pool. You can determine how much space those files occupy by issuing a QUERY DISK command for that minidisk. Use the BLKSIZE and BLKS USED values to estimate the number of blocks you

need. If the block size of the minidisk is not 4096, you need to divide to convert it. Suppose you enter a QUERY DISK A command and see the following:

```

LABEL  VDEV M  STAT  CYL TYPE BLKSIZE  FILES  BLKS USED-(%) BLKS LEFT  BLK TOTAL
MB191  191 A   R/W   2 3380 2048    11    63-12    477    540
Ready;

```

The example shows sixty-three 2048-byte blocks are used, which means roughly thirty-six 4096-byte blocks would be needed.<sup>2</sup> More space may need to be allocated to the catalog storage group to handle the additional files.

If you do not have specific space requirements, you might make an estimate of the number of 4KB blocks needed based on past experience. Or, you can forego estimations altogether and add an arbitrarily chosen amount, such as an entire DASD volume or a half of a volume. You do not need to supply the size of the minidisk to FILESERV MINIDISK or FILEPOOL MINIDISK processing. It computes the number of blocks—any estimates you do are strictly for your own space management purposes.

Adding an entire DASD volume at a time certainly eases space management (let the file server manage the space), but could cause you some problems if you later want to move the file pool minidisks to smaller-capacity devices. Difficulties could arise because a file pool minidisk must be replaced with a minidisk of *equal* or *greater* size. You cannot replace a single large file pool minidisk with two smaller ones.

Suppose, for instance, you are setting up a file pool on a large processor with the intent of later moving that file pool to a smaller departmental processor. In this case, you would not want to allocate entire volumes of IBM 3380s to the file pool because DASDs of that capacity may not be available on the smaller processor. Unless you are certain that you will *never* want to replace the file pool DASDs with models having a smaller capacity, you should consider using a more-conservative approach to allocating space to the file pool.

If, on the other hand, you are planning to migrate to higher-capacity DASDs in the future, you might consider choosing a minidisk size on your current device that could be moved to the higher-capacity drive with minimal waste. You can think of these minidisks as "common denominator" minidisks. For example, suppose your current devices can hold about 75% of what your planned devices could hold. You might define minidisks on your current devices that are 50% of the capacity of your planned device. When you move the minidisks to your new devices, 2 of the "old" minidisks can be moved to the new device. The new device would be almost 100% full. If, instead, you allocated entire volumes of your current devices as minidisks, only one of those minidisks would fit on each new device (the entire current device would fill about 75% of the new device). 25% of each new volume would be unused. While you could use the space for other purposes, your file pool might occupy more volumes than you had originally hoped.

If you use a "common denominator" approach to defining minidisks, keep in mind you should factor in an allowance for differences in device geometry and in the way CP and CMS use those devices. It may be necessary to make the minidisks on the new devices slightly larger (in terms of raw megabyte (MB) capacity) than minidisks on the old. Therefore, when you determine the "common denominator" size, it is best to reduce it by a percent or two. In the above example, you might use 48% or 49% instead of 50%. This might cause some waste on the new devices, but far less waste than otherwise could be caused if your "common denominator" minidisks did not fit properly on the new device.

You can use this "common denominator" principle also if your installation currently has a number of different DASD models, and you want to be able to move file pool minidisks among them.

Another factor you might want to weigh is that after a minidisk is allocated to the file pool there is no easy way to reclaim that space. You can move the minidisk to a different device, as was just mentioned, but you cannot easily remove the space altogether. (See "Removing Space from a File Pool" on page 206 for one technique.) If you are committed to using file pools instead of user-owned minidisks, however, it is easier to give the DASD space to the file pool and let the server worry about managing it.

<sup>2</sup> The exact number of blocks would vary slightly because the number of control blocks CMS needs for a minidisk varies according to the block size and the number and size of the files on the minidisk.

If you are reluctant to allocate an entire volume to a particular storage group because you suspect that storage group may be under-utilized, keep in mind users from other storage groups can be moved to the group. This, by the way, is a good technique to use if one storage group becomes an I/O bottleneck because it is over-used. You can shuffle users back and forth between storage groups to balance storage group workload. You are, in effect, managing users—the file pool server manages the DASD.

Having decided how much space is needed, you next need to decide on which device (or devices) the space should be allocated. If you have just purchased a new device and want to assign its space to the file pool or if you have free space on only one device, the answer is obvious. If you have a choice of many devices, some of which may have other data on them, you might want to be more selective so disk access performance is optimized. See [“Where to Place the Catalog Minidisks”](#) on page 255 and [“Where to Place the User Minidisks”](#) on page 255 for guidance on minidisk placement.

## Update the z/VM Directory

Using your local operating procedures, update the z/VM directory entries of the file pool server machine that owns the file pool. Add MDISK control statements for the new minidisks. Example directory entries are shown in [Figure 25](#) on page 198.

```
MDISK 307 3380 cyl 10 volser1 R readpw writepw
MDISK 308 3380 cyl 10 volser2 R readpw writepw
MDISK 411 3380 cyl 200 volser3 R readpw writepw
MDISK 4AB 3380 cyl 200 volser4 R readpw writepw
```

*Figure 25. Example MDISK Statements for Additional File Pool Minidisks*

### Notes:

1. Do not define temporary disks or virtual disks in storage as file pool minidisks.
2. Do not add LINK control statements to the z/VM directory for file pool minidisks. File pool server machines link to the file pool minidisks when needed.
3. The MDISK statements shown are for count-key-data devices, so you must supply a cylinder relocation factor (*cyl*). For FB-512 devices, specify the appropriate block relocation factor (*blkr*) and number of blocks (*blks*).
4. For FBA storage group minidisks, make sure they are aligned on a 4KB boundary. See [“Restrictions for Using Data Spaces on FBA Minidisks”](#) on page 244 for details on aligning FBA minidisks on 4KB boundaries.
5. Set the volume serial numbers (*volser*) as appropriate for your DASD volumes.
6. **The MDISK control statements for the file pool must include both a read and a write password. Also, it is strongly recommended the minidisks have an access mode of ‘R’.** Specify your own read and write passwords for *readpw* and *writepw*.
7. All storage group minidisks should be eligible for expanded storage caching of minidisks. Do not specify MINIOPT NOMDC.
8. Virtual device numbers can be used in any order. A file pool server does not rely on any particular device addresses.

When you enter the FILESERV MINIDISK or FILEPOOL MINIDISK command, you must be prepared to supply the virtual device numbers of the minidisks. You should write down your choices of virtual device numbers. (You need them for a later step.)

9. In the example, minidisks 307 and 308 are shown as having 10 cylinders each. The intent is to assign both of those minidisks to the catalog storage group. Notice the total addition of 20 cylinders to the catalogs is split into two 10-cylinder minidisks on separate devices. Our fictitious administrator intends to add the 411 and 4AB minidisks to user storage groups 2 and 3.

At this time, you might want to update the MAXCONN operand of the OPTION control statement in the server's z/VM system directory entry. Increase the MAXCONN value by the number of minidisks you are

adding. If, for example, MAXCONN was set to 202 and you are adding four minidisks, set MAXCONN to 206. If MAXCONN is already set to a high value (for example, 2000), there is no need to update it.

After you have completed these two mandatory tasks, decide if you want to add minidisks in multiple user mode (using the FILEPOOL MINIDISK command), or add minidisks in dedicated maintenance mode (using the FILESERV MINIDISK command). If you have scheduled to stop the file pool and add minidisks using the FILESERV MINIDISK command, follow the instructions on [“Adding Minidisks in Dedicated Maintenance Mode”](#) on page 201.

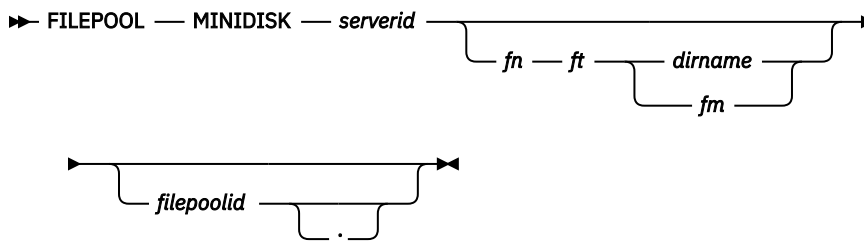
## Adding Minidisks in Multiple User Mode

To use the FILEPOOL MINIDISK command, you should be logged on to a machine that has administrator authorization for the file pool to which you will be adding minidisks. If the administrator is on a system remote to the server machine, you need to FORMAT and RESERVE the minidisks prior to issuing this command. See the FORMAT and RESERVE steps in [“Replacing File Pool Minidisks”](#) on page 129. Depending on the size of the minidisks, the FORMAT and RESERVE processing might cause a delay.

The FILEPOOL MINIDISK command allows you to add any number of minidisks. However, the total number of minidisks must not exceed the MAXDISKS value specified in the file pool server's POOLDEF file. If you plan to add more than one minidisk, you need to create a Minidisk Definition Control Statement file on the administrator's work disk. (This is also referred to as just control statement file in this section.) The description about this file and instructions on how to create it are shown in [“Creating a Minidisk Definition Control Statement File”](#) on page 202. You can add a single minidisk at a time without creating the control statement file. However, you will still need the same information available as if you had created a minidisk definition control statement file. You will be providing this information as response to prompts by the command processing.

### Enter the FILEPOOL MINIDISK command

When you have created the control statement file (if adding more than one minidisk), enter the FILEPOOL MINIDISK command. An abbreviated format of the command is:



For *serverid*, specify the user ID of the server machine that owns the file pool to which you are adding the minidisks.

If you are adding more than one minidisk, for *fn ft dirname* or *fn ft fm* specify the file that contains the control statements which you have already created. *dirname* represents the SFS directory containing the file. *fm* represents a minidisk or accessed directory containing the file. If you do not specify the file name, file type, and file mode or fully qualified directory name, you will be prompted for input for a single minidisk.

For *filepoolid*, specify the filepool ID of the file pool to which you will add the minidisks. If you do not specify a filepool ID, the default filepool ID will be used.

For example, if you wanted to add minidisks to the local file pool VMSYSU which is owned by the VMSESRVU user ID, first make sure you have added the minidisks to the VMSESRVU user ID directory entry. If you have a minidisk definition control statement file named DISKS NEEDED A, you would enter:

```
filepool minidisk vmseervu disks needed a vmsysu
```

The FILEPOOL MINIDISK command updates the POOLDEF file in the server machine. It will create a temporary file, \$\$TEMP \$POOLDEF, during its processing on the same file mode as the POOLDEF file.

For more information, see [“FILEPOOL MINIDISK” on page 470](#).

### Completion of FILEPOOL MINIDISK Processing

When the minidisks have been successfully added by the FILEPOOL MINIDISK command, the following message will be displayed on the administrator machine's console:

```
DMS3922I n minidisk(s) were added to the file pool
```

If the BACKUP startup parameter is specified, the FILEPOOL MINIDISK command then schedules a backup of the control data which is directed to the current assignment of the control data backup file. The backup is performed independently of the command. That is, the command will complete prior to backup completing or even starting. The minidisks will not be available for use until the control data has been successfully backed up. You will get a reader file, \$\$\$SFS \$MSG\$, indicating the success or not of the control data backup. There is a possibility you may not get this file because of spool space problems or the network not being available. If you wait for the file longer than the time it usually takes for control data to be backed up, you should take a look at the server console. If the server indicates successful control data backup and that a file was sent to your user ID, the file pool and newly added minidisks are available for use. However, if there was a backup error, follow the recovery procedure shown in the next section.

### What to Do if FILEPOOL MINIDISK Is Unsuccessful

If the FILEPOOL MINIDISK command is unsuccessful for any reason, the action you take next depends on which step was unsuccessful.

If any command is unsuccessful during the sequence of steps that include the FILEPOOL MINIDISK function, the situation should be corrected and the FILEPOOL MINIDISK command entered again.

If the SFS server stops as a result of an error, the action you take depends on whether message DMS3922I was displayed on the SFS server console when the error occurred. Message DMS3922I is:

```
DMS3922I n minidisk(s) were added to the file pool
```

Take recovery actions as follows:

- Message DMS3922I is **not** displayed

If the server stopped and message DMS3922I is not displayed, and if there are messages indicating the error, take the appropriate action and rerun the command. Otherwise, if there are no messages, start the server and rerun the command.

- Message DMS3922I is displayed

If the server stopped and message DMS3922I is displayed, check to see if you also received a message indicating a backup has been started. If the server was unsuccessful prior to completing the control data backup, the server cannot be restarted until you enter the FILESERV BACKUP command to back up the control data. The minidisks are available for use after the server is successfully started.

If the server stopped and message DMS3922I is displayed, but you do not also get a message indicating a back up has been started, you need to edit the POOLDEF file. If the POOLDEF file does not contain the control statements for all the minidisks you added, you need to add them from the \$\$TEMP \$POOLDEF file. To do this, edit the \$\$TEMP \$POOLDEF file and delete all the lines *except* for the ones with the values you assigned to the minidisk or minidisks that were to be added. Then, copy the \$\$TEMP \$POOLDEF file to the bottom of the POOLDEF file which adds the statements to the POOLDEF file for the new minidisks.

After you have updated the POOLDEF file, if the BACKUP startup parameter is specified, you must enter the FILESERV BACKUP command before starting the file pool server. If the BACKUP parameter is not specified, restart the file pool server.



## Adding Minidisks in Dedicated Maintenance Mode

Follow the steps below to add minidisks when the file pool is in dedicated maintenance mode.

### Step 1: Stop Multiple User Mode Processing

If the server that owns the minidisks is currently running in multiple user mode, stop server processing by entering the STOP NOBACKUP operator command. It is not necessary to take a control data backup before issuing the FILESERV MINIDISK command, because you will have to take one immediately after FILESERV MINIDISK processing. However, if you are running with BACKUP in effect, ensure you have a valid control data backup before proceeding.

### Step 2: Create a Minidisk Definition Control Statement File

Create a Minidisk Definition Control Statement file on the work disk of the file pool server machine by following instructions in [“Creating a Minidisk Definition Control Statement File”](#) on page 202.

### Step 3: Correct the Server Startup Parameters

The FILESERV MINIDISK command uses the server startup parameters. Make sure the FORMAT startup parameter is specified in the DMSPARMS file or is allowed to default. To check, edit the *serverid* DMSPARMS file. If the NOFORMAT startup parameter is specified, delete it from the file. No other parameters need to be changed.

### Step 4: Enter the FILESERV MINIDISK Command

Now enter the FILESERV MINIDISK command to process the control statements in the file created in Step 2 above. The format of the command is:

```
►► FILESERV — MINIDISK — fn — ft —————►
                               |
                               | fm
                               |
                               |
```

For *fn* *ft* and *fm*, specify the file that contains the control statements. If you do not specify a file mode, the first file in the search order having the specified file name and file type is used. For example, if the file was EXPAND DATA A, you would enter:

```
fileserv minidisk expand data a
```

The FILESERV MINIDISK command is a dedicated maintenance mode command. It updates the POOLDEF file and accesses all the other file pool minidisks. It will create three temporary files during its processing. They are created on the first file mode in the server's search order that is accessed in read/write mode:

```
$$TEMP $$INPUT
$$TEMP $$POOLDEF
$$TEMP $$UPDATE
```

The files are erased when FILESERV MINIDISK processing successfully completes. If FILESERV MINIDISK processing is unsuccessful for any reason, follow the instructions in [“What to Do If FILESERV MINIDISK Is Unsuccessful”](#) on page 201.

When the FORMAT startup parameter is in effect, FILESERV MINIDISK processing entered FORMAT and RESERVE commands for the new minidisks. Depending on the size of the minidisks, the FORMAT and RESERVE processing might cause a delay. The minidisks are formatted with a 4096-byte (4KB) block size.

### What to Do If FILESERV MINIDISK Is Unsuccessful

If the FILESERV MINIDISK command is unsuccessful for any reason, the action you should next take depends on whether the message DMS3922I was displayed:

```
DMS3922I n minidisk(s) were added to the file pool
```

If the error occurred *before* message DMS3922I was displayed, follow the instructions from the messages (if necessary) and correct the error, and then rerun the FILESERV MINIDISK command. Then take these actions:

1. If the rerun of the command is unsuccessful because of an attempt to add duplicate minidisks, edit the POOLDEF file and add the FILESERV MINIDISK control statements to it. For example, if the file pool ID is VMSYS and the FILESERV MINIDISK control statements are in the file EXPAND DATA A, you would enter:

```
xedit vmsys pooldef a
bottom
get expand data a
file
```

2. If the rerun is unsuccessful for another reason, rerun the command again. If the command is still unsuccessful, restore the control data from the backup made in Step 2 of the add minidisk process, and start the add minidisk steps again. (If you were using non-SFS facilities, you have to restore the entire file pool.)
3. If the rerun of the command is successful, no further action is needed.

If the error occurred *after* message DMS3922I was displayed, do not rerun the command. Manually update the *filepoolid* POOLDEF file as described in [“1” on page 202](#) above.

### Step 5: Make a Backup

Because the FILESERV MINIDISK command significantly changes the control data in the file pool, you should back up the control data before allowing multiple user access.

If you use file pool facilities for backing up the control data, you must make another control data backup in dedicated maintenance mode by entering the FILESERV BACKUP command. The FILESERV START command will not work until the backup is made.

If you use a non-file pool facility for backing up the file pool (in which a backup of the entire file pool is made at one time), you can defer making a backup until the time it is usually scheduled. A data loss prior to your next backup would cause your file pool to regress to the state it was in before the minidisks were added. You would need to add the minidisks again.

### Step 6: Start Multiple User Mode Processing

After the backup is made, the addition of the minidisks is complete. You can start the server to access the file pool in multiple user mode.

## Creating a Minidisk Definition Control Statement File

Create a file containing control statements to be used during FILESERV MINIDISK or FILEPOOL MINIDISK processing. The control statements identify an internal name (known as a *ddname*) a server uses to identify the minidisk. They also identify the minidisk's virtual device number and the storage group to which it is to be assigned. An example file is shown in [Figure 26 on page 202](#).

```
* The following control statements are for the catalog storage group:
DDNAME=MDK00007   VDEV=307   GROUP=1
DDNAME=MDK00008   VDEV=308   GROUP=1

* These are for user storage groups 2 and 3:
DDNAME=MDK00009   VDEV=411   GROUP=2
DDNAME=MDK00010   VDEV=4AB   GROUP=3
```

Figure 26. Example Control Statements for FILESERV MINIDISK or FILEPOOL MINIDISK Processing

You can use any file name and file type for the file. For FILEPOOL MINIDISK, the file should be on an accessed administrator's minidisk or in a directory. For FILESERV MINIDISK it should reside on the file pool server machine's work minidisk.

Enter control statements on separate lines within positions 1 to 72. Control statements that begin with an asterisk (\*) or that have all blanks in positions 1 through 72 are ignored. Command processing uses the control statements to make additions to the POOLDEF file in the server, which identifies all file pool minidisks.

The format of the control statements is shown in [Figure 27 on page 203](#). Any number of spaces can be between operands.

```
DDNAME=MDKnnnnn  VDEV=vdev  GROUP=grpnum  BLOCKS=blocks
```

Figure 27. Format of the Control Statements for FILESERV MINIDISK or FILEPOOL MINIDISK Processing

### DDNAME=MDKnnnnn

This operand identifies an internal name by which the server will know the minidisk.<sup>3</sup>

*nnnnn* can range from 1 to 32767 and must be used in ascending consecutive numeric sequence. It is incorrect, for example, to use DDNAME=MDK00005 before DDNAME=MDK00004. It is also not correct to skip numbers. DDNAME=MDK00005 must follow DDNAME=MDK00004.

When adding minidisks, you must know what the last-used MDKnnnnn number is. If you cannot remember what the last number was, you can determine it by taking the following step:

- If you are logged on as an administrator, that is, you plan to use the FILEPOOL MINIDISK command, you can get this information by using the QUERY FILEPOOL MINIDISK command. For a description, see [“QUERY FILEPOOL MINIDISK” on page 586](#). In the response of the command under Storage Group Minidisk Information, find the highest number under the column Minidisk Number. This is the *nnnnn* part of the last used MDKnnnnn number.

For example, if your POOLDEF file contained the following DDNAMEs:

```
DDNAME=MDK00001      VDEV=303      GROUP=1  BLOCKS=292
DDNAME=MDK00002      VDEV=304      GROUP=2  BLOCKS=292
DDNAME=MDK00003      VDEV=305      GROUP=3  BLOCKS=292
DDNAME=MDK00004      VDEV=306      GROUP=4  BLOCKS=292
DDNAME=MDK00005      VDEV=193      GROUP=4  BLOCKS=1190
```

and you enter the QUERY FILEPOOL MINIDISK command and look at the STORAGE GROUP MINIDISK INFORMATION, you would see:

Storage Group No.	Minidisk Number	4K Blocks In-Use	4K Blocks Free	Virtual Address
1	1	124 - 42%	168	0303
2	2	101 - 35%	191	0304
3	3	71 - 24%	221	0305
4	4	149 - 51%	143	0306
4	5	115 - 10%	1075	0193

The next available MDKnnnnn is MDK00006 (1 plus the last minidisk number shown).

- If you are logged on the file pool server machine, that is you plan to use the FILESERV MINIDISK command, edit the POOLDEF file for the file pool. The highest MDKnnnnn value will be at the bottom of the file. To edit the POOLDEF file of the VMSYS file pool, for example, you would enter:

```
xedit vmsys pooldef
```

### VDEV=vdev

For *vdev*, specify the virtual device number of the minidisk being added to the file pool.

<sup>3</sup> Command processing builds FILEDEF commands to identify the file pool minidisks. The MDKnnnnn values are used as the *ddnames* for those FILEDEF commands.

### **GROUP=grpnum**

For *grpnum* specify the number of the storage group to which you want the minidisk assigned. Storage group 1 contains the file pool catalogs. Storage groups 2 through 32767 contain user data.

The storage group number you specify cannot be greater than the number specified for MAXDISKS during file pool generation. You can find the MAXDISKS value for a file pool in the POOLDEF file, or from the QUERY FILEPOOL OVERVIEW command if you are in multiple user mode processing. (See the Maximum Number of Storage Groups value in the output of the QUERY FILEPOOL OVERVIEW command.)

The first storage group to contain user data must be number 2, but after that groups can be numbered in any order. Assigning a minidisk to a storage group that does not exist defines a new storage group.

### **BLOCKS=blocks**

This is an optional parameter and is handled automatically just as it is during FILESERV GENERATE. See [“FILESERV GENERATE” on page 513](#) for more information on the BLOCKS parameter.

For additional information on when it might be appropriate to provide the BLOCKS parameter, see [“Replacing File Pool Minidisks” on page 129](#) and [“Moving a File Pool” on page 204](#).

## Getting Information about Space in a File Pool

---

To find out information about the physical space allocated to a file pool, enter the QUERY FILEPOOL MINIDISK command or use the Query User Storage Group CSL routine. To check the number of 4KB blocks currently occupied on minidisks in the VMSYS file pool, for instance, you would enter:

```
query filepool minidisk vmsys
```

For a complete description, see [“QUERY FILEPOOL MINIDISK” on page 586](#). If you are interested in more extensive monitoring of DASD usage, see [z/VM: Performance](#).

## Moving File Pool Minidisks to Different Physical Devices

---

There may be times when you want to move your file pool minidisks to different DASD devices. For instance, if you installed new devices and wanted to get rid of the old ones (or use them for something else).

To move file pool minidisks, follow the instructions in [“Replacing File Pool Minidisks” on page 129](#).

## Moving a File Pool

---

It is possible to move (or copy) an entire repository file pool, rather than individual minidisks, from one set of DASD to another. You may want to do this, for example, when an image of a file pool is needed on another system, when hardware is being upgraded, or when you want to change the configuration of your file pool storage groups. The following procedure describes how to accomplish this. Note that if you want to change the configuration of your file pool storage groups, you must use the FILEPOOL UNLOAD command followed by the FILEPOOL RELOAD command. If you use the FILEPOOL BACKUP command followed by the FILEPOOL RESTORE command, you are not allowed to change the configuration of the file pool storage groups. Note that this procedure only applies to repository file pools, and *not* to dedicated CRR or FIFO file pools.

1. Back up the user data.

Back up the user data in all storage groups using the FILEPOOL BACKUP or FILEPOOL UNLOAD command. See [“Backing Up User Data” on page 111](#) for this procedure.

2. Define the new server machine.

Using your local operating procedures, update the z/VM system directory entries for the new server. See [“Step 4: Define a Server Machine” on page 256](#) for a description of this procedure. Make sure you have MDISK control statements for all the required minidisks:

- Work minidisk on which the POOLDEF file resides

- Two file pool log minidisks
- Two file pool CRR log minidisks (if needed)
- Storage group 1 minidisks
- Storage group 2 through  $n$  minidisks

Record the new virtual device numbers so you can properly update the POOLDEF file at a later step.

If you are using FILEPOOL BACKUP followed by FILEPOOL RESTORE to perform the copy, make sure all new storage group minidisks are the same size as the corresponding old minidisks or larger. If the new minidisks are of the same device type as the old ones, define the same number of cylinders or blocks. (When both the devices are FB-512 devices, you can consider them to be the same device type, even if they are different DASD models. Make sure the FB-512 devices start and end on a 4KB boundary.) If the minidisks are on a different device type, you should define the new minidisk to be slightly larger than the old. To determine approximately equivalent sizes of minidisks on various devices, see your device documentation.

**Attention:**

If you are using FILEPOOL BACKUP followed by FILEPOOL RESTORE to perform the copy, you cannot increase the capacity of a storage group by declaring a larger minidisk in the new file pool server or by combining two or more of the existing minidisk into one large minidisk. To increase the size of a storage group (including storage group 1) you must add minidisks to it using the FILESERV MINIDISK command or the FILEPOOL MINIDISK command after your new file pool is generated. For more information, see [“FILESERV MINIDISK” on page 525](#) or [“FILEPOOL MINIDISK” on page 470](#).

3. Update the POOLDEF and DMSPARMS files.

Copy the POOLDEF and DMSPARMS files for the old file pool to the new file pool server machine's work disk (A-disk).

Change the file name of the DMSPARMS file to the user ID of the new server machine (the *serverid*).

If you are going to rename the new file pool, change the file name of the POOLDEF file to the new file pool ID.

Change the parameters in the DMSPARMS file as necessary. For example, you may want to change some of the startup parameters like FILEPOOLID or ADMIN.

Change the POOLDEF file as needed.

**Attention: If you are using FILEPOOL BACKUP followed by FILEPOOL RESTORE to perform the copy, the virtual device numbers are the only values that can be changed in the POOLDEF file. Values for all other parameters in that file must remain the same as they were for the old file pool.**

See [“Step 7: Create a Startup Parameter File” on page 263](#) and [“Step 8: Generate the File Pool” on page 267](#) if you need to review the DMSPARMS and POOLDEF file entries.

4. Generate the new file pool.

Generate the new file pool using the FILESERV GENERATE command. See [“FILESERV GENERATE” on page 513](#).

5. Back up the control data.

If you have specified the BACKUP startup parameter, verify there is a valid control data backup destination defined in the POOLDEF file. If necessary, redefine the backup file destination with the FILESERV DEFBACKUP command. See [“Defining or Changing the Default Control Data Backup File” on page 107](#) for more information.

Back up the server control data using the FILESERV BACKUP command, see [“Step 9: Back Up the Control Data” on page 270](#). Although there is no user data in the file pool at this time, this step is required to start the server.

6. Start multiple user mode operation.

## Removing Space

Start the server by entering the FILESERV START command, and disconnect from the server machine by entering the commands:

```
#CP SET RUN ON  
#CP DISC
```

### 7. Restore the user data.

Log on an administrator machine.

Restore the user data in all storage groups using the FILEPOOL RESTORE command if you used FILEPOOL BACKUP for the backup, or the FILEPOOL RELOAD command if you used the FILEPOOL UNLOAD for backup. The procedure is outlined in [“Restoring User Data” on page 115](#).

**Attention: Restore the storage groups before doing other file pool operations.** If you use the storage group (for example to enroll a new user) prior to restoring it, the storage group backup becomes obsolete. In the case of FILEPOOL RESTORE, restoring the storage group from that backup would cause unpredictable results. The restore might succeed, but the data in the storage group or catalog data related to the storage group would be corrupted. Your only recourse would be to generate the file pool again and restore the storage groups using backups made prior to the preceding operation. In the case of FILEPOOL RELOAD, restoring would wipe out any users that were enrolled after the generate.

### 8. Back up the control data.

If you have specified the BACKUP startup parameter, log on the server machine and back up the control data using the BACKUP server operator command after restoring all the user storage groups. This backup reflects all the changes made to the control data by the restores of the user storage groups. It also frees the log space consumed by those restores. See [“BACKUP” on page 366](#) for more information on the BACKUP server operator command.

## Removing Space from a File Pool

When a minidisk is allocated to a file pool, either by FILESERV GENERATE, FILESERV MINIDISK, or FILEPOOL MINIDISK processing, the POOLDEF file is changed to show the addition of the minidisk. The control minidisk is also updated to map the additional physical blocks that are now available. After FILEPOOL MINIDISK processing completes, or when FILESERV START is entered after FILESERV MINIDISK processing completes, a file pool server machine begins using blocks in the new minidisk to satisfy user requests in that storage group. Because user requests are unpredictable and because the server uses space from all minidisks in the storage group, it is almost impossible to tell what files, or portions of files, are stored on a particular minidisk. Thus, there is no command to let the minidisk be reclaimed. After you allocate space to a file pool, file server processing assumes the space will be there forever.

If, however, it is essential to reclaim the minidisk space added by FILESERV MINIDISK or FILEPOOL MINIDISK processing, you have one option that works only for user storage groups. It is not possible to reduce the amount of DASD space in the catalog storage group (storage group 1).

1. Move all users from the storage group in which the minidisk resides to some other storage group using the two commands FILEPOOL UNLOAD and FILEPOOL RELOAD or the FILESERV MOVEUSER command. Follow the procedure in [“Moving an SFS User or BFS File Space in Multiple User Mode” on page 95](#) or the procedure in [“Moving an SFS User in Dedicated Maintenance Mode” on page 96](#).
2. Disable all access to the storage group using the DISABLE file pool server machine operator command. Specify both the EXCLUSIVE and DETACH options of the DISABLE command. For example, to disable storage group 5 you would enter:

```
disable group 5 exclusive detach
```

For a complete description, see [“DISABLE” on page 412](#).

After the storage group is disabled, you should abandon its use forever. Be sure to warn other file pool administrators about the storage group so they do not inadvertently enable it. To help prevent someone from inadvertently enabling the storage group, specify the FOR *owner\_userid*

option on the DISABLE command. For *owner\_userid*, specify a nonexistent user ID that would help remind administrators the storage group should not be enabled. The *owner\_userid* you use must have administrator authority. Use the ENROLL ADMINISTRATOR file pool administrator command to temporarily enroll *owner\_userid* as an administrator. From an administrator ID you would enter:

```
enroll administrator handsoff filepoolid:
```

You would then enter the DISABLE operator command:

```
disable group 5 exclusive detach for handsoff
```

Use the QUERY DISABLE operator command to display information about disabled storage groups or file spaces. For a complete description, see “QUERY DISABLE” on page 555.

3. The control statements for the minidisks in the POOLDEF file cannot be deleted. Also, the MDISK control statements for those minidisks in the z/VM system directory entry for the server machine cannot be deleted. You can, however, change the MDISK statements to allocate minimally-sized minidisks, thereby freeing DASD space. Retain the same virtual device numbers for these minidisks. See *z/VM: CP Planning and Administration* for information on how to update MDISK statements in a directory entry.
4. If you change an MDISK statement to minimally allocate a minidisk, you must use the following procedure to format and reserve the changed minidisk.
  - a. Enter the following CMS command to find a file mode (disk access letter) that is currently unused:

```
QUERY ACCESSED
```

If, for example, you do not have anything accessed as C, note it for use later in this procedure.

- b. Enter the following CP command to make sure the minidisk is write accessible to the server machine:

```
LINK * vdev vdev W
```

where:

**\***

indicates the minidisk is defined in the z/VM directory entry for this virtual machine.

**vdev**

specifies the virtual device number used on the MDISK statement (use the same value twice).

**W**

indicates write access is required for this minidisk.

- c. Enter the following CMS command to re-initialize and label the minidisk:

```
FORMAT vdev filemode (BLKSIZE 4096
```

where:

**vdev**

is the same value you used for the CP LINK command previously.

**filemode**

specifies the disk access letter you obtained from the QUERY ACCESSED command.

The FORMAT command will prompt you for the minidisk volume label (disk label). You can enter any valid label value. FORMAT will also ask you whether it should erase all files on the minidisk. Answer 1 for YES to this prompt (after making sure you have specified the correct minidisk).

- d. Enter the following CMS command to allocate the minidisk as a block I/O file:

```
RESERVE filename MDKnnnnn filemode
```

where:



## Setting the Warning Threshold

### **filename**

is the file pool ID.

### **nnnnn**

is the ddname (for example, MDK00005).

### **filemode**

is the same value you used for the FORMAT command previously.

The RESERVE command will ask you whether it should erase all files on the minidisk. Answer 1 for YES to this prompt (after making sure the filemode letter is correct).

- e. Enter the following CMS command to release the minidisk:

```
RELEASE vdev (DET
```

where:

### **vdev**

is the same value you used for the LINK and FORMAT commands previously.

5. It is important to remember you (or any other user with administration authority) should **never** enable the storage group or enter commands (such as FILEPOOL BACKUP, FILEPOOL RESTORE, or FILEPOOL CLEANUP) that operate on the storage group. If you do, the results are unpredictable—file pool server processing will stop.

## Setting the Warning Threshold

---

File pool server processing displays a warning message when the physical DASD space in a storage group is 90% full. In computing the amount of physical DASD space that is consumed, the server counts all allocated blocks, including those allocated to support uncommitted work. You can change the percentage at which this message is displayed by specifying the GROUPTHRESH startup parameter in the DMSPARMS file. To have the warning message displayed at 80% instead of 90%, for instance, you would specify:

```
GROUPTHRESH 80
```

For more about the startup parameter, see [GROUPTHRESH](#).

## Using DFSMS/VM for Storage Management

---

DFSMS/VM can help you manage your SFS storage and move data to new storage devices quickly and easily with minimal impact to users.

Its main functions, accessible through interactive menus and DFSMS/VM commands, are:

### **Moving Data**

Enables you to move a CMS minidisk with one command.

### **Managing User Storage Groups**

Enables you to move and compress seldom-used files (or any files you choose) to lower-cost storage devices. These moved files are then considered to be in *migrated status* or just *migrated*. You can specify these files be automatically recalled when they are referenced, or they may be explicitly recalled. (For SFS, files in file control directories can be migrated; files in directory control directories cannot be migrated.) You can also specify files (in both file control and directory control directories) be automatically deleted under certain conditions. For example, if you have files that have not been referenced for a certain number of days, you can specify the files be automatically deleted.

### **Classifying Files**

Enables you to automatically assign a *management class* to a file or directory. A management class is a name that can be associated with a set of attributes that define criteria for storage management. These criteria determine when the files can be automatically placed in DFSMS/VM migrated status (that is, have their data moved into the DFSMS/VM storage repository), erased, or both.

See the DFSMS/VM books for complete information about how to use the functions of DFSMS/VM to help you manage file pool storage.



## Activating the Storage Use Exits (SFS and BFS)

---

The SFS storage use exits enable you to implement a policy regarding what happens when storage group space becomes exhausted, or when a user's file space limit is exceeded.

In SFS file spaces it is possible to consume storage in excess of the file space limit as long as:

- the storage consumption is not committed
- the storage group limit is not exceeded

This is a beneficial feature of SFS file spaces that allows for occasional, temporary excesses in file space storage consumption. It only becomes a problem when it is abused or where there are excesses in its practice such that storage group limit errors are encountered. You should consider activating storage use exits in which you can rollback selected SFS users that hold excessive temporary storage.

BFS file spaces do not permit temporary storage consumption in excess of the file space limits. Therefore, BFS storage consumption is always limited by the current maximum number of blocks set for the target BFS file space. Because BFS file spaces cannot directly contribute to the temporary storage excesses, they do not directly participate in these storage use exits designed to control the potential problem. The file pool server calls these exits by calling the DMSSFSEX CSL routine. DMSSFSEX is an installation-wide exit.

There are two file pool storage use exits in DMSSFSEX:

- The SFS server calls DMSSFSEX for the **file space usage** exit when a user's combined committed and uncommitted file space usage reaches or exceeds an integer multiple of the user's file space allocation limit. (This is the limit set for the user by the file pool administrator, either upon enrollment or in a later modification.) This is not applicable for BFS file spaces.

For example, suppose a user has a 100-block file space allocation limit and has already committed 20 of those 100 blocks. The user (or someone else) then starts copying a file into the file space, in 20-block units. (The blocks used for this are uncommitted until the copy completes.) When 80 blocks (4 times 20) are written, the total number of blocks used in the file space (the 80 plus the 20 previously committed) reaches 100. This is exactly one multiple of the user's allocation limit, and SFS calls the exit.

If five more 20-block units are copied into the file space, file space usage reaches 200 blocks, and the file pool server calls the exit again. If the user then starts writing 30-block units to the file space, and writes three units (90 blocks, for a total file space usage of 290 blocks), no exit is called. However, if the user writes one more 30-block unit, file space usage reaches 320 blocks, which exceeds three times the original file space allocation limit, and the file pool server calls the exit.

- The file pool server calls DMSSFSEX for the **user storage group full** exit when a user tries to write to a file space located in a storage group that has no more available space. This is not applicable for storage groups that consist only of BFS file spaces.

DMSSFSEX is the CSL name assigned to the DMS5XX module. DMS5XX contains dormant exit functions you can activate, or you can define your own.

### SFS General Data Buffer

As input to DMSSFSEX (DMS5XX), the file pool server supplies a pointer to a general data buffer. This buffer includes information such as which exit is being called and data about current file space usage. The buffer contains one list entry for the file space usage exit. The buffer contains one or more entries for the user storage group full exit; there is a corresponding entry in the buffer for each work unit and associated SFS file space in the storage group. There are no entries in the buffer for BFS file spaces in the storage group. Each list entry in the buffer includes a flag field (described later) set by the exits. To see exactly what information the file pool server provides to the exits, you can use the EXITBUFF macro to map the contents of the buffer. For information about EXITBUFF, see *z/VM: CMS Macros and Functions Reference*.

## Activating the Sample Exit Functions

The default function for both exits is to take no action when called, but to simply return control to the file pool server with a return code of 5. This return code requests the file pool server to suppress further calls to the exit. However, the DMS5XX module also contains sample functions you can activate by uncommenting a few lines in the source file, DMS5XX ASSEMBLE.

The sample functions in DMS5XX analyze the information passed in the general data buffer, as follows:

- For the file space usage exit, DMS5XX marks the user for rollback if the user's file space usage (both committed and uncommitted) is more than twice the user's allocated file space size.
- For the user storage group full exit, DMS5XX examines each entry in the list, determines the logical unit of work that has the most space in excess of file space allocation limits that is not allocated, and marks that user for rollback. In case of a tie, DMS5XX selects the one that is first in the list.

To mark a user for rollback, DMS5XX sets a flag in the EXIOUTFL (Out Flags) field in the list entry for that user in the general data buffer. Upon return from the exit, the server takes action based on this flag. If DMS5XX marks a user for rollback, the server rolls back that user. If DMS5XX does not mark *any* user for rollback, the server takes action according to which exit was called:

- For the file space usage exit, the file pool server continues usual processing.
- For the user storage group full exit, the file pool server rolls back the user who triggered the exit.

To make the revised DMS5XX available, see [“Making the New or Modified Exit Routine Available”](#) on page 211.

## Modifying or Replacing the Exit Routine

You can customize the storage use exits to meet your installation's needs by either modifying the IBM-supplied DMS5XX module or replacing it with your own version. The source file is DMS5XX ASSEMBLE. For general information about writing a CSL routine, see *z/VM: CMS Application Development Guide for Assembler*. For information about the CSLENTRY, CSLGETP, and CSLEXIT macros used in writing a CSL routine, see *z/VM: CMS Macros and Functions Reference*.

If you write your own CSL routine, make sure it returns the necessary information to the SFS server. The IBM-supplied CSL template file for the DMS5XX (DMSSFSEX) routine is DMS5XXTP TEMPLATE, shown in [Figure 28 on page 210](#). This file identifies the routine's input and output parameters. The general format of a CSL template file is described in *z/VM: CMS Application Development Guide for Assembler*.

3	3		3 Pairs Maximum, 3 Required
SBIN	4	OUTPUT	1 Return Code
SBIN	4	OUTPUT	2 Reason Code
SBIN	4	INPUT	3 General Data Buffer Pointer

*Figure 28. DMS5XXTP TEMPLATE File*

DMS5XXTP TEMPLATE contains templates for three required parameters:

1. The **return code** from DMS5XX. Your routine must return one of the following values:

Value	Meaning
0	Successful. The action the file pool server takes upon return from the exit depends on how your routine sets the out flags in the general data buffer. See <a href="#">“Setting the Out Flags”</a> on page 211.
4	Function not supported by the exit. The file pool server takes its default action.
5	Function not supported by the exit. Further calls to the exit for this file pool function are suppressed. The file pool server takes its default action.

2. The **reason code** from DMS5XX.

**Note:** The reason code from this routine is not used by the file pool server. If you specified the ETRACE startup parameter when the server was initialized, the reason code is placed in the trace record the file pool server creates upon return from the exit.

3. A **pointer to the general data buffer**, in which the file pool server passes certain information (based on the type of exit) to DMS5XX. The buffer has fixed and variable size components.

## Setting the Out Flags

Each list entry in the general data buffer contains an 8-bit field called EXIOUTFL (Out Flags). Your DMS5XX routine can set these flags to indicate which user or users should be rolled back. When the return code from the routine is 0, the file pool server uses these flags to determine the actions it should take. Table 21 on page 211 shows the settings you can make. This field is the *only* area in the buffer your routine is permitted to modify.

Table 21. General Data Buffer Out Flags Settings

Bit	Settings
0	<p>Set this bit to 1 if you want the file pool server to roll back this connection. The exit routine can mark any active users for rollback except those the file pool server has marked not eligible in the EXIINFL (In Flags) field. If the exit does not mark any user for rollback, the server implements its current policy for the exit type:</p> <ul style="list-style-type: none"> <li>• File space usage exit: The server continues usual processing.</li> <li>• User storage group full exit: The server rolls back the logical unit of work that triggered the storage group full condition.</li> </ul>
1-7	Reserved for IBM use.

## Special Considerations for Writing Your Own Exit Routine

When writing your own storage use exit routine, keep in mind the following special considerations:

- Your exit routine must not request access to files or other file pool information for any file pool. Your exit routine can request access to CMS minidisk files.
- Your exit routine cannot request APPC/VM or IUCV services.
- While your exit routine is executing, the file pool server cannot perform any other tasks. Therefore, your exit routine design should consider execution time performance.
- Use of the interval timer in server exits could affect the integrity of the file pool server.
- Disabling or enabling of interrupts in file pool server exits is not recommended because it may produce unpredictable results when the file pool server runs in a CMS multiuser environment.
- The file pool server will enter the file space usage exit with all interrupts enabled. Interrupts will be disabled on entry to the user storage group full exit.
- Do not modify any of the EXITBUFF data except for the EXIOUTFL (Out Flags) field in each entry.

## Making the New or Modified Exit Routine Available

If you modify or replace DMS5XX ASSEMBLE, you must arrange to have your version of DMS5XX (DMSSFSEX) called instead of the IBM-supplied module in VMLIB. You can do this by putting your DMS5XX (DMSSFSEX) in your own callable services library:

1. Assemble your DMS5XX ASSEMBLE to build a new DMS5XX TEXT.
2. Create a CSL control file for your library. Although you can use any file ID for the control file, you might want to adopt the convention of using the name of your library and the file type CSLCNTRL. For example, if you plan to name your library MYLIB, name the control file MYLIB CSLCNTRL.

## Activating Storage Use Exits

In the control file, create a ROUTINE record that identifies the CSL routine name (the name to be called by the file pool server; in this case, DMSSFSEX), the TEXT file name, and the CSL template file name. For example, your ROUTINE record might look like this:

```
ROUTINE DMSSFSEX DMS5XX DMS5XXTP
```

The complete format of a ROUTINE record is described in [z/VM: CMS Application Development Guide for Assembler](#).

3. Use the CSLGEN command to build your callable service library. You can store your library on a minidisk or in a saved segment. For example, to build MYLIB on a minidisk accessed as file mode A, enter:

```
cslgen dasd mylib from mylib
```

For more information about the CSLGEN command, see [z/VM: CMS Commands and Utilities Reference](#).

4. Update the file pool server's PROFILE EXEC:
  - a. Add a command to access the minidisk on which your library resides. (This is not necessary if your library is in a saved segment.)
  - b. Add an RTNLOAD command to load your routine. For example, to load DMSSFSEX from the MYLIB library, add the command:

```
rtnload dmssfsex (from mylib system
```

You do not need to add a GLOBAL command because the library is specified in the RTNLOAD command. You do not need to add an RTNDROP command either; your own DMSSFSEX overrides the IBM-supplied version after the RTNLOAD command is processed. For more information about the RTNLOAD command, see [z/VM: CMS Commands and Utilities Reference](#).

- c. If applicable, release the minidisk on which your library resides.

## Chapter 10. Reorganizing the File Pool Repository Catalogs

This chapter is applicable only to repository file pool servers, not dedicated CRR recovery or FIFO servers.

To help improve the performance of file pool requests, server processing uses indexes on the information stored in the catalog storage group. The catalog storage group contains information about the files, directories, and authorizations that exist in the file pool. Because this information satisfies most file pool requests, it is critical for good performance that the server can get to the data quickly.

After much file pool activity, it is possible for the indexes to become fragmented and, therefore, less efficient. It is also possible for the blocks in which the catalog data resides to be used inefficiently as well. To ensure the optimum use of the catalog data and index space, you should periodically (every few months or so) check the catalogs for fragmentation as described in Step 1 below, and reorganize if you are experiencing a performance problem. (See *z/VM: Performance* for details.)

The FILESERV REORG command deletes unused file pool catalog entries for each user and reorganizes the file pool catalogs to ensure optimum use of catalog data and index space. To do this, it copies all the catalog data to an external file, clears out all the catalog data in the file pool, and then replaces the data from the copy stored in the external file.

The steps you should follow when considering catalog reorganization are:

1. Determine whether reorganization will help performance.

Through periodic monitoring of catalog space usage, you can determine the need for a reorganization of the file pool catalogs. Monitoring can be done by entering the following command from an administration machine:

```
QUERY FILEPOOL CATALOG filepoolid
```

The file pool must be available in multiple user mode. It displays information about the catalogs, similar to the following:

```
⋮
CATALOG SPACE INFORMATION
      4000 Data Blocks
      3000 Occupied Data Blocks
       75% Percent Occupied Data Blocks
      4000 Index Blocks
      3600 Occupied Index Blocks
       90% Percent Used Index Blocks
```

The display shows the number of data blocks (4000), which is the number of available 4096-byte blocks to hold information about files, directories, aliases, enrolled users, and similar information. Occupied Data Blocks shows the number of data blocks that actually have information on them (3000). This does not mean the blocks are full, it just means at least one of those 4096 bytes has useful information in it. The percent of occupied data blocks is calculated from the preceding two values.

The index blocks contain only index values and pointers to the data blocks. The displayed numbers for the index blocks have meanings equivalent to those for the data blocks.

If the ratio of occupied data blocks to occupied index blocks has decreased significantly, extensive index fragmentation has occurred. If the performance has degraded during this period also, reorganization may be beneficial.

2. Log on or reconnect to the file pool server machine as follows:

- If the file pool server is currently processing in multiple user mode, reconnect to it and enter a STOP operator command. (For more about stopping multiple user mode processing, see [“Stopping](#)

Server Processing” on page 65.) If you use file pool backup facilities, do not inhibit the backup that is automatically made at shutdown.

- Otherwise, if the server is not currently processing in multiple user mode, log on it.

If you set up the server machine for automatic starting, as instructed in Chapter 15, “Generating a File Pool and Server,” on page 247, you have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC issues a FILESERV START command, which is not desired at this time.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

Next, process the SETUP EXEC to access minidisks needed by the server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

If you forget to inhibit the execution of the PROFILE EXEC and the file pool server successfully starts, immediately stop the server by entering:

```
stop immediate
```

### 3. Back Up the Control Data

If you did not do so in Step “2” on page 213, back up the control data so the file pool can be restored if FILESERV REORG cannot be run to completion. Use the FILESERV BACKUP command (For more detail, see “[FILESERV BACKUP](#)” on page 500).

4. Update the CATBUFFERS startup parameter. Increasing the buffer size lets the catalog reorganization process run more efficiently. If a CATBUFFERS startup parameter is already specified, write it down before updating it. You will need to reset CATBUFFERS in a later step.

Update the CATBUFFERS startup parameter in the DMSPARMS file:

```
CATBUFFERS 5000
```

If the server machine does not have enough virtual storage available to it, you may need to reduce the CATBUFFERS value.

5. Enter a FILEDEF Command for ddname TEMP.

During its processing, the FILESERV REORG command writes data to the file associated with *ddname* TEMP and then later reads that data. The FILEDEF can identify a tape or DASD file. Most users identify a tape file. If you use a DASD file, the file can reside either on a minidisk or in another file pool. When using a minidisk file, ensure the minidisk contains enough free space to contain about half of the physical blocks currently allocated to the catalog storage group. When using a file pool, ensure the server machine is enrolled with enough space and ensure there is enough physical DASD in the storage group in which the server is enrolled.

These are example FILEDEFs for DASD and tape:

```
filedef temp disk filename filetype filemode  
filedef temp tap1 sl
```

**Note:** If a tape is used, do not specify the LABOFF operand on the FILEDEF command. Do not let the FILEDEF command default to LABOFF either.

6. Enter the FILESERV REORG command:

```
fileserv reorg
```

If the FILESERV REORG command fails to complete successfully, the action you need to take varies depending on when the failure occurred.

If the command failed after the DMS3701I message is written to the server console, FILESERV REORG processing must complete before the file pool can be used for any other processing. Message DMS3701I is:

```
File Pool catalogs unloaded to DDNAME=TEMP. Timestamp: n1
```

In this case, you must rerun the FILESERV REORG command and ensure the file identified by *ddname* TEMP is the same file created by the FILESERV REORG processing that failed. The file identified by *ddname* TEMP must be read as input to re-build storage group 1.

If FILESERV REORG fails before message DMS3701I is displayed, a rerun is not required before the file pool can be used for other purposes. You can rerun FILESERV REORG if you wish.

If FILESERV REORG fails and cannot be rerun, you can recover by restoring the control data by issuing the FILESERV START command with RESTORE specified in the DMSPARMS file.

7. Reset the CATBUFFERS startup parameter.

If your DMSPARMS file originally contained a CATBUFFERS startup parameter, reset it to its old value. If it did not, delete the CATBUFFERS startup parameter.

8. If the BACKUP startup parameter is in effect, enter a FILESERV BACKUP command to back up the control data. (The FILESERV REORG command makes your current control data backup obsolete.)

9. Enter the FILESERV START command to resume multiple user mode processing:

```
fileserv start
```

**Catalog Reorganization Tip:** Because the catalogs should be reorganized periodically, you can save some time by creating a DMSPARMS file that contains your usual startup parameters as well as the CATBUFFERS startup parameter. When it is time to reorganize the catalogs, rename the current DMSPARMS file to some temporary name. Then rename the alternate parameter file to *serverid* DMSPARMS. After running the FILESERV REORG command, you would rename the files back to their original names. (You might code an exec to do this.)





## Chapter 11. Regenerating a Repository File Pool

This chapter is applicable only to repository file pool servers, not dedicated CRR recovery or FIFO servers. Regenerating a file pool can be done to a dedicated CRR or FIFO file pool, but it is not applicable because they would never need to increase the:

- MAXDISKS value greater than the recommended 2.
- MAXUSERS value greater than the recommended 5.
- Size of the control minidisk, because the dedicated CRR recovery or FIFO server never adds records to the catalogs and never writes any file blocks. If you need to generate a CRR recovery or FIFO server again, see [“Restoring a File Pool by Generating It Again” on page 127](#).

During the life of a file pool, you may need to expand the limits established for it during FILESERV GENERATE processing, namely:

- Control minidisk size
- Maximum number of minidisks in the file pool (MAXDISKS)
- Estimated maximum number of enrolled users who will be individually authorized to create objects (such as base files, directories, and aliases) in a file pool (MAXUSERS).

You can determine the current MAXDISKS and MAXUSERS settings by looking in the POOLDEF file. To expand any of the above limits, you must *regenerate* the file pool. Regeneration involves:

- Taking a backup of the control data
- Replacing the control minidisk with a larger one
- Entering the FILESERV REGENERATE command
- Taking another backup

Detailed instructions follow.

Regeneration is not something that can be done hurriedly. It makes extensive changes to the control data, and, done carelessly, could destroy the file pool. Please read all the instructions before attempting to regenerate the file pool:

1. Back up the control data.

If you are using file pool backup facilities, back up the control data while no users are accessing the file pool. You can either do the backup at shutdown, or by issuing the FILESERV BACKUP command.

Do not skip this step. If something goes wrong during regeneration, you need a current backup to repair the file pool.

2. Log off the server machine. (It is necessary to log off the server machine because the z/VM system directory entry is changed in the next step.)
3. Define another larger file pool control minidisk.

**Note:** This is a new minidisk definition and not an extension of the existing control minidisk. Do not delete the present MDISK entry until the FILESERV REGENERATE command has finished.

The FILESERV REGENERATE command requires as input the virtual device number of a new file pool control minidisk. The minidisk must be larger than the old minidisk. Use 25% as a rule of thumb for the amount by which you should increase the size of the control minidisk. A 25% increase allows for nominal increases to MAXUSERS and MAXDISKS as well as the potential addressable file pool space.

If the control minidisk is too small, FILESERV REGENERATE fails with an error message:

```
DMS3913E File pool CONTROL disk is incorrect size. Reason = 3
```

If this happens, increase the size by another 5% to 10% and try it again.

## Regenerating a Repository File Pool

If you are anticipating rapid file pool growth, you might want to be more generous with the increase in the size of the control minidisk. The goal is to avoid frequent regenerations. See [“Step 3: Determine Initial DASD Allocations” on page 250](#) for more information on determining the size of the control minidisk.

Here is an example MDISK statement for a file pool control minidisk:

```
MDISK 401 3380 cyl 17 XXXXXX R FPOOLRPW FPOOLLWPW
MINIOPT NOMDC
```

Do not add a LINK control statement to the z/VM system directory for the minidisk. Server machines link to the file pool minidisks when FILESERV START is entered.

Set the volume serial numbers (**XXXXXX**) as appropriate for your DASD volumes.

It is strongly recommended you specify R for the access mode, as shown above. Specify your own read and write passwords; FPOOLRPW and FPOOLLWPW are examples.

The previous example assumes a count-key-data device, which means you must supply a cylinder relocation factor (*cyl*). For an FB-512 device, specify the appropriate block relocation factor (*blkr*) and number of blocks (*blks*). See [z/VM: CP Planning and Administration](#) for more about the MDISK statement.

#### 4. Log on the server machine.

If you set up the server machine for automatic starting, as instructed in [Chapter 15, “Generating a File Pool and Server,” on page 247](#), you have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC calls a FILESERV START command, which is not desired at this time.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

Next, process the SETUP EXEC to access minidisks needed by the server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

If you forget to inhibit the execution of the PROFILE EXEC and the server successfully starts, stop the server by using the STOP IMMEDIATE operator command:

```
stop immediate
```

#### 5. Use the CMS QUERY ACCESSED command to determine a mode letter not currently in use (for example, if mode C is not in use, note it for use below).

#### 6. Enter the following CP command to make the minidisk write-accessible to the server machine:

```
link * vdev1 vdev2 w
```

where:

**\***

means the minidisk is defined in the z/VM system directory entry for this virtual machine.

**vdev1 vdev2**

specify the virtual device numbers used on the MDISK statement.

**w**

indicates write mode access is required for this minidisk.

#### 7. Enter the following CMS command to initialize and label the minidisk:

```
format vdev access-letter (blksize 512
```

where:

**vdev**

is the virtual device number you used on the CP LINK command above.

**access-letter**

is the mode letter you obtained in step “5” on page 218.

**blksize 512**

a block size of 512 must be specified for the control minidisk.

The FORMAT command prompts you for the minidisk volume label (disk label). You can enter any valid volume label value.<sup>4</sup>

The FORMAT command prompts you whether to `erase all files` on the minidisk. Reply 1 for YES to this message (after ensuring you have specified the correct minidisk).

8. Enter the following CMS commands to allocate the minidisk as a block I/O file:

```
reserve filename filetype filemode
```

For:

**filename**

any valid file name can be specified. For naming consistency, it is recommended you use the file pool ID for the file name.

**filetype**

although anything can be specified for the file type, it is recommended you specify NCONTROL.

**filemode**

specify the access-letter you entered above for the CMS FORMAT command.

The RESERVE command prompts you whether to `erase all files` on the minidisk. Reply 1 for YES to this message (after ensuring you have specified a file mode letter that matches the CMS FORMAT command).

9. Enter the following CMS command to release and detach the minidisk:

```
release vdev (det
```

For `vdev`, specify the virtual device number used on the LINK and FORMAT commands.

10. Increase the CATBUFFERS startup parameter (only when you intend to increase the MAXUSERS value). Increasing the buffer size allows the catalog reorganization process to run more efficiently. (FILESERV REGENERATE processing automatically reorganizes the catalogs when the MAXUSERS value is increased.)

If a CATBUFFERS startup parameter is already specified, write down the old value before increasing it. You need to reset CATBUFFERS in a later step.

Add the following CATBUFFERS startup parameter to the DMSPARMS file:

```
CATBUFFERS 5000
```

If the server machine does not have enough virtual storage available to it, you may need to reduce these CATBUFFERS values.

11. Enter a FILEDEF command for a temporary file (only when MAXUSERS is specified and is greater than the current value).

If you want to increase the estimated maximum number of enrolled users who will be individually authorized to create objects (such as base files, directories, and aliases) in the file pool (MAXUSERS), you need to enter a FILEDEF command for `ddname TEMP`. The FILEDEF can identify a tape or DASD file. Most users identify a tape file. If you use a DASD file, the file can reside either on a minidisk

<sup>4</sup> FILESERV commands that call FORMAT commands internally use the letter "T" followed by the minidisk's virtual device number for the minidisk volume label.



13. Reset the CATBUFFERS startup parameter.

If your DMSPARMS file originally contained a CATBUFFERS startup parameter, reset it to its old value. If it did not, delete the CATBUFFERS startup parameter.

14. Back up the control data.

Because regeneration changed the control data, the backup taken on step “1” on page 217 is now obsolete. To protect your data from potential loss, you should backup the control data again (using the FILESERV BACKUP command).

15. Enter FILESERV START to resume usual operations.

Having successfully completed regeneration, you can now reclaim the DASD space occupied by the old file pool control minidisk. You can remove the MDISK statement for the old control minidisk from the z/VM system directory of the file pool server machine and use the space for other purposes.

## What to Do If Something Goes Wrong

---

If the FILESERV REGENERATE command does not complete successfully, you must rerun it before doing any other processing on the file pool.

Usually you should rerun FILESERV REGENERATE using the same parameters. An exception to this is when the error was caused by a control minidisk error. In that case, you must redefine, format, and reserve another control minidisk prior to rerunning the command.

When you increase the MAXUSERS value, the file pool catalogs are also reorganized during FILESERV REGENERATE processing. When catalog reorganization begins, it writes the following message:

```
DMS3701I File Pool catalogs unloaded to DDNAME=TEMP. Timestamp: n1
```

If FILESERV REGENERATE fails after you see that message, ensure the file identified by ddname TEMP is the same file created by the processing that failed. When you rerun FILESERV REGENERATE, that file is read, and the data within it is used to rebuild storage group 1. Without that file, it is not possible for FILESERV REGENERATE to rebuild storage group 1, in which case your only alternative is to restore the control data from the backup made in step “1” on page 217.

If you make a mistake you feel has damaged the file pool beyond repair, remember you can **always** restore the control data from the backup made in step “1” on page 217.



## Chapter 12. Accounting

This chapter describes:

- How to start the accounting facility
- How to stop the accounting facility
- Accounting record formats
- How to customize user accounting records

This chapter applies to all file pool servers because all types of servers use the accounting facility.

### Starting the Accounting Facility

In this chapter the word *server* is used. For a complete description of the possible combinations of file pool servers, see Chapter 1, “File Pool Administration Overview,” on page 3.

To have a server contribute accounting records to the z/VM accounting system service, do the following:

1. If you are not familiar with z/VM accounting, review the description in [z/VM: CP Planning and Administration](#).
2. Ensure the ACCT operand is specified on the OPTION control statement in the z/VM system directory entry for the server machine. Add the ACCT operand if necessary. The ACCT operand is needed because servers call DIAGNOSE code X'4C' to generate accounting records. For example, suppose the OPTION control statement is currently:

```
OPTION MAXCONN 2000 APPLMON
```

You should update that statement as follows:

```
OPTION MAXCONN 2000 APPLMON ACCT
```

Use your usual procedures for updating the z/VM system directory. See [z/VM: CP Planning and Administration](#) for more information about the ACCT operand of the OPTION control statement.

**Note:** Keep in mind the records generated by server processing will increase the total number of accounting records written to the z/VM accounting system service.

3. If you already have routines to process other accounting records, you may want to modify those routines to handle the file pool server records.
4. If the server is currently processing in multiple user mode, reconnect to it and enter a STOP operator command. For more about stopping multiple user mode processing, see “Stopping Server Processing” on page 65. Then enter LOGOFF. (It is necessary to log off the server and log back on again in the next step to put the z/VM system directory entry changes into effect.)
5. Log on the server machine.

If you set up the server machine for automatic starting, as instructed in Chapter 15, “Generating a File Pool and Server,” on page 247, you have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC issues a FILESERV START command, which is not desired at this time.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

Next process the SETUP EXEC to access minidisks needed by the server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

## Stopping the Accounting Facility

If you forget to inhibit the execution of the PROFILE EXEC and the server successfully starts, immediately stop the server by entering:

```
stop immediate
```

6. Edit the DMSPARMS file. Add the ACCOUNT startup parameter to the file:

```
ACCOUNT
```

If the NOACCOUNT startup parameter is specified in the file, delete it. For more about the startup parameter, see [ACCOUNT](#).

7. Start the server for usual multiple user mode access. Enter:

```
fileserv start
```

When the server is started, it will automatically begin generating accounting records.

**Note:** The ACCOUNT startup parameter is ignored in dedicated maintenance mode. Accounting records are not generated during dedicated maintenance mode processing.

For a description of the accounting record formats and at what processing points they are generated, see [“Accounting Record Formats”](#) on page 224.

## Stopping the Accounting Facility

---

To stop a server from generating accounting records:

1. Log on or reconnect to the server machine as follows:

- If the server is currently processing in multiple user mode, reconnect to it and enter a STOP operator command. (For more about stopping multiple user mode processing, see [“Stopping Server Processing”](#) on page 65.)
- Otherwise, if the server is not currently processing in multiple user mode, log on it and enter a STOP NOBACKUP operator command.

2. Edit the DMSPARMS file. Delete the ACCOUNT startup parameter. You do not need to specify NOACCOUNT—it is the default.

If you later want to resume the generation of accounting records, simply add the ACCOUNT startup parameter to the DMSPARMS file.

For more about the startup parameters, see [ACCOUNT](#) and [NOACCOUNT](#).

3. Enter:

```
fileserv start
```

to start multiple user mode processing.

## Accounting Record Formats

---

### PI

Servers generate the following types of accounting records:

#### Initialization

A server writes an accounting record at the end of its own initialization. This record describes the resources consumed during the initialization process.

#### Checkpoint

After doing a checkpoint, server processing writes an accounting record for the checkpoint process. A *checkpoint* is an internal server operation during which the changes recorded on the file pool log minidisks are permanently made to the file pool. The server takes checkpoints after a certain number of file pool log minidisk blocks have been written.



**Note:** Accounting records are generated only for file pool log checkpoints. This does not apply to CRR logs.

### Operation

During termination, servers write an accounting record for processing done on behalf of the server operator commands since multiple user mode was started.

### System

Servers perform internal housekeeping tasks that cannot be attributed to a single user. The server automatically starts these *subtasks* as needed. At the completion of each subtask, the server writes a *system* accounting record summarizing the resources consumed in processing the subtask.

### Termination

Servers write an accounting record summarizing the server resources consumed during the entire multiple user mode execution.

### User

Servers write an accounting record for each user summarizing the processing done on the user's behalf.

These fixed-length, 80-byte records are placed in the z/VM system accounting file.

Servers write accounting records at the end of every server session. A *session*, in this discussion, is the length of time a user ID remains connected to a file pool or a server. The session begins when the first request is called to the server machine and an APPC/VM connection is made between the user's virtual machine and the server. The session ends when one of the following happens:

- The user machine's CMS session ends.
- The user machine is disconnected from the server by means of the server FORCE operator command.
- The server ends usually or unusually.
- The last communication path from a user machine is severed by processing that the user machine itself begins.
- An APPC/VM communication that a server begins to a user machine fails and there are no other communication paths to the user machine available.

Server processing also writes accounting records in response to the Write Accounting CSL routine (DMSWRACC). (See the description of the DMSWRACC routine in *z/VM: CMS Callable Services Reference*.) When this request is made to a server, it will write accounting records for all users currently connected to it.

**Note:** A server collects accounting data only for the processing it does on behalf of users or its own internal processes, not for processing done within other machines, such as administration machines or user machines.

Servers do not write accounting records during dedicated maintenance mode processing. If the ACCOUNT startup parameter is specified, it is ignored.

The accounting records do not contain information about the number of committed 4KB DASD blocks users hold. To determine the number of committed blocks, use [“QUERY LIMITS”](#) on page 655.

Following are the formats of the accounting records.

## Initialization Record

The format of the initialization accounting record is shown in [Figure 29](#) on page 225.

```

Displacements:
0           8           16           32           44  48  52           74  78
|           |           |           |           |   |   |           |   |
SFSM      SFS_INIT      051387182005           ISFSC0

```

Figure 29. Initialization Accounting Record Format

## Accounting Record Formats

- 0**  
z/VM User ID of the server machine (fixed by CP)
- 8**  
"SFS\_INIT"
- 16**  
Reserved
- 32**  
Date and time of the accounting record (MMDDYYHHMMSS)
- 44**  
A binary fullword containing the duration of the server initialization process (in seconds)
- 48**  
A binary fullword containing the processor time used by the file pool initialization process (in milliseconds)
- 52**  
Reserved
- 74**  
"xSFS" identifier to separate file pool server accounting records from other z/VM accounting records (x = I for Initialization)
- 78**  
Record identifier (Character 'C0') fixed by CP

## Operator and Checkpoint Records

The format of the operator/checkpoint accounting record is shown in [Figure 30](#) on page 226.

Displacements:								
0	8	16	32	44	48	52	74	78
SFSM	SFS_SYS		051387182005				CSFSC0	

*Figure 30. Operator/Checkpoint Accounting Record Format*

- 0**  
z/VM User ID of the server machine (fixed by CP)
- 8**  
"SFS\_SYS "
- 16**  
Reserved
- 32**  
Date and time of the accounting record (MMDDYYHHMMSS)
- 44**  
Reserved
- 48**  
A binary fullword containing the processor time used (in milliseconds)
- 52**  
Reserved
- 74**  
"xSFS" identifier to separate file pool server accounting records from other z/VM accounting records (x = C for Checkpoint or O for Operator)
- 78**  
Record identifier (Character 'C0') fixed by CP

## System Record

The format of the system accounting record for server subtasks is shown in [Figure 31 on page 227](#).

```

Displacements:
0           8           16           32           44  48  52  56  60  64  68           74  78
|           |           |           |           |  |  |  |  |  |           |  |
SFSM      SFS_SYS      |           051387182005 |  |  |  |  |  |           |  |
  
```

*Figure 31. System Accounting Record Format*

- 0**  
z/VM User ID of the server machine (fixed by CP)
  - 8**  
"SFS\_SYS "
  - 16**  
Reserved
  - 32**  
Date and time of the accounting record (MMDDYYHHMMSS)
- The following totals reflect values accumulated for a subtask:
- 44**  
A binary fullword containing the active time (that is, time the server was processing for the subtask) in seconds.
  - 48**  
A binary fullword containing the processor time used (in milliseconds).
  - 52**  
A binary fullword containing the number of physical I/O requests made by the server on behalf of the subtask.
  - 56**  
Reserved.
  - 60**  
A binary fullword containing the number of 4KB file blocks read by the subtask. Blocks read from the log, control, or catalog minidisks are not included in this count.
  - 64**  
A binary fullword containing the number of 4KB file blocks written by the subtask. Blocks written to the log, control, or catalog minidisks are not included in this count.
  - 68**  
Reserved.
  - 74**  
"xSFS" identifier to separate file pool server accounting records from other z/VM accounting records (x = S for System).
  - 78**  
Record identifier (Character 'C0') fixed by CP.

## Termination Record

The format of the termination accounting record is shown in [Figure 32 on page 228](#).

```

Displacements:
0           8           16           32           44  48  52           74  78
|           |           |           |           |   |   |           |   |
SFSM      SFS_TERM      051387182005           TSFSC0

```

Figure 32. Termination Accounting Record Format

- 0** z/VM User ID of the server machine (fixed by CP)
- 8** "SFS\_TERM"
- 16** Reserved
- 32** Date and time of the accounting record (MMDDYYHHMMSS)
- 44** A binary fullword containing the time, in seconds, from server startup to server shutdown
- 48** A binary fullword containing the total number of DASD I/Os during the entire execution of the file pool server (extracted from the data collected for the CP monitor file)
- 52** Reserved
- 74** "xSFS" identifier to separate file pool server accounting records from other z/VM accounting records (x = T for Termination)
- 78** Record identifier (Character 'C0') fixed by CP

## User Record

The format of the user accounting record is shown in [Figure 33 on page 228](#).

```

Displacements:
0           8           16           32           44  48  52  56  60  64  68           74  78
|           |           |           |           |   |   |   |   |   |           |   |
SFSM      JESSICA A101TEST      051387182005           USFSC0

```

Figure 33. User Accounting Record Format

- 0** z/VM User ID of the server machine (fixed by CP).
- 8** z/VM User ID of the user machine accessing SFS and CRR.
- 16** Installation-supplied data (such as account numbers). You can code a program that supplies information to this field. For more information, see [“Customizing User Accounting Records” on page 229](#). If you have not coded such a program, the field contains character blanks.
- 32** Date and time of the accounting record (MMDDYYHHMMSS).

The following totals reflect values accumulated for a user:

- 44** A binary fullword containing the active time (that is, time the server was processing requests for the user) in seconds.
- 48** A binary fullword containing the processor time used (in milliseconds).
- 52** A binary fullword containing the number of physical I/O requests made by the server on behalf of the user.
- 56** Reserved.
- 60** A binary fullword containing the number of 4KB file blocks read by the user. Blocks read from the log, control, or catalog minidisks are not included in this count.
- 64** A binary fullword containing the number of 4KB file blocks written by the user. Blocks written to the log, control, or catalog minidisks are not included in this count.
- 68** Reserved.
- 74** "xSFS" identifier to separate file pool server accounting records from other z/VM accounting records (x = U for User).
- 78** Record identifier (Character 'C0') fixed by CP.

## Customizing User Accounting Records

---

File pool server user accounting records contain a 16-byte area that your installation can use for account numbers or other information. To provide installation-defined accounting data for all primary user IDs on the system (the user IDs of the virtual machines doing the work), you must replace the DMS2AB CSL routine, supplied by IBM in the VMLIB callable services library. DMS2AB is an installation-wide exit.

CMS in the user machine calls the DMS2AB exit routine during the first request that causes communications with a file pool server. For example, the request might be an SFS command or an SFS CSL call. DMS2AB is called regardless of whether the ACCOUNT startup parameter is specified for the server with which the user machine is establishing communications.

CMS passes a 16-byte area of storage to DMS2AB. Initially, the area contains character blanks. Your version of DMS2AB can store accounting data, such as an account number, into that area. CMS does no error checking on the data. In subsequent communications, CMS passes whatever data is in the area to the server for inclusion in the accounting records for that user. If you do not supply your own version of DMS2AB, the 16-byte area remains set to blanks, and blanks appear in the user accounting records.

## Writing Your Own Accounting Exit Routine

Source code for the DMS2AB module is in file DMS2AB ASSEMBLE. For general information about writing a CSL routine, see *z/VM: CMS Application Development Guide for Assembler*. For information about the CSLENTRY, CSLGETP, and CSLEXIT macros used in writing a CSL routine, see *z/VM: CMS Macros and Functions Reference*.

Make sure your CSL routine returns the necessary information to CMS. Your DMS2AB must not perform any operations that would begin communications with a file pool server. The IBM-supplied CSL template file for the DMS2AB routine is DMS2AB TEMPLATE, shown in Figure 34 on page 230. This file identifies the routine's output parameters; there are no input parameters. The general format of a CSL template file is described in *z/VM: CMS Application Development Guide for Assembler*.

```
2 2
SBIN 4 OUTPUT
FCHR 16 OUTPUT
```

Figure 34. DMS2AB TEMPLATE File

DMS2AB TEMPLATE contains templates for two required parameters:

1. The **return code** from DMS2AB. A return code of 0 indicates DMS2AB has completed its function successfully.

Any positive return code from DMS2AB causes CMS to return an error return code from the command or program function that attempted communications with a file pool server. This lets you reject a user's attempt to communicate with a file pool server when the user has improper accounting information.

CSL processing uses negative return codes for errors. Therefore, your DMS2AB routine should not set negative return codes. If a negative return code is encountered after the call to the exit, a return code or message describing a CSL error results.

2. A 16-byte storage area for the **accounting data** to be included in the user accounting records.

## Making Your Accounting Exit Routine Available to CMS

You must arrange to have your version of DMS2AB called instead of the IBM-supplied module in VMLIB. You can do this by putting your DMS2AB in your own callable services library as described below:

1. Assemble your DMS2AB ASSEMBLE to build a new DMS2AB TEXT.
2. Create a CSL control file for your library. Although you can use any file ID for the control file, you might want to adopt the convention of using the name of your library and the file type CSLCNTRL. For example, if you plan to name your library MYLIB, name the control file MYLIB CSLCNTRL.

In the control file, create a ROUTINE record that identifies the CSL routine name (the name to be called by the file pool server), the TEXT file name, and the CSL template file name. For example, your ROUTINE record might look like this:

```
ROUTINE DMS2AB DMS2AB DMS2AB
```

The complete format of a ROUTINE record is described in [z/VM: CMS Application Development Guide for Assembler](#).

3. Use the CSLGEN command to build your callable service library. You can store your library on a minidisk or in a saved segment. **Do not store your library in a file pool.**

For example, to build MYLIB on a minidisk accessed as file mode A, enter:

```
cslgen dasd mylib from mylib
```

For more information about the CSLGEN command, see [z/VM: CMS Commands and Utilities Reference](#).

4. Update the system profile (SYSPROF EXEC):
  - a. Add a command to access the minidisk on which your library resides. (This is not necessary if your library is in a saved segment.)
  - b. Following the RTNLOAD command for VMLIB, add an RTNLOAD command to load your routine. For example, to load DMS2AB from the MYLIB library, add the command:

```
rtnload dms2ab (from mylib system
```

You do not need to add a GLOBAL command because the library is specified in the RTNLOAD command. You do not need to add an RTNDROP command either; your own DMS2AB overrides the IBM-supplied version when the RTNLOAD command is processed. For more information about the RTNLOAD command, see [z/VM: CMS Commands and Utilities Reference](#).

c. If applicable, release the minidisk on which your library resides.

When a user IPLs CMS, the system profile makes your version of DMS2AB available.

## Multiple User ID Support Considerations

To supply user accounting data for an alternate user ID (for example, if a service machine is doing work on behalf of another user ID), you can use the *acctdata* and *userid* parameters of the DMSGETWU (SFS Get Work Unit ID) CSL routine. See [z/VM: CMS Application Development Guide](#) for information on SFS Multiple User ID support.

The *userid* parameter specifies the user ID that will appear in the user accounting records generated for all file pool work for the work unit done on behalf of that user ID. It is stored starting at byte 8 of the user record shown in [Figure 33 on page 228](#).

The *acctdata* parameter associates accounting data with the user ID specified in the *userid* parameter. If you specify the *userid* parameter but not the *acctdata* parameter, the 16-byte installation-supplied data area (starting at byte 16) in the user accounting records will contain blanks.

See [z/VM: CMS Callable Services Reference](#) and [z/VM: CMS Application Development Guide](#) for more information about DMSGETWU.

**PI end**





## Chapter 13. Setting Up a File Pool for Remote Use

This chapter contains topics related to the remote use of file pools. It assumes you have some familiarity with the connectivity concepts described in *z/VM: Connectivity*. It also assumes a connection is set up at your installation to the remote system. You can set up a connection using the Transparent Services Access Facility (TSAF), the Inter System Facility for Communication (ISFC), or the APPC/VM VTAM Support (AVS). In the topics that follow, the term *local* represents your processor, file pool, or users of that processor. *Remote*, on the other hand, identifies any processor that is not your local processor, any file pool that is not on your local processor, or any user ID defined on a processor other than your local processor.

This chapter applies mainly to repository file pools. FIFO file pool servers can be accessed by remote users if the FIFO objects are defined in remote file pools and the remote file pools designates remote FIFO servers with the startup parameter. CRR file pools also apply, but have very little use here, except when you want to set up remote administration of a CRR recovery server by a user machine that has administration authority. For more information, see [“Set Up Remote User Machine Administration of CRR Recovery Server”](#) on page 326.

The topic on "Enrolling Remote Users" has no relevance to CRR because with CRR you have a minimal size file pool not intended for users to use repository functions. This assumes you have separate servers for file pool and CRR functions, which is strongly recommended.

The chapter contains the following topics:

- Making your file pool available to other processors (applicable to all file pool servers)
- Restricting access from other processors (applicable to all file pool servers)
- Enrolling remote users (applicable only to repository file pool servers, not dedicated CRR recovery or FIFO servers)
- Using nicknames and local IDs for remote users (applicable to all file pool servers)

See the section on planning for OpenExtensions in *z/VM: CP Planning and Administration* for information on establishing permission to use BFS file spaces. Also see [Chapter 8, “Security,”](#) on page 137 for information about the authorization mechanisms supplied with file pool servers.

### Making Your File Pool Available to Other Processors

If you use ISFC links to connect VM systems, those systems are members of a CS collection. If your processor is a member of a TSAF or CS collection, you can make your file pool available to other processors within the collection. If AVS is used, you can make your file pool available to other z/VM processors within the Systems Network Architecture (SNA) network. To make your file pool available to remote processors, do the following:

1. Determine if the file pool is eligible.

Any file pool that has a file pool ID beginning with the characters "VMSYS" cannot be made available to other processors. Such file pools are restricted to access from the local processor.

If you have a file pool that has an ID beginning with "VMSYS", you have to change the file pool ID before you can make the file pool available to other processors. See [“Changing the File Pool ID”](#) on page 67 for instructions on changing a file pool ID.

The IBM-supplied file pool VMSYS should not be renamed and made available to other processors. Not only is the file pool ID required for various internal functions, it contains licensed materials of IBM. If you make the file pool available to other processors, you may be violating the z/VM licensing agreement and the agreement of any other IBM or non-IBM licensed program installed in that file pool.

The IBM-supplied file pools VMSYSU and VMSYSR may be made available to other processors after their file pool IDs are changed.

2. Review your TSAF, ISFC, or AVS configuration.

For a remote request to get to a server, a TSAF, ISFC, or AVS connection must exist to allow communications from the remote processors to the server. For instructions on setting up and using TSAF, ISFC, and AVS connections, see *z/VM: Connectivity*. That document contains general connectivity information as well as examples of using SFS in those environments.

When reading that manual, it is useful to keep in mind the following information:

- If a TSAF or CS collection is connected to a SNA network through AVS, the processors in the SNA network that are not within the collection are considered *outside* the TSAF or CS collection.
- Advanced Program-to-Program Communications/VM (APPC/VM) is used to communicate between the server and the user machines. Because APPC/VM is used, TSAF, ISFC, and AVS can be configured such that any user machine can communicate with any server in the local TSAF or CS collection, or any server on any processor in the SNA network.
- In SNA terminology, a server is considered a *transaction program*.
- A CMS communications directory entry can be used to refer to a file pool server located on the same system or on a different system in the same TSAF or CS collection. A CMS communications directory entry is required if the file pool server is located on a system in a different TSAF or CS collection. See *z/VM: Connectivity* for a complete description of CMS communication directories.
- It is recommended the system-level CMS communications directory entries for file pools on your system, file pools on different systems within your TSAF or CS collection, and file pools outside your TSAF or CS collection indicate:

```
:security.same
```

To successfully access file pools outside the TSAF or CS collection with `:security.same`, the following is necessary:

- The local and remote collection must support the *already verified* security level across the session between them. The session is selected by the specification of the locally known LU name (:luname tag) and mode name (:modename tag). Session characteristics are determined by VTAM LU definitions; sessions are activated by AVS gateway (AGW) commands.
- The AVS in the remote collection must identify the user IDs in the local collection that are considered to be already verified and provide mapping as necessary to eliminate user ID duplication.

If these conditions cannot be satisfied for file pools located outside the TSAF or CS collection, then for those file pools indicate

```
:security.pgm.
```

**Note:** VM/SP release 6 AVS does not support *already verified* security level. To access file pools in a system running VM/SP release 6 outside your TSAF or CS collection, specify

```
:security.pgm.
```

No matter what access security is identified for a file pool server machine, the server will be presented with a locally validated user ID that uniquely identifies the requester. For more information on access security information, see *z/VM: Connectivity*.

- In a CMS communications directory entry for a file pool, the `:tpn` tag should indicate the file pool ID.

**Note:** The file pool ID of the file pool should not be specified with a trailing colon (:).

### 3. Re-evaluate the enrollment of PUBLIC.

If you have enrolled PUBLIC in the file pool, you should re-evaluate whether you still want PUBLIC to be enrolled when the file pool is made available to other processors.

When the file pool is not available to other processors, the set of users that can connect to the file pool by virtue of the enrollment of PUBLIC is limited to the set of users defined on the processor. For many installations, this does not compromise the security of the file pool. Because the users may be

comfortable with the scope of security in this situation, they may have granted various read and write authorizations to PUBLIC.

When you make the file pool available to other processors, however, the set of users that can exercise the authorizations granted to PUBLIC is greatly increased. If you have sensitive information in the file pool, you may want to delete PUBLIC from the file pool (see [“Deleting Everyone Enrolled by ENROLL PUBLIC”](#) on page 94). This would restrict access to only those who have been explicitly enrolled. While this would exclude some users who previously had access, it might be most prudent for your security. During this period of restricted access, you might ask the file pool users to re-evaluate any authorities they have granted to PUBLIC. After the users have secured any sensitive information, you can re-enroll PUBLIC.

4. Log on or reconnect to the server machine as follows:

- If the server is currently processing in multiple user mode, reconnect to it and enter a STOP operator command. (For more about stopping multiple user mode processing, see [“Stopping Server Processing”](#) on page 65.)
- Otherwise, if the server is not currently processing in multiple user mode, log on the server.

If you set up the server machine for automatic starting, as instructed in Chapter 15, [“Generating a File Pool and Server,”](#) on page 247, you have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC issues a FILESERV START command, which is not desired at this time.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

Next, process the SETUP EXEC to access minidisks needed by the server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

If you forget to inhibit the execution of the PROFILE EXEC and the server successfully starts, stop the server by using any variation of the STOP operator command:

```
stop
```

Enter STOP NOBACKUP if the BACKUP startup parameter is in effect and you do not want to back up the control data at this time.

5. Specify the REMOTE or SSI startup parameter in the DMSPARMS file. Edit the DMSPARMS file, and add a record containing the appropriate startup parameter:

```
REMOTE
```

or

```
SSI
```

If the LOCAL startup parameter is specified in the DMSPARMS file, delete it.

The REMOTE startup parameter causes the server to accept requests from remote processors. Server processing does not distinguish between processors in your TSAF or CS collection and processors outside of your TSAF or CS collection. When the server receives a request for data, it merely checks to see whether the request is from the local processor (the processor on which it is running), or a remote processor. If the REMOTE startup parameter is in effect, it will accept and process the remote request. Otherwise it rejects the request.

The SSI startup parameter causes the server to accept requests from remote processors, as long as they are cluster members of the same single system image as the local processor (the processor on which it is running). If the local processor is not a single system image cluster member, only local connections are accepted.

## Restricting Access

For more information about the LOCAL, REMOTE and SSI startup parameters see [“How to Specify Startup Parameters”](#) on page 339.

6. Start the server, enter:

```
fileserv start
```

When the server is started, it will identify itself to CP as a GLOBAL resource, and it will accept file pool connections from remote users. Depending on how TSAF, ISFC, and AVS are used, this could be any processor in the collection or any processor in the SNA network.

## Restricting Access from Other Processors

---

To restrict file pool access to users on the local processor:

1. Inform remote users they will no longer be able to use the file pool.
2. Log on or reconnect to the server machine as follows:
  - If the server is currently processing in multiple user mode, reconnect to it and enter a STOP operator command. (For more about stopping multiple user mode processing, see [“Stopping Server Processing”](#) on page 65.)
  - Otherwise, if the server is not currently processing in multiple user mode, log on it.  
Enter STOP NOBACKUP if the BACKUP startup parameter is in effect and you do not want to back up the control data at this time.
3. Edit the DMSPARMS file and remove the REMOTE startup parameter from it. You do not need to specify LOCAL in the DMSPARMS file. LOCAL is the default. For more details on LOCAL, REMOTE and SSI startup parameters, see [“Startup Parameter Descriptions”](#) on page 342.
4. Start the server, enter:

```
fileserv start
```

When the server is started, it will identify itself to CP as a GLOBAL resource (as it always does, except for file pool IDs beginning with "VMSYS"). It will not, however, accept any remote connections to the file pool.

## Enrolling Remote Users

---

When your file pool is available to users on other processors, you can expect that some remote users will want to be enrolled in the file pool. The process of enrolling a remote user is the same as that described in [“Enrolling SFS Users and Creating BFS File Spaces”](#) on page 71, with the following considerations:

1. If the person is a user in another processor within the same TSAF or CS collection, the person's user ID should be unique within that collection. (One of the requirements to forming a TSAF or CS collection is that the installation ensures, administratively, that all user IDs within the collection are unique.) Assuming the user ID is unique, you can specify that user ID on the ENROLL USER command.

Notice that you do not need to tell the server the user's node ID. The server does not use node IDs or passwords. It bases its processing solely on the user ID. The server relies on APPC/VM communications to route requests from and responses to user virtual machines. These APPC/VM communications do not use a node ID. The server also assumes any user ID from which it receives a communication is a valid z/VM user ID, so it does not check passwords.

If the user wanted his or her top directory accessed as file mode A, the remote administrator would specify the file pool ID on the IPL CMS control statement in the z/VM system directory entry for the user. For CMS to successfully access the top directory when the user logs on, the processor on which the file pool is defined would already have to be running. Unless the processors within the collection are started at about the same time, it is not a good idea to have CMS automatically access a remote top directory as file mode A.

When the user is referring to files and directories in his or her file space, the user can omit the user ID from commands that allow directory names to be specified. Because the person's CP user ID was enrolled in the file pool, everything will work fine.

2. If the person uses the SNA network to access the file pool, you must:

a. Add the user to your z/VM system directory.

The user does not need to be assigned minidisks. The enrollment is needed only to give the person a user ID and password on your system. As always, the user ID must be unique within your collection. For example, suppose Chris has a user ID of CHRIS on a processor in the JC1 collection. She wants to be enrolled in a file pool you manage in the NEWYORK collection. There is already a user ID of CHRIS defined in your collection, so you add her to your z/VM system directory as CHRISB with a password of SECRET1.

b. Have the remote administrator update the user's z/VM system directory entry.

The administrator of the user's "home" system should add an APPCPASS z/VM system directory control statement that identifies the locally known LU name along with the user ID and password the user is assigned on your system. See [z/VM: Connectivity](#) for more about the APPCPASS control statement.

Continuing our example, the JC1 collection administrator would update user CHRIS's z/VM system directory entry with:

```
APPCPASS JC1      NEWYORK CHRISB SECRET1
```

When Chris enters a CMS command or uses a CSL program function that refers to your file pool in the NEWYORK collection, VM passes the user ID and password specified in APPCPASS on the communication from JC1 to NEWYORK. VM in NEWYORK verifies the user ID and password is valid for the NEWYORK collection. If it is, z/VM forward the request to the server, but does not forward the password. VM has already checked the password so the server does not need to check it.

c. Next, you would enroll CHRISB in your file pool.

When Chris wants to refer to files or directories in your file pool, she will have to specify CHRISB for the user ID in directory names, unless she has first issued the SET FILESPACE command to change the user ID (file space ID) default. If she were to let the user ID default (without first issuing the SET FILESPACE command), CMS in her machine would supply CHRIS instead. For more information on the SET FILESPACE command, see [z/VM: CMS Commands and Utilities Reference](#).

The effect of omitting the user ID from directory names without issuing the SET FILESPACE command is your server would initially receive a connection from CHRISB. Because CHRISB is enrolled, all is well. Next, it would receive a request for files or directories belonging to user ID CHRIS. Such a user ID may or may not be enrolled in the file pool. In any case, Chris is not requesting the correct files, and will not receive the desired results.

d. As a result of the preceding discussion, when you enroll remote SNA users, you should tell them the user IDs under which they are enrolled (that is, their file pool user IDs). You should also remind them to specify that user ID in CMS commands that operate on your file pool.

When referring to the user in commands that operate on the file pool, you should also remember to use the enrolled user ID (CHRISB) instead of the remote CP user ID (CHRIS). If, for example, you were to grant authority to Chris, you would use CHRISB.

## Using Nicknames and Local IDs for Remote Users

Some CMS commands that operate on file pools let the user specify a nickname instead of a user ID. When the user ID is on the same processor as the file pool, this works well. The z/VM CP user ID is the same as the user ID that is enrolled in the file pool.

For example, when a user on your system asks to be enrolled, you simply enroll the person's user ID in the file pool. Now suppose you create a nickname for that user. When you use the nickname, the nickname

## Coding a Program to Supply Local IDs

resolves to the local CP user ID, which is the same user ID under which the user was enrolled in the file pool. In this case, the nickname works perfectly.

When the user is remote, however, nicknames alone may not resolve properly. Suppose you have created a nickname TOM. Tom is a remote user whose user ID is TOMPAO. Tom's node ID is CARRIB1. When you created the nickname TOM, you specified TOMPAO in the Userid field and CARRIB1 in the Node field. This let you use the nickname TOM on communication commands like TELL and SENDFILE. Because the node ID is required for commands like TELL and SENDFILE, everything worked fine.

Now Tom wants to be enrolled in a file pool you happen to administer. Tom is not on your local system. (For example, Tom is on a system in your TSAF collection or on a system in the SNA network accessible through AVS and VTAM.) You must give him a local CP user ID. It would be convenient if you could give him a CP user ID of TOMPAO, but after some investigation you find someone is already assigned that user ID. So, you update your z/VM system directory with the user TOMP, which is not in use. Then you tell Tom his user ID on your system is TOMP. You also tell him his password and ask him to have his system administrator add an APPCPASS control statement to his z/VM system directory entry.

Next, you enroll TOMP in your file pool. You then enter a GRANT AUTHORITY command to grant Tom read authority on your top directory:

```
grant auth . to tom (read
```

CMS looks in your NAMES file for the nickname TOM. It finds TOM, but the user ID is TOMPAO. If CMS were to send that user ID to the server, your grant would succeed for the wrong user ID—remember Tom's user ID in your file pool is TOMP. The user ID alone in your NAMES file is not sufficient for remote users. Instead, a tag is also needed to identify the user's local ID on your system. This tag is known as the LOCALID tag.

Whenever you have a nickname for a remote user who is also enrolled in a local file pool, you should specify the LOCALID tag in your NAMES file. To specify the LOCALID on the NAMES screen, type LOCALID in one of the Tag fields and the person's local user ID (TOMP in our example) in the corresponding Value field. (See *z/VM: CMS User's Guide* or *z/VM: CMS Commands and Utilities Reference* for more about the NAMES command, including the LOCALID tag.)

When you specify a nickname on a command that causes CMS to send a request to a server machine, CMS uses the LOCALID instead of the user ID in its communication with the server. When you use that same nickname on communication commands like TELL or SENDFILE, CMS uses the USERID and NODE tags to route your communication. The same nickname can be used for both server-related commands and communication-related commands.

To prevent a user from inadvertently granting authority to the wrong person, CMS will not process a command that refers to a file pool if:

- A nickname is used, and
- A node is specified for the nickname that is different from the local node, and
- No LOCALID tag is specified.

When the above conditions are true, CMS cannot be sure the user ID specified for the USERID tag is, in fact, the same user ID enrolled in the file pool. And, as already mentioned, even if the user ID were enrolled, it might not belong to the same person.

Note that even if you could give Tom a user ID of TOMPAO on your system, you would still need to specify a LOCALID tag for TOMPAO. Because Tom has a node ID different from your node ID, you must specify the LOCALID tag. In this case, both the USERID and LOCALID tags would be set to TOMPAO.

## Coding a Program to Supply Local IDs for Remote Users

There is a way for you to spare users on your system from having to enter LOCALID tags in their NAMES files for remote users. The method for avoiding the LOCALID tag involves coding a program that automatically supplies local IDs for the remote users. Users on your system then need to enter only user



IDs and node IDs. For CMS to use this routine, it must be named DMSJNE. DMSJNE is an installation-wide exit.

The DMSJNE routine lets you use a table of user-ID/node-ID/local-ID mappings, or any other method of your choosing, to determine the local ID. For the purposes of this discussion, assume you are using a table. How you create, access, and maintain such a table is completely up to you; no skeleton is supplied. Just make sure that whenever a remote user is enrolled in a local file pool, a new record is added to the table. Also note that table entries should exist for all remote users, even those that are on other processors within your TSAF or CS collection.

## Nickname Resolution for SFS Commands

When a user on your system enters a nickname in a command that refers to a file pool, CMS resolves the nickname as shown in the following list (try #1; if #1 is not true, try #2; and so on).

1. If the user's NAMES file entry for the nickname contains a LOCALID tag, CMS uses the associated local ID.
2. If the NAMES file entry does not specify any node ID in either the "Node" field or the "List of Names" field, CMS uses the user IDs specified in the entry and assumes they are on the local processor.
3. If node IDs are specified in the NAMES file entry, but represent the local processor, CMS uses the specified user IDs.
4. If any node IDs are specified that represent other processors, CMS checks whether a nickname exists for that user-ID/node-ID combination. If a nickname exists, and the nickname entry contains a LOCALID tag, CMS uses the associated local ID.
5. CMS looks for the DMSJNE routine; if it exists, CMS calls it, passing the user ID and node ID. DMSJNE looks up the user ID and node ID combination in the user-ID/node-ID/local-ID table to determine the local ID. If DMSJNE finds and returns a local ID, CMS uses it.
6. If none of the above cases exist, CMS returns an error indicating the nickname cannot be resolved.

## Coding the DMSJNE Routine

When DMSJNE gets control from CMS, the general-purpose registers contain the following:

- Register 1 — address of a parameter list
- Register 14 — return address
- Register 15 — entry point address

CMS saves its register contents before calling your routine and restores them when you return to CMS.

Use the IBM-supplied DMSJNEPL macro to map the parameter list passed by register 1. DMSJNEPL provides the mapping shown in [Table 22 on page 239](#).

Label	Byte Offset (Hex)	Byte Length	Contents
JNEMOD	0	8	The name of the function being called (always 'DMSJNE ').
JNEFUNCT	8	8	Identifies the function being processed when the call to DMSJNE is made. This field always contains 'USERNODE', which identifies the function of translating a user-ID/node-ID combination to a local ID.
JNEUSER	10	8	User ID.
JNENODE	18	8	Node ID.
JNEFPID	20	8	File pool ID.

*Table 22. DMSJNEPL Macro Mapping (continued)*

<b>Label</b>	<b>Byte Offset (Hex)</b>	<b>Byte Length</b>	<b>Contents</b>
JNELOCID	28	8	ID of the user (identified by JNEUSER and JNENODE) on the processor in which the above file pool (identified by JNEFPID) resides.
JNEPLSIZ	N/A	N/A	An EQUATE for the length of the preceding structure (which begins with JNEMOD).

DMSJNE must use the user ID and node ID passed in the parameter list to look up the local ID in the user-ID/node-ID/local-ID table. Then DMSJNE must pass the local ID back to CMS in JNELOCID.

If DMSJNE processes successfully, it must set register 15 to zero before returning to CMS. If DMSJNE cannot find the user-ID/node-ID combination in the table, the routine must pass a non-zero value in register 15. DMSJNE should also return a non-zero return code for any other error condition in the routine (such as not being able to locate the table). DMSJNE must return to the calling routine when its processing is complete.

### **Installing the DMSJNE Routine**

After assembling DMSJNE, enter the LOAD and GENMOD CMS commands to create a module. Use the RLDSAVE option on the LOAD command. Place the module on the CMS system disk (usually file mode S).

At the beginning of nickname resolution processing, CMS detects whether the DMSJNE module exists and issues a NUCXLOAD command to load it as a nucleus extension. At the end of nickname resolution processing, if DMSJNE was loaded, CMS issues a NUCXDROP command.



## Chapter 14. Using Data Spaces (SFS Repository Servers Only)

The XC virtual machine architecture contains functions that improve the performance of certain kinds of processing. SFS can exploit these functions for directory control directories. The greatest benefits are for directories that are seldom updated and that are *accessed* in read-only mode by many users on the local processor. As an administrator, you control which directories are eligible for performance boosts and which are not.

### Data Spaces

In XC architectures, SFS uses the *data spaces* facility. In a XC context, a data space is an area of virtual storage that can be accessed by several virtual machines.

To improve performance for a particular directory, SFS *maps* the directory to a data space. The mapping causes CP to associate data space pages with the appropriate file pool minidisk blocks. SFS maps the blocks for all files within the directory and control blocks related to those files (File Status Tables—FSTs, for example).

Authorized users who access the directory in *read-only* mode do not communicate with the server. Instead, CMS, which is running in the users' machines, gets desired data *directly* from the data space. This provides several performance advantages:

- CMS retrieves data from shared virtual storage, which is far more efficient than having the server read from DASD for each user.
- *All* the performance overhead associated with communicating with the server machine is eliminated.
- When a user has an XC virtual machine, CMS does not use its own virtual storage to maintain control blocks associated with an accessed directory (such as FSTs). Instead, it gets the data directly from the shared copy in the data space. Because users refer to a single copy of the FSTs, rather than their own private copies, real storage requirements are reduced.
- XA virtual machines also use the data space when retrieving file data. However, control blocks associated with the accessed directory are maintained in the user's virtual storage, and the data space is accessed indirectly, through CP, so some efficiency is lost.

The use of data spaces is subject to the following restrictions:

- Those who access in read/write mode use the usual communications to the server.
- CSL routines that directly read the files without accessing the directory do not use the data space. However, if the directory has been accessed, the CSL routines read from the data space.
- When a user reads an INPLACE file, CMS in the virtual machine communicates with the server. It does not use the data space.
- Only user machines on the same processor as the server machine can use data spaces—data spaces cannot be accessed from other processors.

### Finding Data Space Candidates

Your data space candidates can be either existing directories or minidisks you would like to move into SFS. The minidisk or directory should have the following characteristics:

- It must be updated infrequently.
- Updating should be confined to brief periods of time, preferably when concurrent read access is low.
- You do not need file control access to the files.

## Selecting a File Pool

Some of the best candidates are read-only minidisks and directories that have a great number of local users, such as tools disks and product libraries<sup>5</sup>. Relative to directories, the benefit is increased performance. Relative to minidisks, the benefits include:

- Performance similar to minidisks with minidisk caching is obtained without the need for expanded storage.
- XC virtual machines share FSTs, which are kept in a separate data space instead of each virtual machine's address space. Furthermore, changes to the directory that affect the FSTs are managed automatically. In contrast, shared FSTs for minidisks must be kept in a shared segment and managed manually.
- Each read-only accessor of a directory control directory sees a consistent, unchanging version of that directory, even if the directory is changed. Minidisks do not have this stability.
- Space for an SFS directory is provided flexibly from a shared pool of DASD space. It is necessary to determine and adjust the size of a shared minidisk.
- Hierarchical directories can be used to organize the files better. For example, files needed by general users can be placed in a directory used with data spaces, while related files not usually accessed by general users can be placed in a separate directory that does not use data spaces.
- Users on other systems can access the data (although data spaces do not apply in this case).
- There is no need to maintain passwords and to communicate them to other users. Instead, users are authorized using the GRANT and REVOKE CMS commands.

## Selecting a File Pool

---

The simplest approach is to place the directories you plan to use with data spaces in your production file pool (the one your users are enrolled in). However, there are advantages to placing such directories in a separate, read-only file pool:

1. The read-only file pool 1) requires less scheduled downtime for administrative activities, and 2) is less subject to unplanned downtime, because it receives fewer server requests.
2. In large systems with several production file pools, it is easier for users to remember a single file pool of shared directories.

There should be no need to have more than one read-only file pool, even on the largest systems. The file pool capacity limits discussed in [“Step 1: Estimate Maximum File Pool Size”](#) on page 23 do not apply to a read-only file pool, as those capacity limits assume frequent changes to the production file pool.

The VMSYS file pool is a good choice for a read-only file pool.

## Configuring Your Server for Data Spaces

---

To configure a server machine for data spaces, you need to add three control statements to the z/VM system directory entry for the server. The statements can be specified in any order, but all three statements must precede any device statements in the server's directory entry. The first statement (MACHINE) specifies the virtual machine architecture. The remaining two statements (XCONFIG ADDRSPACE and XCONFIG ACCESSLIST) determine the number and total size of the data spaces the server can support. If the server reaches those limits, it will not be able to create additional data spaces. In this case, the server operator is notified, but users do not receive errors. Instead, CMS uses APPC/VM to communicate with the server. Except for performance differences, the lack of a data space is invisible to users.

The MACHINE statement you should specify is:

```
MACHINE XC
```

XC designates the ESA/XC or z/XC architecture, which is required for the server to use data spaces.

---

<sup>5</sup> The S and Y system disks are not candidates. SFS does not support auxiliary directories. Because the S- and Y-disks use auxiliary directories, they cannot be placed in SFS directories

The format of the XCONFIG ADDRSPACE control statement is:

```

▶▶ XCONFIG — ADDRspace — MAXNUMBER — nnnnn — TOTSIZE —  $\left. \begin{array}{l} \text{nnnnnM} \\ \text{nnnnG} \end{array} \right\}$  — SHARE ▶▶

```

The XCONFIG ADDRSPACE control statement authorizes the virtual machine to create data spaces. It also sets limits on the number and size of those data spaces. For MAXNUMBER *nnnnn*, specify the maximum number of data spaces you want the server to be able to create and have exist concurrently. MAXNUMBER can range from 1 to 32767. As a guideline, make *nnnnn* five times the number of directories you plan to make eligible for data spaces. This will accommodate multiple versions of directories.

For TOTSIZE, specify the maximum total size (in gigabytes or megabytes) of all the data spaces the server can create and have existing concurrently. (Each data space has a maximum size of 2G.) You can specify from 1 to 99999 for the *nnnnn* portion of the *nnnnnM* form. If you use the *nnnnG* form, specify from 1 to 8192. For SFS server machines, make TOTSIZE the maximum, 8192G, because it is better to control data space usage with the MAXNUMBER parameter.

The SHARE operand should always be specified for file pool server machines. It authorizes the server machine to share its data spaces with other virtual machines.

For example, to let the server create up to 100 data spaces you would specify:

```
XCONFIG ADDRSPACE MAXNUMBER 100 TOTSIZE 8192G SHARE
```

The final statement you need to add to the server's z/VM system directory entry is XCONFIG ACCESSLIST. An abbreviated format is:

```
XCONfig ACCEsslist ALSIZE nnnn
```

The ALSIZE operand determines the number of entries in the *host access list* for a virtual machine. CP maintains the host access list in real storage. When a virtual machine needs to use a data space, it must first add an entry for that data space to the host access list.

When the server creates a data space and maps a directory control directory to it, it places an entry in the list, creates the data space, and then removes the entry. So, whenever the server creates a data space, it uses one host-access-list entry. Because the server does work for more than one user at a time, it might use several entries in the host access list.

The maximum setting is recommended to be for ALSIZE, 1022, which makes up to four pages of real storage available for the server's host access list.

```
XCONFIG ACCESSLIST ALSIZE 1022
```

Initially, CP allocates enough real storage to support six host-access-list entries. It is unlikely the server would need to use 1022 entries, so setting ALSIZE high for an SFS server does not typically cause unnecessary real storage consumption. To control a server's use of data spaces, use the DATASPACE command to add directories to or remove directories from the server's data space eligibility list.

If the server reaches the capacity of its host access list, it is not able to create and map additional data spaces until host access list entries are removed. Instead, CMS uses APPC/VM to communicate with the server.

For more information about address spaces and data spaces, see *z/VM: CP Programming Services*. For more about the XCONFIG control statements, see *z/VM: CP Planning and Administration*.

To update the z/VM system directory entries for an SFS server machine:

1. Log on or reconnect to the server machine.
2. Stop multiple user mode processing if it is in effect.
3. Log off the server machine.
4. Update the z/VM system directory entry for the server machine using standard operating procedures for your installation.

## Restrictions for Using Data Spaces on EAV Minidisks

5. Log on the server machine and start multiple user mode processing.
6. Disconnect from the file pool server machine.

## Restrictions for Using Data Spaces on Extended Address Volume Minidisks

Any user storage group minidisk in the file pool that resides on an Extended Address Volume (EAV) minidisk must have the end extent of the minidisk below 65520 cylinders in order for the server to be able to create and use data spaces. You will receive a warning message if all or part of any minidisk resides at or above 65520 cylinders. Server processing continues, but any attempt to access any data space eligible directory in the file pool will result in no data space being created.

## Restrictions for Using Data Spaces on FBA Minidisks

If you are planning to use data spaces with FBA minidisks, you must ensure all FBA minidisks are allocated in 8-block increments and are aligned on a 4KB boundary. You will receive informational and warning messages if you have FBA minidisks not aligned properly. In some instances, such as adding minidisks using FILESERV MINIDISK or FILEPOOL MINIDISK commands or enabling a file pool in certain situations, processing will be unsuccessful if any FBA minidisks are not aligned on 4KB boundaries.

Before changing the alignment of an FBA minidisk, make sure the data in these minidisks is saved. Then, after the minidisk has been aligned and formatted or replaced, restore the data.

To align an FBA minidisk on a 4KB boundary, you update the MDISK statement in the system directory. A minidisk is aligned on a 4KB boundary when the block numbers that it starts and ends on are multiples of 8. To do this, make sure both of the following can be divided evenly by eight:

- The starting block
- The ending block plus one or the total number of blocks being defined

For example, consider the following SFS storage group minidisk definition:

```
MDISK 199 9336 6000 18250 SFSVOL R PURPLE ORANGE
```

$6000 / 8 = 750$ . The starting block is aligned on a 4KB boundary.  $18250 / 8 = 2281.25$ . 18250 is the number of blocks being defined. Because this number does not divide evenly by eight, the ending block is not aligned on a 4KB boundary.

To align the ending block on a 4KB boundary, you could round 2281.25 up to the nearest integer, 2282, then multiply by eight.  $2282 \times 8 = 18256$ . Actually, if you are aligning this minidisk so it can work with your SFS file pool server, you must round up. To replace a minidisk used with SFS, the new minidisk must be the same size or bigger than the original minidisk. So, to make this minidisk 4KB aligned, it could be redefined like this:

```
MDISK 199 9336 6000 18256 SFSVOL R PURPLE ORANGE
```

Note that this would work only if there is not another minidisk you would be overlapping.

## Moving System Minidisks into Data Spaces

To make existing minidisks data space eligible:

1. Log on a file pool administration machine.
2. Create a directory control directory.

```
create directory .tools (dircontrol
```

3. Make the directory eligible to be assigned to a data space by entering a CMS DATASPACE command for the directories chosen.

For example, to make the VMSYSU:BENDERT.BULLETINS directory eligible for data space use, enter:

```
dataspace assign vmsysu:bendert.bulletins
```

- Grant authority to users who will be using these files. For example:

```
grant authority .tools to public
```

- Access the minidisk to be moved. For example:

```
acc 195 q
```

- Access the new directory.

```
acc .tools b
```

- Copy the data.

```
copyfile * * q = = b (olddate
```

- Tell your users to access the directory instead of the minidisk.
- If you used the system profile to access the minidisk for all users, change the profile so the directory is accessed.

## Making Existing Directories Data Space Eligible

---

To make an existing directory data space eligible:

- Log on a file pool administration machine.
- If it is a file control directory, use the DIRATTR command to reset the directory attribute to DIRCONTROL. For details about using the DIRATTR command, see [z/VM: CMS Commands and Utilities Reference](#).
- Make the directory eligible to be assigned to a data space by entering a CMS DATASPACE command for the directories chosen.

For example, to make the VMSYSU:BENDERT.BULLETINS directory eligible for data space use, enter:

```
dataspace assign vmsysu:bendert.bulletins
```

Data space eligibility will persist across server sessions. It lasts until one of these things happens:

- You process a DATASPACE RELEASE command for the directory.
- The directory is erased.
- The directory attribute is changed from DIRCONTROL to FILECONTROL.

## Monitoring Data Space Usage

---

To determine which directories are eligible for data space residence, enter the QUERY DATASPACE command from an administration machine. For instance, to list the directories eligible for data spaces for the VMSYSU file pool, enter:

```
query dataspace vmsysu:
```

To determine who is currently accessing a directory control directory, use the QUERY ACCESSORS command:

```
query accessors vmsysu: (dataspace
```

When entered with the DATASPACE option, QUERY ACCESSORS shows whether the directory currently resides in a data space. It also shows how many levels of that directory exists. (You must have administration authority to use the DATASPACE option.)

For optimal server performance, ensure both of the following:

1. The number of data spaces being used is not excessive.
2. All directory control directories that are made eligible for data space usage are indeed using them.

To do this, first use the MAXNUMBER parameter on the XCONFIG statement to set a reasonable limit to the number of data spaces the server can have. See [“Configuring Your Server for Data Spaces”](#) on page 242 for how to do that. You can then rely on the DMS2020I warning message displayed on the server console to notify you if there are any cases where an eligible directory is not able to use a data space in an applicable situation. If this message occurs, see the message description for the exact cause and recommended corrective action in *z/VM: CMS and REXX/VM Messages and Codes*.

The most likely problem is the maximum number of data spaces allocated to the server has been reached. If this is the case, you may choose simply to increase the maximum. You do this by increasing the MAXNUMBER value on the XCONFIG ADDRSPACE control statement in the server's z/VM system directory entry.

Before doing so, however, you may wish to see if there are any directories that are using an excessive number of data spaces. This happens when multiple versions of the directory get created as a result of a series of changes being made to the directory at the same time other users are accessing it. Each such version requires a separate data space. To check for this condition, enter the QUERY ACCESSORS command with the DATASPACE option. If the results show there are one or more directories with an excessive number of levels, there are several actions you could take:

- Contact the person responsible for updating that directory and ask that changes be grouped together and made, when practical, during times of low directory usage. This will minimize the number of directory levels that get created.
- Request users to reaccess the directory. When users reaccess, the server lets them see the most current level. The server discards an older level when all users have released that level of the directory.
- Use the DATASPACE RELEASE command to make the directory ineligible for use with data spaces. This may be the best course of action if the directory is updated frequently and neither of the actions listed above are practical. In this case, you should also consider changing the directory to be a file control directory by using the DIRATTR command.

---

## Chapter 15. Generating a File Pool and Server

This chapter describes how to generate a new file pool server and file pool. File pool servers and their file pools can be generated for any of the following servers:

- SFS server repository services
- BFS server repository services
- CRR recovery server services
- FIFO server services

A file pool server may be dedicated to one of the services listed above or it can involve combinations of them. SFS repository file pool servers can be combined with BFS repository file pool servers simply by enrolling BFS file systems, for example. However, the CRR recovery server and FIFO server services must be established (or defaulted) by startup parameters.

The CRR recovery server is used by SFS operations, but not by BFS operations. The FIFO server is used by BFS operations but not by SFS operations.

To define, or *generate*, a new file pool you need to estimate how big the file pool will ever become. You also need to define a virtual machine (or update an existing one) and, in so doing, decide what the initial DASD storage for the file pool should be.

While many decisions you make during file pool generation are easily changed, some are not. To change certain file pool maximums, you must regenerate the file pool. File pool regeneration can be time-consuming, especially if there is a lot of data in the file pool. To avoid having to regenerate your file pool sometime in the future, it is recommended you read the entire chapter before trying to generate one.

The file pool and server being defined have characteristics similar to the IBM-supplied VMSYSU file pool and VMSERVU server. Below is a list of the steps to follow. You may want to use the list to check your progress.

### Step 1

Estimate the maximum number of users (see [Step 1](#)).

### Step 2

Estimate the maximum number of minidisks (see [Step 2](#)).

### Step 3

Determine initial DASD allocations (see [Step 3](#)). This step has five parts:

#### Step 3A

Determine the control minidisk allocation (see [Step 3A](#)).

#### Step 3B

Determine the file pool repository and CRR log minidisk allocations (see [Step 3B](#)).

#### Step 3C

Determine the work minidisk allocation (see [Step 3C](#)).

#### Step 3D

Determine the catalog minidisk allocations (see [Step 3D](#)).

#### Step 3E

Determine the user data minidisk allocations (see [Step 3E](#)).

### Step 4

Define a server machine (see [Step 4](#)).

### Step 5

Define an administration machine (see [Step 5](#)).

### Step 6

Log on the new server machine (see [Step 6](#)).



**Step 7**

Create a startup parameter file (see [Step 7](#)).

**Step 8**

Generate the file pool (see [Step 8](#)).

**Step 9**

Back up the control data (see [Step 9](#)).

**Step 10**

Start multiple user mode operation (see [Step 10](#)).

**Step 11**

Disconnect from the server machine (see [Step 11](#)).

**Step 12**

Log on the administration machine (see [Step 12](#)).

**Step 13**

Enroll the administration machine in the file pool (see [Step 13](#)).

**Step 14**

Create a PROFILE EXEC (see [Step 14](#)).

**Step 15**

Consider enrolling PUBLIC in the file pool (see [Step 15](#)).

**Step 16**

Enroll the initial set of file pool users (see [Step 16](#)).

**Step 17**

Set up the server for automatic starting (see [Step 17](#)).

**Step 18**

Copy the sample files to the new file pool (see [Step 18](#)).

The generation process described in this chapter sets up the file pool so its control data is backed up to a tape file. If you prefer to use DASD or if there are no tape drives available, you must decide where the backup file is to reside and also ensure there is enough DASD space to contain the control data backup. See [“Backing Up The Control Data” on page 104](#) for guidance on backing up the control data to DASD before following the generation steps in this chapter.

One of the main steps in generating a file pool is the definition of a virtual machine, also referred to as a server machine. (You can update an existing machine if you prefer, but when it is used as a server machine, it will not be available for other work.)

When defining the server machine, you need to allocate the file pool's first DASD storage in the form of minidisks. While minidisks can be added to the file pool at any time, there is a minimum number of minidisks you need to allocate. One minidisk is needed to serve as the machine's work disk, one is for the file pool control data, two are needed for the file pool repository logs, two are needed for the CRR logs (if generating a CRR recovery server), at least one for the catalogs, and at least one for user files. Beyond this minimum, you may want allocate additional minidisks for user files if you are willing to estimate how much space will be needed for them. If you do not want to make an estimate, you can just allocate one minidisk and add others later as needed.

After you have defined a server machine, and have an initial set of minidisks, you enter a command that formats those minidisks and places internal file pool control information on them. The command, entered at the server machine console, is named FILESERV GENERATE.

FILESERV GENERATE not only initializes the minidisks, it also establishes maximum values for the file pool, such as the maximum number of users that will ever be enrolled, and the maximum number of minidisks that will ever be added to the file pool. FILESERV GENERATE processing does not automatically choose these values. You must supply them.

Defining a virtual machine (or updating one) is done by making changes to the z/VM system directory. The changes needed are described in this chapter. To make the changes, you can use the z/VM system directory program, which is described in [z/VM: CP Planning and Administration](#).



After the file pool is generated, you should enroll the administration machine, an initial set of users, and, perhaps, enroll PUBLIC.

**Note:** If this is a dedicated CRR recovery server or FIFO server, you would not enroll a set of users and you would not enroll PUBLIC.

Finally, you can optionally set up your system so the server machine is automatically logged on and started in multiple user mode when z/VM is started.

## Step 1: Estimate the Maximum Number of Users (MAXUSERS)

**Note:** If you are generating an SFS file pool server, please read [“Step 1: Estimate Maximum File Pool Size”](#) on page 23 before beginning this step.

One of the input parameters you must supply to the FILESERV GENERATE (and can supply to the FILESERV REGENERATE) command is MAXUSERS. The MAXUSERS control statement specifies the estimated maximum number of file pool users who will be authorized to create objects (such as directories, files, SFS aliases or Byte File System (BFS) links) in the file pool. FILESERV GENERATE (and FILESERV REGENERATE) processing use this value to compute the maximum logical size of the catalog space. This value computes how much space is needed on the control minidisk to represent logical catalog space. The amount of space assumes the average user has 100 directories and 80,000 files.

The following formula shows the amount of logical space reserved for the file pool as specified by MAXUSERS (rounded up to the nearest multiple of 101):

```
Number of Catalog Space Blocks = MAXUSERS * 85
```

These blocks are allocated approximately 50% to data blocks and 50% to index blocks.

The value specified for MAXUSERS does not limit the number of users you can enroll in the file pool or the number of Byte File Systems you can define in the file pool. It is possible, for example, to specify 500 MAXUSERS and enroll 1000 users in the file pool. But it does determine how much logical catalog space is available to be used when objects are created by users of the file pool. The logical space in the catalog can be monitored by looking at QUERY FILEPOOL REPORT (with CATALOG option specified) command output under Catalog Space Information or by looking at QUERY FILEPOOL CATALOG command output. Overestimating MAXUSERS will cause the server to use a nominal amount of additional space for its control disk. But, using some extra space is better than having to regenerate the file pool when the logical catalog space becomes full. See [Chapter 11, “Regenerating a Repository File Pool,”](#) on page 217 for more information about regenerating a file pool.

If you are generating a dedicated CRR recovery server or FIFO server (and therefore do not have users connected for file pool functions), you might want to save some space and choose a MAXUSERS value of 5. MAXUSERS must be at least 5, but no greater than 32767.

## Step 2: Estimate the Maximum Number of Minidisks (MAXDISKS)

Another input parameter you must supply to the FILESERV GENERATE command is MAXDISKS. MAXDISKS represents the maximum number of minidisks that will ever be allocated to the file pool storage groups. (The control and log minidisks are not counted in the MAXDISKS value.) MAXDISKS also represents the highest storage group number you are able to use.

You cannot exceed the MAXDISKS value without regenerating the file pool, so your estimate for MAXDISKS should be generous. The cost of being generous is a nominal amount of additional storage needed for the control minidisk.

After the file pool is available for access, you will probably find you add relatively large minidisks to the file pool when you begin to run out of space. To simplify DASD management, consider assigning one entire DASD volume at a time to the file pool.

Because each minidisk added will probably be large enough to support more than one user, your choice for MAXDISKS should be equal to or somewhat smaller than MAXUSERS.

## Determine Initial DASD Allocations

For most file pools, a MAXDISKS value of 500 is a good choice. If you are generating a dedicated CRR recovery server or FIFO server (therefore do not have users connected for file pool functions), you might want to save DASD and choose a MAXDISKS value of 2. MAXDISKS must be at least 2, but no greater than 32767.

### Step 3: Determine Initial DASD Allocations

When defining a new server machine, you use MDISK z/VM system directory control statements to make initial minidisk allocations for the file pool. Determine allocations for:

1. Control minidisk
2. Log Minidisks
3. Work minidisk
4. Catalog minidisk
5. User data minidisk.

As you determine the allocations, you may want to write them down in MDISK statement form, because that is ultimately what you will be entering into the system. The MDISK z/VM system directory control statement is described in [z/VM: CP Planning and Administration](#).

#### Step 3A: Determine the Control Minidisk Allocation

To determine the size of the control minidisk, you must decide how many bytes (characters) you would like the file pool to be able to hold. There are no formulas that can tell you what this number might be because it varies depending on the number of users and the number and size of the files they create. If you have some experience with other file pools or with non-file pool minidisk used at your installation, you might have some idea of how much space the average person uses. Those averages can serve as guides.

In any case, you should overestimate the size of the file pool. Be extremely generous. The cost of estimating too high is that a nominal amount of control minidisk space may not be used. If you estimate too low, however, you have to regenerate the file pool. File pool regeneration is a time-consuming process.

Space in the file pool is used for both user data and catalog data. The catalogs keep track of users' files and directories. This catalog DASD space counts towards the total file pool size because the control disk keeps track of all of it.

You should estimate the size of your file pool in terms of *gigabytes (GB)*. A gigabyte is  $2^{30}$  (1 073 741 824), or roughly one billion bytes. A GB consists of 262 144 4KB blocks. A small file pool, for example, would be 0.5GB to 1GB, while a large file pool might be 800GB to 900GB. After you have estimated the size of your file pool, you can estimate the size of the control minidisk needed to support it by referring to [Table 23 on page 250](#). The estimates of the control minidisk size in [Table 23 on page 250](#) are based on MAXUSERS=1000 and MAXDISKS=500. If MAXUSERS is greater than 1000, use the following formula to determine the number of control minidisk 512-byte blocks to add to the number in [Table 23 on page 250](#):

$$((\text{MAXUSERS} - 1000) / 100) * 170$$

If MAXDISKS is greater than 500, use the following formula to determine the number of control minidisk 512-byte blocks to add to the number in [Table 23 on page 250](#):

$$((\text{MAXDISKS} - 500) / 100) * 4$$

Size of File Pool (gigabytes)	Size of File Pool (4KB blocks)	Size of Control Minidisk (512-byte blocks) (MAXUSERS=1000, MAXDISKS=500)
0.1	26215	1821

Table 23. Estimates of Control Minidisk Sizes (continued)

Size of File Pool (gigabytes)	Size of File Pool (4KB blocks)	Size of Control Minidisk (512-byte blocks) (MAXUSERS=1000, MAXDISKS=500)
0.5	131072	1905
0.7	183501	1943
1.0	262144	2008
5.0	1,310,720	2831
10	2,621,440	3862
50	13,107,200	12105
100	26,214,400	22407
1000	262,144,000	207865

For a repository file pool server, the minimum size you should specify for a control minidisk is 5 cylinders on any count-key-data DASD model or 5000 blocks on FB-512 devices. The maximum size of a control minidisk depends on the values chosen for MAXUSERS and MAXDISKS. Generally, the maximum number of 512-byte blocks for the control minidisk is about 1,600,000 blocks.

For a dedicated CRR recovery or FIFO server, specify 2 cylinders on any count-key-data DASD model or 2000 blocks on FB-512 devices. This should be sufficient in all cases.

For a CMS formatted minidisk, the number of 512-byte blocks per cylinder varies based on the device type. To determine this number, see the table of maximum minidisk sizes in the MDISK directory control statement description in [z/VM: CP Planning and Administration](#).

## Where to Place the Control Minidisk

The placement of the control minidisk is not critical to server performance because it usually has a relatively low level of I/O activity. It is, however, a good idea to place the control minidisk on the volume (or one of the volumes) that contains storage group 1. The control minidisk and storage group 1 are recovered together. Placing the control minidisk on one of the same volumes that hold storage group 1 reduces the total number of volumes that contain these areas. This reduces the probability that a given DASD error will be on a volume that contains one or more of these areas.

**A Note about Maximum File Pool Sizes:** This note applies only to repository file pool. Dedicated CRR and FIFO file pools are minimal in size. A server can manage about 8 *terabytes* (TB) of information. A TB is 2 to the 40th power (1 099 511 627 776), or roughly one trillion and 99 billion bytes of information. Therefore, 8TB is almost 9 trillion bytes.

While it is possible for a server to manage 8TB, the server does need a control minidisk large enough to keep track of all the data. The maximum size of a control minidisk depends on the values chosen for MAXUSERS and MAXDISKS. Generally, the maximum number of 512-byte blocks for the control minidisk is about 1,600,000 blocks.

## Step 3B: Determine the File Pool Repository & CRR Log Minidisk Allocations

Every file pool must have two identically-sized file pool repository log minidisks. Also, if this is a CRR recovery server, you also need two identically-sized CRR log minidisks. To achieve an identical size, both file pool repository log minidisks and CRR log minidisks must be defined on the same type of device. (Internal space allocation algorithms make it impossible to define two minidisks having the same size on two different device types.) Where both the devices are FB-512 devices, you can consider them to be the same device type, even if they are different DASD models.

## Determine Initial DASD Allocations

For CRR recovery servers, specify 2 cylinders on any count-key-data DASD model or 2000 blocks on FB-512 devices for each CRR log minidisk and specify 1 cylinder on any count-key-data DASD model or 1000 blocks on FB-512 devices for each file pool repository log minidisk.

**Note:** IBM recommends CRR recovery servers run without control data backups (use the NOBACKUP startup parameter).

For repository file pool servers, the size of the file pool repository log minidisks needed for your file pool depends on these factors:

- Whether CMS file pool facilities back up the control data
- The rate at which log data is generated

If you continue to follow the instructions in this chapter, your repository file pool will be set up to use CMS file pool facilities for backing up the control data. These backup copies are needed to restore the control data should a media error occur.

### File Pool Log Size When File Pool Backup Is Used

When file pool backup facilities back up the control data, the file pool repository log must be large enough to hold all changes to the control data done between backups. On smaller systems, you would typically back up the control data one time a week, perhaps twice. On larger systems, you would back up the control data more often, perhaps one time a day. At the end of a backup, server processing automatically reclaims file pool repository log space—that is, it "cleans house" so it can reuse the space. Between backups, data accumulates in the log.

To prevent the file pool repository log from filling, which would cause server processing to stop, the server automatically backs up the control data when 80% of the file pool repository log space is filled. By backing up the control data, the server frees file pool repository log space so it can continue operation. While this avoids an untimely shutdown, an unscheduled backup can itself be inconvenient. Because server performance is degraded for the duration of the backup, a backup that starts automatically during peak hours is undesirable. Also, if you back up to tape, and a tape is not mounted when the logs reach 80% full, the logs may become so full the server will have to stop. Special consideration should be taken to prevent this situation from happening. Ideally, the file pool repository log should be large enough to prevent unexpected backups. For more about monitoring file pool repository log usage, see [“Limit Monitoring” on page 58](#).

After you get a feel for the amount of logging your users cause, you can use the FILESERV LOG command to increase or decrease the size of the file pool repository logs. For an initial estimate, use this rule:

```
LOGSIZE = HOURS * ACTIVE_USERS * 0.08
```

The above formula estimates LOGSIZE, which represents the size of each file pool repository log in megabytes. The minimum size you should specify for a file pool repository log minidisk is 5 cylinders on any count-key-data DASD model or 5000 blocks on FB-512 devices.

HOURS is the maximum number of hours you want the file pool to be available between control data backups. If your system runs unattended over the weekend, you would want to specify the number of hours that would lapse between your Friday evening and Monday morning backups. If you stop your system over the weekend or if an operator is available, the amount of hours between backups should be determined as a function of your recovery time requirements. See [“Frequency of Control Data Backup” on page 106](#) for more information.

ACTIVE\_USERS is your estimate of the number of active users using this file pool, averaged over the period of hours between successive control data backups. *Active users* is the average number of logged-on users who have interacted with the system (in any way) during a typical one-minute interval. VMPRF output (or output from an equivalent program) can help you to estimate the average number of active users.

Note that ACTIVE\_USERS \* 0.08 is an estimate of the rate at which file pool repository log data is generated. If you have monitored the rate on other similar file pools in your installation, you may adjust the formula to better suit your environment. If, for example, you find your users consume twice as much

file pool repository log space as the above formula predicts, you would multiply the result of the formula by two.

One last fact you should consider before defining your file pool repository logs is that the time it takes to recover the control data is directly related to the amount of data recorded in the file pool repository log. If you back up the control data weekly, the file pool repository log will reflect all changes to the control data made during the week. If an error occurs just before you make your weekly backup and you must restore the control data, the server's restore processing will have to redo all the changes made during the week. This could take quite a bit of time. If you backup the control data frequently, the amount of time it would take to restore the control data would be proportionally reduced.

If your installation requires minimal loss of computer service for any error, it might be worthwhile to back up the control data often. On the other hand, if you are willing to endure a delay if the control data needs to be restored, you might back up the control data less frequently. By changing the time between backups you are essentially making a trade-off between time spent making backups and (potential) time spent in doing restores. Because DASDs are usually reliable, many users prefer to risk a lengthy restore (that might never be needed) rather than spending the time to do frequent backups.

## Where to Place the Logs

For best performance:

- For repository file pool servers, define the log minidisks so they are on separate channels and control units. This maximizes the likelihood the server can do I/O to the two log minidisks in parallel, thus reducing response time.

For CRR recovery servers, if there is high CRR log minidisk I/O activity, define the CRR log minidisks so they are on separate channels and control units. The log minidisks have low I/O activity; therefore, there are no log minidisk placement performance considerations.

**Note:** You should get high CRR log minidisk I/O activity only if you have heavy usage of applications or participating products designed to exploit CRR. Such programs would have to concurrently update multiple protected resources (for example, multiple repository file pools) or use protected conversations.

- For repository file pool servers, when the log minidisks are placed on DASD volumes that have no other I/O activity, log minidisk I/O is optimized because there is very little seek activity. Therefore, to minimize seek time, place each log minidisk on a DASD volume that otherwise has low I/O activity, or place them on a DASD volume where most of the I/O activity is to a small area on that volume. In this latter case, place the log minidisk adjacent to this area.

For CRR recovery servers, if there is high CRR log minidisk I/O activity, then define the CRR log minidisks so they are on DASD volumes with low I/O activity. The log minidisks have low I/O activity; therefore, there are no log minidisk placement performance considerations.

*Always* define the logs (and the CRR logs, if this is a CRR recovery server) on separate volumes of the same device type. (Where both the devices are FB-512 devices, you can consider them to be the same device type, even if they are different DASD models.) If you define both logs (or both CRR logs) on the same volume, loss of that volume would cause the loss of both logs (or both CRR logs), which defeats the purpose of having two logs for each file pool repository and CRR.

**Note:** Two file pool repository logs that contain duplicate information are required because the file pool repository logs synchronize updates made to the file pool. When a user or application commits or rolls back a work unit, the repository file pool server uses the logs to help ensure the integrity of the file pool. If the file pool repository log data was lost, it would be possible for the repository file pool to be left in an inconsistent state. In that case, you would need to return the repository file pool to a consistent state by restoring it from backups made prior to the error.

Because restoring an entire file pool is, in many cases, a time-consuming task, server processing requires the use of two duplicate file pool repository log minidisks. This reduces the chance of losing the file pool repository log data and lengthy recovery measures. Duplicate minidisks are required because, unlike user data and control data, the file pool repository logs are not backed up.

## Determine Initial DASD Allocations

For CRR recovery servers, two CRR logs that contain duplicate information are required because the CRR logs synchronize updates made to multiple file pools within a coordinated transaction.

### File Pool Log Size When File Pool Backup Is Not Used

This section applies only to repository file pool servers, not dedicated CRR or FIFO servers.

Most readers should skip this section. You need to read this only if you are *not* planning to use the CMS file pool backup facility.

When the NOBACKUP startup parameter is in effect, the server usually reclaims file pool repository log space every few minutes (depending on the amount of file pool activity). A long-running logical unit of work, however, would prevent space from being reclaimed. To avoid running out of log space (which would cause the server to roll back long-running logical units of work), the logs should be large enough to hold the logging that is done in a few hours. As a start, try using 10 cylinders (on any count key data device type) or 10,000 blocks (on any fixed-block architecture device). The minimum size you should specify for a file pool repository log minidisk is 5 cylinders on any count-key-data DASD model or 5000 blocks on FB-512 devices.

For more about monitoring log usage, see [“Limit Monitoring” on page 58](#). Use the FILESERV LOG command to increase or decrease the size of the logs.

### Step 3C: Determine the Work Minidisk Allocation

A server machine must have a read/write work disk. The work disk should be large enough to hold:

- POOLDEF file
- DMSPARMS file
- Audit file (optional)
- Control data backup file (optional, and not recommended for dedicated CRR recovery or FIFO servers)
- Other incidental files

All of these files were discussed in Chapter 1. With the exception of the optional audit and backup files, none of the files use an inordinate amount of space. For all types of file pool servers, two or three cylinders on any count-key-data device type should be adequate. For FB-512 devices, 2000-3000 blocks should be adequate. If you find you have not allocated enough space, you can later increase the minidisk size without affecting the file pool. Because it is not heavily used in comparison to the other file pool minidisks, the physical location of this minidisk is not critical.

The output of a control data backup would not typically be directed to the work minidisk. It would, instead, be directed to tape or to another file pool or to some other DASD volume. See [“Backing Up The Control Data” on page 104](#) for more information about control data backups.

**Note:** Control data back ups are recommended for repository file pools, but not for dedicated CRR or FIFO file pools.

In these instructions, set up your file pool to back up to tape. It is easy to switch from tape to DASD or from DASD to tape, after the file pool is generated.

While the “work disk” is typically a minidisk, it could instead be an SFS directory in some other file pool. In this case, the other file pool must be available when this server machine is started. The instructions in this chapter assume you are using a minidisk for the work disk. You should use a minidisk unless you have experience in generating file pools.

### Step 3D: Determine the Catalog Minidisk Allocation

File pool generation processing requires you define at least one minidisk to hold catalog data. You can define more than one if you wish, or you can define just one and add minidisks later as catalog space is consumed. Adding a minidisk (or minidisks) can easily be done by issuing the FILESERV MINIDISK or FILEPOOL MINIDISK command. So, in some senses, it does not matter what you initially allocate.

For a rough estimate of the amount of DASD space that might be needed for the catalogs, allow 400 bytes for every SFS base file, alias, and directory that will be created in the near future. Allow 500 bytes for each BFS file or other BFS object.

For repository file pool servers, the minimum size you should specify for a catalog storage group minidisk is 5 cylinders on any count-key-data (CKD) DASD model or 5000 blocks on FB-512 devices.

For dedicated CRR recovery or FIFO servers, you should specify 2 cylinders on any count-key-data (CKD) DASD model or 2000 blocks on FB-512 devices for the catalog storage group minidisk.

## Where to Place the Catalog Minidisks

For repository file pool servers, when allocating minidisks to the catalog storage group, keep the following points in mind:

- A sizable fraction of all file server I/Os are to storage group 1. Therefore, it may be necessary to spread the minidisks of storage group 1 across multiple DASD volumes. This prevents any one volume from becoming an I/O bottleneck. As a rule of thumb, spread storage group 1 across USERS/130 DASD volumes. (USERS is the value specified for the USERS startup parameter.)

**Note:** It is best not to spread storage group 1 across more volumes than you really need to avoid an I/O bottleneck. The more volumes storage group 1 resides on, the greater the chance a DASD error will be on one of these storage group 1 volumes. Whenever a DASD error occurs on a storage group 1 volume, the file pool control data must be restored, which is an off-line operation.

- A storage group 1 minidisk is a relatively small, I/O-intensive area. Therefore, it is recommended that, for any given volume, it be placed adjacent to any other small, frequently referenced areas on that volume in order to minimize seek time.

See [Chapter 9, “Managing Storage,” on page 195](#) for additional DASD storage considerations.

For dedicated CRR recovery or FIFO servers, the catalog minidisks have very low usage so their location is not important.

## Step 3E: Determine the User Data Minidisk Allocation

File pool generation processing requires you define at least one minidisk to hold user data. You can define more than one if you wish, or you can define just one and add minidisks later as users consume space. Adding a minidisk (or minidisks) can be done by issuing the FILESERV MINIDISK or FILEPOOL MINIDISK command. So, in some senses, it does not matter what you initially allocate.

If your users will easily use an entire DASD volume, it is best to allocate the entire volume to the file pool. This reduces the amount of space management you have to do—let the server worry about whose files go where.

For repository file pool servers, the minimum size you should specify for a user data minidisk is 5 cylinders on any count-key-data (CKD) DASD model or 5000 blocks on FB-512 devices.

For a dedicated CRR recovery or FIFO server (does not have users connected for file pool functions), specify 1 cylinder on any count-key-data (CKD) DASD model or 1000 blocks on FB-512 devices.

## Where to Place the User Minidisks

For repository file pool servers, even though you may be allocating only one minidisk to one user storage group, you should do so with the following points in mind:

- The more storage groups you have, the less time it takes to recover any given one of them. Define enough storage groups such that your recovery time requirements are met.
- Because of I/O load balancing considerations, it may often be necessary to have part of storage group 1 on the same volume as one of the user storage groups. It should not, however, generally be necessary to place more than one user storage group on a given volume. It is best to have just one user storage group per volume so you do not need to restore more than one user storage group in the event of a DASD error on a given volume.



## Define a Server Machine

- When a storage group spans volumes, the server satisfies storage requests for that storage group by allocating space evenly across those volumes. This tends to spread the I/O demand for that storage group evenly across those volumes. For the overall I/O demand to those volumes to be balanced, you should:
  - Make sure any non-file pool repository space on those volumes is of low usage or uniform usage. Avoid volumes with uniformly high usage.
  - Make sure all volumes within a storage group are of the same device type, or have similar performance characteristics.
  - Give the storage group about the same amount of space on each volume.

See [Chapter 9, “Managing Storage,” on page 195](#) for additional DASD storage considerations.

For CRR recovery servers, the user data minidisk has very low usage so its location is not important.

## Step 4: Define a Server Machine

---

By the time you reach this step, you should know the following:

- Your estimate for MAXUSERS
- Your estimate for MAXDISKS
- Your estimate, in gigabytes, of the maximum size of the file pool
- A size and physical volume location for each of these minidisks:
  - One control minidisk
  - Two file pool repository log minidisks
  - Two CRR log minidisks (if this is a CRR recovery server)
  - One work minidisk
  - One or more catalog minidisks
  - One or more user data minidisks

Using your local operating procedures, update the z/VM system directory entries for the new server. [Figure 35 on page 257](#) shows an example of the z/VM system directory entries. The text following the figure explains the example statements. The user ID of this example server is FPSERV1.



```

1--> USER FPSERV1 FPSERVPW 32M 32M BG
2--> ACCOUNT acctnum distcode
3--> OPTION MAXCONN 2000 NOMDCFS APPLMON ACCT QUICKDSP SVMSTAT
4--> SHARE REL 1500
5--> MACHINE XC
6--> XCONFIG ADDRSPACE MAXNUMBER 100 TOTSIZE 8192G SHARE
7--> XCONFIG ACCESSLIST ALSIZE 1022
8--> IUCV ALLOW
9--> IUCV *IDENT RESANY GLOBAL
10-> IPL CMS
11-> POSIXOPT SETIDS ALLOW
12-> CONSOLE 009 3215 T userid
13-> SPOOL 00C 2540 READER *
14-> SPOOL 00D 2540 PUNCH A
15-> SPOOL 00E 1403
16-> LINK MAINT 190 190 RR
17-> LINK MAINT 193 193 RR
18-> LINK MAINT 19D 19D RR
19-> MDISK 191 3380 cyl 2 volser1 W
    MDISK 250 3380 cyl 30 volser2 R readpw writepw
    MINIOPT NOMDC
    MDISK 405 3380 cyl 10 volser3 R readpw writepw
    MINIOPT NOMDC
    MDISK 406 3380 cyl 10 volser4 R readpw writepw
    MINIOPT NOMDC
    MDISK 260 3380 cyl 10 volser2 R readpw writepw
    MDISK 310 3380 cyl 200 volser5 R readpw writepw
    MDISK 311 3380 cyl 200 volser3 R readpw writepw
    MDISK 312 3380 cyl 200 volser8 R readpw writepw
    MDISK 313 3380 cyl 200 volser9 R readpw writepw
    MDISK 505 3380 cyl 2 volser6 R readpw writepw
    MINIOPT NOMDC
    MDISK 506 3380 cyl 2 volser7 R readpw writepw
    MINIOPT NOMDC

```

Figure 35. Example z/VM System Directory Control Statements for Server FPSERV1

Use the CMS saved segment name (for example CMS) and device types applicable to your z/VM environment. See *z/VM: CP Planning and Administration* for more about z/VM system directory control statements.

Other virtual machines should not have LINK control statements in their z/VM system directory entries that refer to file pool minidisks. If such links exist, server processing may not be able to successfully initialize.

**Note:** The above does not apply to the “work” minidisk.

**Statement 1: USER FPSERV1 FPSERVPW 32M 32M BG**

Defines a virtual machine named FPSERV1 with a password FPSERVPW. Neither the machine name nor the password have any significance to the file pool server. Use machine names and passwords of your own choosing.

Although the minimum virtual storage needed for a server is much smaller, 32MB is recommended because that value is adequate for most file pool server machines, and generally prevents you from having to reset the virtual storage size as the use of the file pool grows.

The privilege classes in the example are B and G. It is possible, though not recommended, to define a server machine with only class G. Privilege class B is needed to attach tapes. Class B is needed if you plan to direct control data back ups or the output of security audits to tape. Both backups and security audits are optional operations, but almost everyone backs up the control data, and most of those people direct the backup output to tape. Because file pool is being set up to back up the control data to tape, specify class B.

**Statement 2: ACCOUNT acctnum distcode**

This statement specifies an account number (*acctnum*) to which a virtual machine may charge its costs and a distribution code (*distcode*) to designate where printed output is to go.

### Statement 3: OPTION MAXCONN 2000 NOMDCFS APPLMON ACCT QUICKDSP SVMSTAT

The MAXCONN value of the OPTION control statement determines the maximum number of IUCV connections allowed for a virtual machine. The server uses IUCV to communicate with user machines, and uses the DASD Block I/O System Service to access the minidisks in the file pool. The DASD Block I/O System Service also uses IUCV. The maximum MAXCONN value is 65,535—the default is 64. You should specify 2000 to avoid having to frequently re-specify MAXCONN as the use of your file pool grows. A value of 2000 should be adequate for all but very large file pools (that is, file pools with more than 700 logged-on users).

For repository file pool servers, you can estimate the number of IUCV connections that a server needs as follows:

$$\text{MAXCONN} = (\text{SFS\_USERS} * 3) + (\text{BFS\_USERS} * 12) + \text{DISKS}$$

For USERS, substitute the number of users expected during peak system activity. A file pool user is someone who accesses files or directories managed by the repository file pool server. (The sum of SFS\_USERS and BFS\_USERS should be the same as the *nnnnn* value used in the USERS startup parameter and Chapter 20, “File Pool Server Startup Parameters,” on page 339.) For DISKS, substitute the number of minidisks defined for the file pool. Include in this count the number of user data minidisks, catalog minidisks, both file pool repository log minidisks, the control minidisk, and if present, both CRR log minidisks. (This is not applicable to FIFO file pool servers.) Do not include the work minidisk in the count unless the “work disk” is really an SFS directory in another file pool.

For CRR recovery servers, you can estimate the number of IUCV connections a server needs as follows:

$$\text{MAXCONN} = (\text{USERS} + \text{RESOURCES} + 20)$$

For USERS, substitute the number of logged-on CRR users, expected during peak system activity. A CRR user is someone who uses:

- Resources that participate in CRR (for example, SFS file pool)
- Protected conversations

If USERS is not known, then substitute the maximum total logged-on CMS users. For RESOURCES, substitute the number of resources participating in CRR (such as SFS file pools). The constant 20, accounts for AVS links and links to other CRR recovery servers and occasional file pool ID links such as for QUERY FILEPOOL REPORT.

For dedicated FIFO servers:

$$\text{MAXCONN} = (\text{USERS} * 2) + \text{SERVERS}$$

For USERS substitute the number of users of the FIFO server expected during peak activity. For SERVERS substitute the approximate number of servers that designate this FIFO file pool in their file pool startup parameter (FIFO (*filepoolid*)).

Server processing sets aside virtual storage for each potential connection. Thus, setting MAXCONN too high may cause virtual storage to be exhausted in the server machine.

Repository file pool servers use NOMDCFS to allow the server machine to use minidisk caching at a rate not limited by the Fair Share Limit. CRR recovery servers should not use NOMDCFS, because the CRR minidisks do not benefit from minidisk caching.

The APPLMON operand lets the server machine generate information needed for performance monitoring.<sup>6</sup>

The ACCT operand is required if you intend to use the file pool accounting facilities for this server. If you do not intend to use the accounting facilities, you can omit ACCT.

---

<sup>6</sup> Server processing issues a DIAGNOSE X'DC' whenever it is initialized. The APPLMON operand is required for virtual machines that issue DIAGNOSE X'DC'

The QUICKDSP operand causes the server to be added to the dispatch list immediately when it has work to do, without waiting in the eligible list.

The SVMSTAT operand specifies this server is a service virtual machine, which causes the server's monitor statistics to be reported separately from end-user virtual machines.

**Statement 4: SHARE REL 1500**

This statement places the server machines in a more favorable position in the z/VM dispatch queue. Use this statement for all server machines.

**Statement 5: MACHINE XC**

This statement specifies the type of architecture a virtual machine simulates. XC designates ESA/XC or z/XC architecture; ESA designates ESA/390 architecture.

For SFS file pool servers, designate XC if you want to exploit data spaces. CRR recovery servers, BFS, and FIFO do not exploit data spaces, and, therefore, should be set to ESA.

**Statement 6: XCONFIG ADDRSPACE MAXNUMBER 100 TOTSIZE 8192G SHARE**

This statement authorizes the virtual machine to create data spaces and sets limits on the number and size of those data spaces. As a guideline, specify the maximum number of data spaces you want as five times the number of directories you plan to make eligible for data spaces. You need to specify this statement if you intend to use data spaces. See [“Configuring Your Server for Data Spaces” on page 242](#) for more information about the XCONFIG statements.

**Statement 7: XCONFIG ACCESSLIST ALSIZE 1022**

This statement sets up space for the server's host access list. You need to specify this statement if you intend to use data spaces. See [“Configuring Your Server for Data Spaces” on page 242](#) for more information about the XCONFIG statements.

**Statement 8: IUCV ALLOW**

This statement lets any other virtual machine establish a communication path with the server machine.

For most server machines, you should specify IUCV ALLOW as shown. If you prefer, however, you can use different IUCV control statements to control the communications between user machines and the server. Server processing uses APPC/VM to communicate with users. See [z/VM: Connectivity](#) for more about inter-machine communications.

**Statement 9: IUCV \*IDENT RESANY GLOBAL**

This IUCV \*IDENT statement authorizes the virtual machine to identify itself as a resource owner.

**RESANY** must be specified because server processing lets you change the resource name when you start it (using the FILESERV startup parameter named FILEPOOLID).

**GLOBAL** should be specified in most cases.

For dedicated CRR recovery servers, regardless of whether the file pool ID has a VMSYS prefix, you must always specify GLOBAL.

For repository and FIFO servers, if the file pool ID does not have a VMSYS prefix, you must always specify GLOBAL. If the file pool ID does have a VMSYS prefix, you can specify LOCAL or GLOBAL (LOCAL is sufficient, because in this case, the server always identifies the file pool ID TPN as local).

For more information on how file pool ID TPNs are identified, see the startup parameter [FILEPOOLID](#).

**Statement 10: IPL CMS**

This statement causes CMS to be loaded into the server machine when it is logged on. If you prefer, you can use the appropriate saved segment name for your installation. IPL ZCMS can also be an appropriate choice.

Do not specify the AUTOOCR initialization parameter for CMS. To enter some of the dedicated maintenance mode commands for the server, it may be necessary to avoid executing the server's PROFILE EXEC. This is not possible if you specify AUTOOCR.

**Statement 11: POSIXOPT SETIDS ALLOW**

This statement allows the BFS server to have CP change its POSIX security value in support of opening executable files.

### Statement 12: CONSOLE 009 3215 T *userid*

During usual operation, you will be running the server machine in disconnected mode. So, you should specify a secondary user ID. This is known as Single Console Image Facility (SCIF). If you will be acting as the administrator or operator of the file pool, you should specify your own user ID. If you are generating the file pool for a secondary user ID who is to receive console output, specify that person's user ID.

### Statement 13: SPOOL 00C 2540 READER \*

This statement defines at 00C a spooled 2540 reader that can read any class of spool file.

### Statement 14: SPOOL 00D 2540 PUNCH A

This statement defines at 00D a spooled 2540 punch that creates only class A spool files.

### Statement 15: SPOOL 00E 1403

This statement defines at 00E a spooled 1403 printer that processes class A spool files (when the spooling class is omitted, it defaults to A).

### Statement 16: LINK MAINT 190 190 RR

This is a link statement for access to CMS.

### Statement 17: LINK MAINT 193 193 RR

This statement is necessary to make the MAINT 193 minidisk available to the server machine. Some of the files a server needs (such as the FILESERV EXEC) are located on this minidisk. This LINK statement is, therefore, required.

### Statement 18: LINK MAINT 19D 19D RR

This is a link statement for the Help Facility.

### Statement 19: MDISK 191 3380 *cyl* 2 *volser1* W

This is the MDISK statement for the read/write work disk on which the POOLDEF file will reside. Use virtual device number 191 and access mode W as shown. In this example, a device type of 3380 that consists of two cylinders is used. You also need to supply the starting cylinder number and the volume serial number.

### Remaining MDISK and MINIOPT NOMDC statements:

```
MDISK 250 3380 cyl 30 volser2 R readpw writepw
MINIOPT NOMDC
MDISK 405 3380 cyl 10 volser3 R readpw writepw
MINIOPT NOMDC
MDISK 406 3380 cyl 10 volser4 R readpw writepw
MINIOPT NOMDC
MDISK 260 3380 cyl 10 volser2 R readpw writepw
MDISK 310 3380 cyl 200 volser5 R readpw writepw
MDISK 311 3380 cyl 200 volser3 R readpw writepw
MDISK 312 3380 cyl 200 volser8 R readpw writepw
MDISK 313 3380 cyl 200 volser9 R readpw writepw
MDISK 505 3380 cyl 2 volser6 R readpw writepw
MINIOPT NOMDC
MDISK 506 3380 cyl 2 volser7 R readpw writepw
MINIOPT NOMDC
```

In the preceding MDISK statements examples, the supplied virtual device numbers (191, 250, 405, 406, and others) and also the device type (3380) are shown. The MDISK statements shown are for count-key-data devices, so you must supply starting cylinder number (*cyl*) followed by the number of cylinders (the number of cylinders was supplied in the example). For FB-512 devices, specify the appropriate block relocation factor (*blkr*) and number of blocks (*blks*). You also must supply the volume serial number (*volser1* through *volser8*) of the real DASD volume on which the minidisk resides. Use access mode R and then supply a read password (*readpw*) and a write password (*writepw*) for each minidisk.

#### Note:

1. Set the volume serial numbers (*volser*) as appropriate for your DASD volumes.
2. **The MDISK control statements for the file pool must include both a read and a write password. Also, it is strongly recommended the minidisks have an access mode of R.** Specify your own read and write passwords for *readpw* and *writepw*.

3. Except for the 191 minidisk, any virtual device numbers can be used in any order. Server processing does not rely on any particular device numbers. In this example, virtual device numbers 250, 405, 406, 260, 310, 311, 312, 313, 505, and 506 were chosen arbitrarily.
4. When you enter the FILESERV GENERATE command (see [“Step 8: Generate the File Pool”](#) on page 267), you must be prepared to supply the virtual device numbers of the minidisks you used in the MDISK statement. You should write down your choices of virtual device numbers along with how they are to be used (for example, control minidisk, file pool repository log minidisk 1, and so on).
5. FBA minidisks must start and end on 4KB boundaries or data spaces will not be created by this file pool server, even for data that is on aligned FBA or CKD DASD.
6. In this example, the minidisk having a virtual device number of 191 is the work minidisk. The 250 minidisk is the file pool control minidisk. It is defined for an IBM 3380.

Minidisks 405 and 406 are the file pool repository logs. Notice the file pool repository logs are defined on different volumes (*volser3* and *volser4*). They are both the same size and are defined on the same device type, which again is an IBM 3380.

Minidisk 260 is the initial catalog minidisk. It is on the same volume as the control minidisk (250) because both are recovered as a unit.

Minidisks 310, 311, 312, and 313 are for user data.

Minidisks 505 and 506 are for the CRR logs. Notice the CRR logs are defined on different volumes (*volser6* and *volser7*). They are both the same size and are defined on the same device type, which again is an IBM 3380.

7. In the preceding example of MDISK control statements, a server is shown performing both repository file pool server functions and CRR recovery server functions.

If this was a repository file pool server, you would not have minidisks for the CRR logs (505 and 506).

If this was a dedicated CRR recovery or FIFO server, the sizes of the control minidisk (250), file pool repository log minidisks (405 and 406), catalog minidisk (260), and the user minidisk (310) would be much smaller. For example, each minidisk would probably be only 1 to 2 cylinders. Also, you would not need the other user minidisks (311, 312, and 313), because CRR recovery or FIFO server activity is not intended for file pool repository use. You would still have the CRR log minidisks (505 and 506).

Use the MINIOPT NOMDC NOCACHE control statement to inhibit expanded storage caching; omit the NOCACHE option (MINIOPT NOMDC) when you are using DASD fast-write. One reason to do this is to improve system performance by not caching the file pool repository log and CRR log minidisks. [Table 24 on page 261](#) lists the uses of the MINIOPT control statement.

*Table 24. Uses of MINIOPT NOMDC and MINIOPT NOMDC NOCACHE*

Minidisk	Repository File Pool	Dedicated CRR or FIFO File Pool
Control	Yes	Yes
Log 1 and Log 2	Yes	Yes
Catalog	No	Yes
User	No	Yes
CRR Log 1 and Log 2	N/A	Yes

## Step 5: Define an Administration Machine

Now define a new virtual machine (or update an existing one) to serve as the first file pool administration machine. The machine you select should be the same machine you specified as the secondary user for

the server machine in the preceding step (Step 4). See the description of the CONSOLE control statement in that step.

By using the same virtual machine as the server's secondary user and as an administration machine, you can control the server's operation and act as a file pool administrator from a single machine.

After you have selected the machine, use your installation's usual operating procedures to update the z/VM system directory. If the virtual machine is new, define a usual user virtual machine with the following adjustments. If the virtual machine already exists, just make the adjustments. The adjustments are:

1. Set up the default file pool for the administration machine.

This is applicable to all file pool servers.

The following instructions assume the file mode A of the administration machine is an SFS directory, not a minidisk. They also assume the machine's primary file pool is the one you will be creating. To have the administration machine's top directory automatically accessed as file mode A, add the following control statement to the z/VM system directory entries:

```
IPL CMS PARM FILEPOOL RESEARCH
```

For RESEARCH, substitute the ID you will be using for your file pool. If the administration machine already has an IPL CMS statement, update it as shown above (do not create a second IPL CMS statement).

2. Allocate a small work minidisk to the administration machine.

This is applicable only to repository file pool servers.

A work minidisk is needed for the administrator to back up or restore the storage group to which he or she is assigned. The backup and restore operations for user storage groups (FILEPOOL BACKUP and FILEPOOL RESTORE commands) write temporary data (global variables) to a read/write file mode. During the backup or restore, users are unable to write to the storage group. So, if the administrator is backing up or restoring his or her own storage group, the administrator will need to temporarily access a minidisk (or an SFS directory in another file pool) as file mode A. The minidisk only needs to be a cylinder or two (1000 to 2000 blocks on FB-512 devices).

## Step 6: Log On the New Server Machine

Now log on the server machine. CMS will automatically be loaded. After CMS is loaded, press Enter. Because the 191 minidisk is not yet formatted, you receive this error message:

```
A(191) device error
```

Format the 191 minidisk by entering:

```
format 191 a
```

The FORMAT command will ask you whether to "erase all files" on the minidisk. Verify the minidisk is the correct one, then reply YES by entering:

```
1
```

The FORMAT command will prompt you for the minidisk volume label (disk label). Enter any valid volume label value.

At this time you might also want to create a PROFILE EXEC for the server machine. The PROFILE EXEC should contain these two commands:

```
'ACCESS 193 C'  
'CP SET EMSG ON'
```

The ACCESS command is needed because some of the files the server needs are on MAINT's 193 minidisk. SET EMSG ON should be specified so that the message number will be included in the messages

the server displays. You can use the message number to look up messages in [z/VM: CMS and REXX/VM Messages and Codes](#).

After you create the PROFILE EXEC, process it by entering:

```
profile
```

The access to MAINT's 193 minidisk is needed for the next step.

## Step 7: Create a Startup Parameter File

Whenever you enter a FILESERV command of any kind, it processes a set of startup parameters. The startup parameters determine the operational characteristics of the server. Most of these characteristics are not important right now, such as the number of internal buffers used and various internal thresholds. The server decides these for itself. (All startup parameters are described in [Chapter 20, "File Pool Server Startup Parameters,"](#) on page 339.)

Some startup parameters, however, you should specify. Those startup parameters are:

### ADMIN *userid\_list*

This startup parameter lets you identify one or more file pool administrators. For repository file pool servers, the file pool administrators typically enroll users, create file spaces, delete file spaces, and modify the amount of space allocated to file spaces.

**Note:** Administration authority may also be granted to server machines that interact with the file pool. For example, if DFSMS/VM is to manage the file pool, additional ADMIN parameters will need to be added. See [z/VM: DFSMS/VM Customization](#) for more information. The file pool administrator typically looks at internal file pool information not available to most users and also grants administrator authority to other users. (More about administration authority is in ["Administration Authority"](#) on page 139.)

### SAVESEGID *savesegname*

This startup parameter indicates the server uses code that has been loaded into a physical saved segment. A *physical saved segment*, for our purposes, can be thought of as a copy of executable code stored on DASD and that can be efficiently shared among users. CMS uses a saved segment for the server code because it is more efficient than loading the server code from MAINT's 193 minidisk. The default physical saved segment for the server code is CMSFILES. The CMSFILES segment is pre-built on the z/VM installed system that you loaded from the z/VM System DDR image.

It is also possible for you to create another saved segment containing the server code. If you change the name from CMSFILES, you must update the SAVESEGID startup parameter with the new server code segment name. If, for example, you create a saved segment named SFSSERV which contains the server code, you would specify the following startup parameter:

```
SAVESEGID SFSSERV
```

See [SAVESEGID](#) and [NOSAVESEGID](#) for more about these startup parameters.

**Note:** Using a physical saved segment is efficient, especially if multiple servers are used on your processor. You should change to NOSAVESEGID only if the physical saved segment was deleted. If NOSAVESEGID is specified, CMS uses its usual search order through accessed file modes to find the server code and load it. (The server code resides on MAINT's 193 minidisk.)

### FILEPOOLID *filepoolid*

This startup parameter lets you name the file pool. This name, or *file pool ID*, is the name by which the file pool will be known in the user community. Choose the file pool ID carefully. While it is easy, mechanically, to change the file pool ID, it is not practical to change it often. Every time you change it, you need to tell everyone what the new name is, and then they have to modify their execs and, perhaps, programs to use the new name, in addition to updating appropriate log name tables.

**Note:** For CRR recovery servers, the file pool is small and the only user is the CRR administrator.



The rules for naming a file pool are similar to those for naming files: up to eight alphabetic or numeric characters without imbedded blanks. The first character must be a character from A to Z. The words SYSTEM, ANY, and ALLOW are not valid. It is best to choose an appropriate name that users can easily remember. Some examples are:

```
FILEPOOLID PUBS
FILEPOOLID DESIGN
FILEPOOLID LIBRARY
FILEPOOLID VOID
```

Additional considerations for choosing a file pool ID are in the FILEPOOLID startup parameter, refer to Chapter 20, “File Pool Server Startup Parameters,” on page 339.

If you intend to make this file pool available to users on other processors, you may need to update your communications directory (SCOMDIR NAMES file) with this file pool ID. Refer to Chapter 13, “Setting Up a File Pool for Remote Use,” on page 233 for more information.

### USERS *users*

For repository file pool servers, *users* is your best estimate of the number of logged-on repository users expected during peak system activity. A repository user is someone who accesses objects in either SFS or BFS file spaces in a file pool repository server. (The value should be the same as the USERS value used in calculating the MAXCONN value for the server machine as described in Chapter 15, “Generating a File Pool and Server,” on page 247.)

For CRR recovery servers, *users* is your best estimate of the number of logged-on CRR users that you anticipate exploiting CRR by doing sync point processing (two-phase commit). The default of 100 should be sufficient for many installations.

**Note:**

It is strongly recommended you do not use the same server for both file pool repository and CRR services. However, if you do so, combine the two values for *users* that would have been used if the file pool repository and CRR services were assigned were to separate servers.

For FIFO servers, *users* is determined as follows:

- If FIFO service is done by the repository server (FIFO startup parameter defaulted), there are no adjustments to *users* for the FIFO service.
- If the current startup is for a file pool server that is doing FIFO services for one or more other file pool servers (ones that specified in their FIFO startup parameter the file pool that is currently being started-up), *users* should include BFS users for each of those other file pool servers, where those BFS users actively employ FIFO objects. That is, you should allow for the total of all potential concurrent BFS FIFO users that are serviced by the current FIFO server.

If the current file pool server provides both its own repository services and FIFO services for other file pools, *users* should include the combination of the value calculated for current file pool repository users with the value calculated above for other file pool FIFO users.

An error occurs if you specify a numeric value of less than 1 or greater than 32767. The default value is 100.

This startup parameter is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

**Note:**

1. Choose *users* carefully. The server optimizes its processing based on this value.
2. In dedicated maintenance mode, this startup parameter is used only to calculate the CATBUFFERS and CTLBUFFERS values. You do not need to change the USERS value when running the server in dedicated maintenance mode.



**DFSMS**

This startup parameter is only used if DFSMS/VM is to manage the SFS file pool. (NODFSMS is the default parameter.)

**Note:** Specifying DFSMS has implications for administering the file pool you should be aware of, including these tasks:

- backup
- moving a user to a different storage group
- enrolling users

The VMSYS file pool that is optional when installing z/VM is required when you plan to install DFSMS/VM.

The use of DFSMS/VM to manage a file pool should be part of an overall SFS storage management strategy. See *z/VM: DFSMS/VM Storage Administration* and *z/VM: DFSMS/VM Planning Guide* for more information about DFSMS/VM and how it can help you manage SFS file pools.

If you are generating a CRR recovery server, you must also select these startup parameters:

**CRR**

This startup parameter indicates the server is going to be the CRR recovery server. For more information, see [CRR](#).

**LUNAME *luname***

This startup parameter supplies a fully qualified LU name to this CRR recovery server. For more information, see [LUNAME](#).

Having selected an initial set of start-up parameters, create the parameter file. Using the virtual machine ID of the server you created (in this example, the ID of the server is FPSERV1), enter this command:

```
copyfile server dmsparms c fpserv1 dmsparms a
```

The SERVER DMSPARMS file is a start-up parameter skeleton file. Modify the copy just made, so enter:

```
xedit fpserv1 dmsparms
```

You see:

## Creating a Startup Parameter File

```
FPSEVR1 DMSPARMS A1 F 80 Trunc=80 Size=21 Line=0 Col=1 Alt=0
```

```
==== * * * Top of File * * *
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
==== ADMIN userid_list
==== BACKUP | NOBACKUP
==== CATBUFFERS catbuf
==== CTLBUFFERS ctlbuf
==== DUMP | FULLDUMP | NODUMP
==== FIFO filepoolid
==== FILEPOOLID filepoolid
==== FORMAT | NOFORMAT
==== GROUPTHRESH threshold_percent
==== LOCAL | REMOTE
==== NOACCOUNT | ACCOUNT
==== NOAUDIT | AUDIT
==== NOCRR | CRR
==== NODFSMS | DFSMS
==== NOSECURITY | ESECURITY
==== NOETRACE | ETRACE
==== NOITRACE | ITRACE buffersize
==== NOLUNAME | LUNAME luname
==== NOMSGS | MSGS
==== NORESTORE | RESTORE
==== RESYNCINTERVAL timed_wait
==== SAVESEGID savesegname | NOSAVESEGID
==== USERS users
====> _
```

X E D I T 1 File

Delete all lines except the lines containing ADMIN, BACKUP, FILEPOOLID, SAVESEGID, USERS, DFSMS if you are planning to use DFSMS/VM, CRR and LUNAME *luname* if you are generating a CRR recovery server and FIFO if you are creating a separate FIFO server.

**Note:** You may also want to keep the RESYNCINTERVAL *timed\_wait*. parameter if you want to change the timed wait interval between resynchronization attempts from the default of 600 seconds (10 minutes). The *timed\_wait* value is the number of seconds between resynchronization attempts.

Your screen should look something like this:

```
FPSEVR1 DMSPARMS A1 F 80 Trunc=80 Size=7 Line=0 Col=1 Alt=14
```

```
==== * * * Top of File * * *
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
==== ADMIN userid_list
==== BACKUP | NOBACKUP
==== FILEPOOLID filepoolid
==== NOCRR | CRR
==== NOLUNAME | LUNAME luname
==== SAVESEGID savesegname | NOSAVESEGID
==== USERS nnnnn
==== * * * End of File * * *

====> _
```

X E D I T 1 File

Now type your choices for the remaining startup parameters.

Specify the file pool administrator, the file pool ID, and the expected number of users. In the following example, the user ID of GEORGE was specified as the administrator, RESEARCH as the file pool ID, and 100 users.

Specify SAVESEGID, NOSAVESEGID, or delete the entire line, as appropriate. In the following example, SAVESEGID CMSFILES is specified because it is running multiple servers on the system.

Specify BACKUP or NOBACKUP. If you are generating a repository file pool server, or if this is the **not recommended** combined repository file pool server and CRR recovery server, specify BACKUP. If you are generating a dedicated CRR or FIFO server, specify NOBACKUP. In this example, BACKUP is specified because this is a combined repository file pool server and CRR recovery server.

Specify CRR or NOCRR and LUNAME or NOLUNAME *luname*. If you are generating a CRR recovery server, or the **not recommended** combined repository file pool server and CRR recovery server, specify CRR and LUNAME *luname*. If you are generating a repository file pool server, specify NOCRR and NOLUNAME. In this example, CRR and LUNAME GDLVME.RECOVER are specified because this is a combined repository file pool server and CRR recovery server.

The startup parameters can be in any order in the file. After updating the file, your screen should be similar to this:

```

FPSERV1  DMSPARMS A1  F 80  Trunc=80 Size=7 Line=0 Col=1 Alt=14

===== * * * Top of File * * *
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
===== ADMIN GEORGE B
===== BACKUP
===== FILEPOOLID RESEARCH
===== CRR
===== LUNAME GDLVME.RECOVER
===== SAVESEGID CMSFILES
===== USERS 100
===== * * * End of File * * *

=====> _

X E D I T  1 File

```

Now enter a FILE command.

## Step 8: Generate the File Pool

Before starting this step, make sure you have your virtual device numbers nearby. Now enter:

```
fileserv generate
```

FILESERV GENERATE does the following:

1. Issues CMS FORMAT and RESERVE commands for the file pool minidisks (including the CRR log minidisks, if this is a CRR recovery server).

Depending on the number and size of your initial minidisks, this could take quite a bit of time.

2. Initializes the file pool minidisks (including the CRR log minidisks, if this is a CRR recovery server).

FILESERV GENERATE processing places internal control information on the file pool minidisks. This control information is needed for usual server operation.

3. Creates the POOLDEF file.

The POOLDEF file has a file name that is the same as the file pool ID you picked in the last step. The file type is POOLDEF. FILESERV GENERATE processing creates the file on the first read/write file mode in the server machine's search order, which happens to be the 191 work disk.

During its processing, FILESERV GENERATE calls XEDIT to display a file containing control statements. The file is named \$TEMP \$POOLDEF. It is a copy of an IBM-supplied sample POOLDEF file named

## Generating the File Pool

DMSIBM POOLDEF. (FILESERV GENERATE makes the copy during its processing.) Your screen looks like this:

```
$$TEMP  $POOLDEF A1  F 80  Trunc=80 Size=10 Line=0 Col=1 Alt=0

===== * * * Top of File * * *
      |...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
===== MAXUSERS=1000
===== MAXDISKS=500
===== DDNAME=CONTROL          VDEV=301
===== DDNAME=LOG1             VDEV=302
===== DDNAME=LOG2             VDEV=303
===== DDNAME=BACKUP   DISK   FN=FILEPOOL  FT=BACKUP  FM=*
===== DDNAME=MDK00001        VDEV=304      GROUP=1    BLOCKS=0
===== DDNAME=MDK00002        VDEV=305      GROUP=2    BLOCKS=0
===== DDNAME=CRR1            VDEV=306
===== DDNAME=CRR2            VDEV=307
===== * * * End of File * * *
=====> _

                                           X E D I T  1 File
```

You need to modify the file to describe your file pool as follows:

1. Change the MAXUSERS value to the value you chose in [“Step 1: Estimate the Maximum Number of Users \(MAXUSERS\)”](#) on page 249. If, for example, you chose a MAXUSERS value of 2000, you would change that line to:

```
MAXUSERS=2000
```

For most repository file pools, the value in the file (1000) is a good choice. For dedicated CRR recovery or FIFO servers, specify a value of 5.

**Note:** Only one MAXUSERS control statement is allowed in the file.

2. Do the same for the MAXDISKS value. If, for instance, you chose 175 for MAXDISKS, you would change the line to:

```
MAXDISKS=175
```

For most repository file pools, the value in the file (500) is a good choice. For dedicated CRR recovery or FIFO servers, specify a value of 2.

**Note:** Only one MAXDISKS control statement is allowed in the file.

3. Change the virtual device numbers (VDEV=nnnn) for the control minidisk (301), the file pool repository log minidisks (302 and 303), the catalog minidisk (304), and the CRR log minidisks (306 and 307) to the virtual device numbers you used on the MDISK statements, see [“Step 4: Define a Server Machine”](#) on page 256.

**Note:** This example assumes you defined only one catalog minidisk, as was recommended in [“Step 3D: Determine the Catalog Minidisk Allocation”](#) on page 254.

If you used virtual device number 250 for the control minidisk, 405 and 406 for the file pool repository log minidisk, 260 for the catalog minidisk, and 505 and 506 for the CRR log minidisks, you would change the file to this:

```

:
===== DDNAME=CONTROL          VDEV=250
===== DDNAME=LOG1             VDEV=405
===== DDNAME=LOG2             VDEV=406
===== DDNAME=BACKUP   DISK   FN=FILEPOOL  FT=BACKUP  FM=*
===== DDNAME=MDK00001        VDEV=260      GROUP=1    BLOCKS=0
===== DDNAME=MDK00002        VDEV=305      GROUP=2    BLOCKS=0
===== DDNAME=CRR1            VDEV=505
```

```
==== DDNAME=CRR2          VDEV=506
:
```

**Note:** The line containing:

```
==== DDNAME=MDK00002      VDEV=305      GROUP=2      BLOCKS=0
```

has not yet changed. It is for a user data minidisk. This line will be updated in step “5” on page 269.

The BLOCKS parameter is automatically handled by the server; you do not need to modify it. See “FILESERV GENERATE” on page 513 for more information on the BLOCKS parameter.

4. This example assumes you are backing up the control. If you want to use tape, change the line containing DDNAME=BACKUP to this:

```
DDNAME=BACKUP TAPE VDEV=181
```

If you want to back up to DASD instead, you have a few choices to make regarding the destination of the output file. You also need to ensure there is enough DASD space to contain the backup file. If you have a tape drive available, it might be easier to specify a tape drive for now and switch to DASD later. If, instead, you prefer to use DASD immediately or there is no tape drive available, change the line containing DDNAME=BACKUP to:

```
DDNAME=BACKUP DISK FN=FILEPOOL FT=BACKUP FM=B
```

The only change made is a replacement of FM=\* with FM=B. You can also change the file name and file type if you wish.

The control statement above identifies FILEPOOL BACKUP B as the file that is to contain the control data backup. You can specify any file mode letter for \*, so long as there is sufficient space on that file mode to contain two copies of the control data backup. Do not leave \*, in the file—replace it with some file mode letter. (Global file ID characters are not supported for the definition of the backup file.) This will allow you to get through file pool generation, but you should refer to “Backing Up The Control Data” on page 104 for DASD considerations before attempting to back up the control data. (The destination file can be easily changed after the file pool is generated.)

**Note:** For a CRR recovery server, IBM recommends you do not back up control data and therefore you would delete the line containing DDNAME=BACKUP. But, in this example, the server shown is performing both repository file pool server and CRR recovery server functions, which is **not recommended**. Therefore, you would keep the line containing DDNAME=BACKUP for this server's repository file pool functions.

5. If you have allocated only one user data minidisk, change the virtual device number in the line containing DDNAME=MDK00002 to the one you specified on the MDISK statement. If you allocated more than one, as in this example, you need to duplicate this line. Suppose you defined four user minidisks. You would enter:

```
=3"== DDNAME=MDK00002      VDEV=305      GROUP=2      BLOCKS=0
```

Then press enter. You see:

```
==== DDNAME=MDK00002      VDEV=305      GROUP=2      BLOCKS=0
==== DDNAME=MDK00002      VDEV=305      GROUP=2      BLOCKS=0
==== DDNAME=MDK00002      VDEV=305      GROUP=2      BLOCKS=0
==== DDNAME=MDK00002      VDEV=305      GROUP=2      BLOCKS=0
```

Now change the virtual device numbers to match those in your MDISK statements. You should also change the "MDK00002" value to ascend in sequential numeric order (MDK00002, MDK00003, MDK00004, . . . , MDKnnnnn). Suppose you used virtual device numbers 310 to 313 for the user data minidisks. You would modify the statements to look like this:

```
==== DDNAME=MDK00002      VDEV=310      GROUP=2      BLOCKS=0
==== DDNAME=MDK00003      VDEV=311      GROUP=2      BLOCKS=0
==== DDNAME=MDK00004      VDEV=312      GROUP=2      BLOCKS=0
==== DDNAME=MDK00005      VDEV=313      GROUP=2      BLOCKS=0
```

**Note:** For a CRR recovery server, you would only want to allocate one user data minidisk. In this example, four user data minidisks were allocated because it is for the generation of the **not recommended** combined repository file pool server and CRR recovery server.

Finally, you should decide whether you want to assign the minidisks to different storage groups. The primary reason for doing this is to make recovery procedures more convenient. Storage groups are the units used when you need to back up user data. If you assign all your user minidisks to a single storage group, you must back up all the user data at the same time. For a large file pool, this is not desirable. Many people prefer to keep the amount of DASD assigned to a storage group to a manageable size.

Another reason you might want to assign minidisks to different storage groups is to control where the user data resides. Storage groups provide the only mechanism of control over where file pool data resides on physical volumes. When users are enrolled, they are given space in a particular storage group. If you know only IBM 3380s are assigned to that storage group, you know all users assigned to that group are using those 3380s. One reason for doing this is to separate fast devices from slower ones.

One last consideration for storage groups is that it lets you cluster, in the file pool, groups of related users or applications. If, for example, your file pool was to serve a set of inventory control applications, a few personnel departments, and a few design departments, you might consider using three storage groups—one for each.

Suppose minidisks 310 and 312 are defined on IBM 3390s while 311 and 313 are defined on IBM 3380s. You have a set of engineering applications you would like to support that logically belong together and you would like to use the IBM 3390s. You decide to put minidisks 310 and 312 in one storage group, and minidisks 311 and 313 in another. You would modify the GROUP operand on the statements to reflect this:

```
==== DDNAME=MDK00002      VDEV=310      GROUP=5      BLOCKS=0
==== DDNAME=MDK00003      VDEV=311      GROUP=2      BLOCKS=0
==== DDNAME=MDK00004      VDEV=312      GROUP=5      BLOCKS=0
==== DDNAME=MDK00005      VDEV=313      GROUP=2      BLOCKS=0
```

Remember storage group 1 is reserved for the catalog data. Do not assign user minidisks to storage group one. Also notice that storage groups do not need to be used in ascending numeric sequence. You can use storage group 5 without having first used storage groups 3 and 4.

For repository file pool servers, when it is time to enroll users, the administrator of this file pool would give space in storage group 5 to those using the engineering applications. Other users would be given space in storage group 2.

After making your changes, enter:

```
file
```

FILESERV GENERATE processing continues using the control statements you specified.

## Step 9: Back Up the Control Data

For CRR recovery servers, IBM does not recommend backing up control data so skip this step.

For repository file pool servers, proceed with this step.

After FILESERV GENERATE processing successfully completes, back up the control data. Enter:

```
fileserv backup
```

If your backup is directed to tape, you will see informational messages followed by a prompt to have a tape mounted to contain the backup. The FILESERV BACKUP command uses a standard label (SL) tape file. When the tape is mounted, respond to the prompt. If you mount a tape that does not contain a VOL1 label on it, you will also be prompted for the tape volume ID:

```
DMS446R Enter 1(volid) (WRITE(volid)), 2 (REJECT), or 3 (NEWTAPE)
```

If the correct tape volume is mounted, enter:

```
1(valid)
```

This will cause a VOL1 label to be written on the tape. The 1 or WRITE must be followed by a left parenthesis and no intervening blanks, with a one to six character volume serial number to be written in the label. If the tape volume is incorrect, enter:

```
3
```

to allow the mounting of the correct tape volume. After you respond to the prompts, the server will back up the control data.

If your backup is directed to DASD, the backup proceeds without further prompting.

## Step 10: Start Multiple User Mode Processing

---

Now, start the server to access the file pool in usual multiple user mode:

```
fileserv start
```

## Step 11: Disconnect from the Server Machine

---

After you see the informational messages that indicate the server is processing usually, disconnect from the server machine:

```
#cp disc
```

## Step 12: Log On the Administration Machine

---

Log on the administration machine that also serves as the secondary user. CMS is loaded automatically because of the IPL CMS control statement in the administration machine's z/VM system directory entry (see [“Step 5: Define an Administration Machine”](#) on page 261).

This paragraph applies to repository file pool servers, not dedicated CRR recovery or FIFO servers. Because the administration machine may not yet be enrolled in the file pool specified as its default, CMS may not be able to access a top directory for the administration machine. The administration machine will not have a file mode A. Error messages will be displayed. Ignore these error messages because the administrator will be enrolled in the next step.

## Step 13: Enroll the Administration Machine in the File Pool

---

For dedicated CRR recovery or FIFO servers, skip this step.

For repository file pool servers, proceed with this step.

Even though the “administration” machine has file pool administration authority, it is not enrolled in the file pool. To enroll the administration machine and give it space in the file pool, enter:

```
enroll user adminid filepoolid (blocks blks storgroup storgrp)
```

For *adminid*, specify the user ID of the administration machine. For *filepoolid*, substitute the file pool ID you specified in the DMSPARMS file. For *blks*, specify the number of blocks the administration machine will be allowed to use. For *storgrp*, specify a storage group number to which you have assigned a minidisk. (See [“Enrolling SFS Users and Creating BFS File Spaces”](#) on page 71.)

Now access the top directory as file mode A by entering the following command:

```
access
```

## Step 14: Create a PROFILE EXEC

---

Now create a PROFILE EXEC for the administration machine if none exists. No special statements are needed. You can create a PROFILE EXEC similar to those used by other virtual machines. The instructions in this manual assume you use the Restructured Extended Executor (REXX) language for the PROFILE EXECs of administration machines.

To prevent losing server console output, you might want to consider spooling the server's console to the user ID of the secondary user. Add the following to the server's PROFILE EXEC:

```
CP SPOOL CONSOLE START TO secondid
```

where:

***secondid***

is the user ID of the server's secondary user.

When the console is closed by issuing the SPOOL CONSOLE CLOSE command, a reader file is sent to *secondid*. The reader file contains a copy of the server's console because the last time a SPOOL CONSOLE CLOSE command was entered. Now, you can view or print the reader file for a closer examination of the console output.

Consider automating the issuing of the SPOOL CONSOLE CLOSE command, by adding the following command to the PROFILE EXEC of the secondary user:

```
CP SEND serverid #CP SPOOL CONSOLE CLOSE
```

where:

***serverid***

is the user ID of the server.

Every time the secondary user ID is logged on and its PROFILE EXEC is run, the SPOOL CONSOLE command is entered. The server sends a reader file that contains a copy of all the server's console output since the last time the secondary user logged on.

## Step 15: Consider Enrolling PUBLIC in the File Pool

---

For dedicated CRR recovery or FIFO servers, skip this step.

For repository file pool servers, proceed with this step.

At this time you might consider whether you want to enroll PUBLIC in the file pool. This allows the broadest range of users to *connect* to the server machine. *Connect* simply means the server will let the user communicate with it. For a description of enrolling PUBLIC, and the security implications of doing so, see [“Enrolling PUBLIC” on page 92](#).

## Step 16: Enroll the Initial Set of File Pool Users

---

For dedicated CRR recovery or FIFO servers, skip this step.

For repository file pool servers, proceed with this step.

Now enroll your initial set of file pool users. Refer to the instructions in [“Enrolling SFS Users and Creating BFS File Spaces” on page 71](#). When you have enrolled the initial set of users, remember to return to Step 17 in this chapter.

## Step 17: Set Up the Server for Automatic Starting

---

You can have the server started in multiple user mode every time z/VM itself is started. This step is optional and can be done any time after the file pool is generated. These instructions assume an AUTOLOG1 virtual machine exists on your z/VM system and you know the password to the machine. For automatic starting, do the following:



1. Log on or reconnect to the server machine if you are not already logged on to it.
2. Stop multiple user mode processing by issuing the server STOP NOBACKUP operator command.
3. Make a copy of the current PROFILE EXEC.

You should make a copy of the PROFILE EXEC as it currently exists. This exec is needed because sometimes it is necessary to log on the server and inhibit the execution of the usual PROFILE EXEC, which will be modified to automatically start the server. In these instances, it is handy to have an exec that contains everything in the PROFILE EXEC except the command to start multiple user mode processing. You would process the exec so the server machine would have access to the minidisks on which the FILESERV commands reside.

In the step-by-step instructions in other parts of this manual, it is assumed you have such an exec with a file name of SETUP. To create the SETUP EXEC enter:

```
copyfile profile exec a setup = =
```

Naturally, you can name the exec anything you like, just remember to substitute your name whenever the instructions ask that you process the SETUP EXEC.

4. Edit the PROFILE EXEC. Add the following commands near the beginning of the PROFILE EXEC:

```
"set autoread off"
"cp set run on"
```

Then add the following command to the end of the PROFILE EXEC of the server machine:

```
"exec fileserv start"
```

It should be the last record in the PROFILE EXEC.

5. Resume multiple user mode processing, if you wish, by entering:

```
fileserv start
```

6. Disconnect from the server machine by entering:

```
#cp disc
```

7. Log on (or reconnect to) the AUTOLOG1 virtual machine.
8. Add the following line to the PROFILE EXEC of the AUTOLOG1 machine:

```
CP AUTOLOG FPSERV1 password
```

In this example, FPSERV1 is the user ID of the server machine. Substitute the password of the server machine for *password*.

9. Log off or disconnect from the AUTOLOG1 user ID.

When z/VM is started, the virtual machine AUTOLOG1 is automatically logged on if it exists. When the PROFILE EXEC of AUTOLOG1 processes, the CP AUTOLOG command will log on the server machine. The z/VM system directory entry for a server machine contains an IPL CMS control statement, so CMS is automatically loaded when the server is logged on. Because additional data was not specified on the AUTOLOG command, the first read to the virtual console (the VM READ immediately after IPL CMS) is automatically satisfied. The PROFILE EXEC is then processed. The PROFILE EXEC of the server contains a FILESERV START command, so multiple user mode is immediately started.

**Note:** Do not specify the AUTOOCR initialization parameter for CMS in the server machine's IPL CMS control statement. If AUTOOCR is specified, you will not be able to inhibit the execution of the PROFILE EXEC when you log on the server machine to do dedicated maintenance mode work.

## Step 18: Copy the Sample Files to the New File Pool (SFS only)

For CRR recovery servers, skip this step.

## Copying the Sample Files

For SFS repository file pool servers, proceed with this step.

*z/VM: CMS User's Guide* and *z/VM: CMS Primer* contain exercises that require the sharing of IBM-supplied sample files. For file pool users to successfully complete the exercises, you must place the sample files in the file pool.

During installation of z/VM, if you chose to install the VMSYSU file pool, the sample files were loaded in the VMSYSU:MAINT.SAMPLES directory. Otherwise, the sample files can be found on the MAINT 193 minidisk. Read authority is granted to PUBLIC at that time and PUBLIC is also enrolled in the file pool.

To make the samples available in your file pool, do the following:

1. If you are not already logged on an administration machine, log on to one now.
2. Enroll the MAINT user ID in your file pool. MAINT must be the owner of the sample files for the exercises in the manuals to work. Specify at least 1000 blocks of storage, as indicated below. You can enroll MAINT in any valid storage group in your file pool. For example, you might enter:

```
enroll user maint newfp (blocks 1000
```

Substitute the ID of your file pool for NEWFP.

3. Now create directories for user ID MAINT in which the sample files will reside. The directory structure must be created as described here for the exercises in the manuals to work. Enter the following commands:

```
create directory newfp:maint.fishing
create directory newfp:maint.feline
create directory newfp:maint.sample
```

4. Copy the sample files into the directories just created.

If you installed VMSYSU, the sample files are in the VMSYSU:MAINT.SAMPLES directory and enter this command:

```
acc vmsysu:maint.samples b
```

If you did not install VMSYSU, the sample files are in the MAINT 193 minidisk. Assuming you have linked MAINT 193 as 193 as is shown in [Figure 35 on page 257](#), enter this command:

```
acc 193 b
```

**Note:** If you have linked MAINT 193 as something other than 193, substitute the address for 193 in the previous ACCESS command.

The rest of the steps are identical whether you copied the sample files from VMSYSU or MAINT 193 minidisk:

```
acc newfp:maint. c
copy * script b = = c
relocate gifts script c to newfp:maint.sample

acc newfp:maint.fishing c
copy deep sea b = = c

acc newfp:maint.feline c
copy cat tales b = = c
```

5. Finally, grant read authority on these files and directories to PUBLIC:

```
grant authority newfp:maint. to public
grant authority newfp:maint.fishing to public
grant authority newfp:maint.feline to public
grant authority newfp:maint.sample to public

grant authority * * newfp:maint. to public
grant authority * * newfp:maint.fishing to public
grant authority * * newfp:maint.feline to public
grant authority * * newfp:maint.sample to public
```

If you plan to generate many file pools for your installation, consider placing the above commands in a exec so you do not need to retype them.

## Step 19: Adjust Cache Size (SFS only)

---

The CMS Shared File System cache defaults to 20 KB for SFS files. This is a good choice for systems that have moderate paging rates. If your system has an especially high paging rate, you may want to consider setting this to a lower value. If your system has a low paging rate, increasing the size of the file cache can probably improve performance.

To change the SFS cache size, you need to update the BUFFSIZ parameter in the DEFNUC macro, which is in DMSNGP ASSEMBLE (DMSZNGP ASSEMBLE for z/Architecture CMS). Then assemble the file and rebuild the CMS nucleus. See *z/VM: CP Planning and Administration* for information about DMSNGP, DMSZNGP, and DEFNUC. See *z/VM: Service Guide* for instructions on rebuilding the CMS nucleus.

**Note:**

You have completed file pool and server generation.

For repository file pool servers, soon after your file pool is in use, you should back up **all** storage groups that contain user data. This helps protect your file pool from loss. You should also develop a schedule for backing up the control data as well as the user data. See [Chapter 7, “Recovery Procedures,” on page 101](#) for more about file pool recovery.

For dedicated CRR recovery or FIFO servers, IBM does not recommend you back up control data or user data. See [“Replace Minidisks in CRR File Pool” on page 327](#) for more about replacing CRR minidisks.



---

## Chapter 16. Deleting a File Pool

This chapter applies to all types of file pools, except where specifically noted.

To delete a file pool from your system, do the following:

1. Inform users of impending deletion (not applicable to CRR or FIFO file pools).

They should have ample time to move data elsewhere. They may also need to change profiles, application programs, or execs. Do not forget to inform remote users of the file pool, if any.

2. Ensure the CRR recovery server log name tables remain current by erasing the entry which relates to the file pool. See [“CRR Log Name Table Management” on page 321](#) for further information.
3. Update IPL statements in user directories (not applicable to CRR file pools).

After everyone is prepared, remove the FILEPOOL operand from the IPL control statement in the z/VM system directory entries of the file pool users.

4. Format all the file pool minidisks.

To avoid any possibility of future unauthorized access to file pool data when the minidisks are reassigned, enter a CMS FORMAT command for every file pool minidisk.

5. Remove the AUTOLOG1 entry.

Most servers are set up so they are automatically logged on when z/VM is started. To undo this, remove the AUTOLOG command for the appropriate server machine from the PROFILE EXEC of the AUTOLOG1 virtual machine.

6. If your file pool is available to other processors, your installation may have created a communications directory. If so, update the communications directory (SCOMDIR NAMES). Then enter the SET COMDIR RELOAD command to put the changes in effect. You should inform remote administrators that they should also update their communications directories. For more information about the SET COMDIR command see *z/VM: CMS Commands and Utilities Reference*. For more about the use of communications directories see *z/VM: Connectivity*.
7. If you used an external security manager for the file pool, update the authorizations of that security manager, as appropriate (not applicable to CRR file pools).

8. Delete server z/VM system directory entries.

Now delete the z/VM system directory entry for the server machine that owned the file pool minidisks. Note the physical allocations on the MDISK statements. These minidisks can now be used for other purposes.

9. Eliminate any tape backups (not applicable to dedicated CRR or FIFO file pools).

Optionally destroy any tape backups of the file pool contents.



---

## Chapter 17. Participation in CRR (SFS only)

This chapter describes how SFS participates in CRR and the steps involved for an SFS operator to manually complete a transaction.

### Overview

---

The Coordinated Resource Recovery (CRR) facility allows the applications on your system to participate in coordinated transactions, that is, transactions that write data to more than one participating resource on a CMS work unit. A CMS work unit can consist of a series of related actions (a logical unit of work) whose changes are to be treated as a single update. Each group of related actions is called a coordinated transaction. Within a coordinated transaction, data may be written to multiple file pools. These file pools can be on the same processor, or distributed among processors within a TSAF collection, CS collection, or within a SNA network. Distributed applications, in cooperation with CRR, use protected conversations to ensure distributed applications can update their resources with integrity. TSAF and CS collections do not support protected conversations. The changes made to the multiple SFS file pools are managed by CRR, which ensures all the SFS file pools involved in the transaction either commit or roll back (sometimes called back out) the work in unison. Other resources, besides SFS file pools, may also participate in CRR. For more information on CRR, see [Chapter 18, “CRR Overview,” on page 287](#).

One part of how SFS participates in CRR is by supporting the two-phase commit protocol for synchronization (sync) point requests. If an error occurs during the second phase of sync point processing, SFS may participate in resynchronization by setting aside the logical units of work involved in the sync point processing (waiting for the second phase of the two phase commit). Those logical units of work are identified as **prepared work**. Prepared work can be either prepared-and-connected work or prepared-and-not-connected work. Prepared-and-connected work has a communication path from the user's virtual machine to the file pool server, whereas the prepared-and-not-connected work has lost its communication path from the user's virtual machine to the SFS file pool server. The prepared-and-not-connected work requires resynchronization processing. When the resource becomes available, resynchronization completes the transaction. See [“Resynchronization Function” on page 305](#) for more information on the resynchronization function.

### Participation Commands

---

The following SFS operator commands have been created for SFS's participation in CRR:

- QUERY PREPARED
- FORCE PREPARED
- ERASE LUNAME
- QUERY LOGTABLE

### Administrator and Operator Intervention

---

There may be times when the file pool operator will need to intervene and manually synchronize the prepared-and-not-connected SFS work. (An operator is anyone that can enter file pool server operator commands.) For example, if communication links are down, or if, for some reason, there will be a long delay before resynchronization, users will not be able to access files or directories that are locked because of the prepared-and-not-connected work. (SFS holds all locks for work that is prepared-and-not-connected.) Then the file pool operator will have to intervene and manually synchronize the prepared-and-not-connected work.

You may be alerted to a problem with a coordinated transaction if the file pool server goes down. After you restart it through the usual procedures, users may complain they cannot access their data. This may also occur even if the file pool server has not system malfunctioned; for example, an application

## Intervention

system malfunction, system malfunction, or communications error can all result in files or directories being inaccessible.

To determine which work requires resynchronization and to manually synchronize it, you need to complete the following steps:

### Step 1

Determine the work that is prepared-and-not-connected (see [Step 1](#)).

### Step 2

Identify the appropriate CRR recovery server (see [Step 2](#)).

### Step 3

Contact the CRR recovery server operator (see [Step 3](#)).

### Step 4

Determine if heuristic action is necessary (see [Step 4](#)).

### Step 5

If action is required, commit or roll back (back out) the prepared-and-not-connected work (see [Step 5](#)).

### Step 6

Determine if any further action is required (see [Step 6](#)).

## Step 1: Determine Work That Is Prepared-and-Not-Connected

To determine the work that is prepared-and-not-connected, get the recovery token and file pool ID from the user experiencing a problem. This recovery token should have been received by the user in one of these ways:

- In message DMS2026E if the user was issuing CMS commands
- From the application the user was executing

For example, if Andy calls you explaining he is unable to access his data and that he received message DMS2026E, ask him for the file pool ID and recovery token provided by the message. The existence of a recovery token indicates resynchronization activity is pending. With the recovery token and the file pool ID you can move on to the next step.

**Note:** It is also possible for a user to get message DMS1138E instead of message DMS2026E. This can happen, for example, when commands are used for a directory that contains a file accessed on a logical unit of work that is prepared-and-not-connected. A number of commands can result in this situation; such as commands to:

- Revoke authority
- Rename
- Relocate
- Create a lock
- Erase a directory with the FILES option

In this case, the user would not have received the recovery token.

The user's application can also supply the recovery token. If the application failed with a reason code of 71800, it can obtain work unit extended error information that contains the recovery token. The error information is returned to the application in the *wuerror* parameter (FPEAUGMT field) of CSL routines and the recovery token can be obtained by using the DMSWUERR CSL routine (ERROR\_REASCODE\_INFO parameter). Then the application programmer can provide the recovery token and file pool ID to the user.

## Step 2: Identify the Appropriate CRR Recovery Server

After you have determined the user's work is delayed because of pending resynchronization activity, you need to get further information from the file pool server (identified in Step 1) in order to identify the CRR recovery server responsible for completing the work involved in the delay.



Enter:

```
query prepared for all
```

on the operator console of the affected file pool. For a complete description, see “QUERY PREPARED” on page 660. Look through the display until you find a recovery token matching the one supplied by the user (or to find those whose state is prepared-and-not-connected, if the user received message DMS1138E). This identifies the logical unit of work that is prepared-and-not-connected.

**Note:** At this point, it is possible you might not find an SFS logical unit of work whose state is prepared-and-not-connected (or cannot find a matching recovery token). If so, the problem has gone away by means of usual automatic periodic retry of resynchronization.

The output of the QUERY PREPARED FOR ALL command might look something like this:

```
Time: hh:mm:ss          QUERY PREPARED - VMSYSP1
Date: mm/dd/yy

Access Userid DBRISEE          Time:
hh:mm:ss                      Taskid: 4
                               LUWID: VM1.SERP2.0001A004774B.98CA
                               LUNAME: *LOCAL
                               TPN: .SERVP2
Recovery Token: 00002507
                               State: PREPARED AND NOT CONNECTED

Transaction Tag Payroll Run 123
```

In this example output, the display contains the information about the prepared-and-not-connected work. You could also see information about prepared-and-connected work and for work that was forced committed and forced backout in the output of the QUERY PREPARED command.

By matching the recovery token, you can determine who has a lock on the resource. In this case, Andy cannot access his data because user DBRISEE has a lock on the file or directory. DBRISEE cannot free the lock because the logical unit of work is prepared-and-not-connected. This means an error occurred while the work was in process, and now resynchronization is required. You should save the recovery token for use in [“Step 3: Contact the CRR Recovery Server Operator” on page 281](#) and [“Step 4: Determine If Heuristic Action is Necessary” on page 282](#).

The LUWID identifies the logical unit of work as it is known to the CRR recovery server. The combination of the LUNAME and TPN identifies the CRR recovery server that is responsible for the work involved in the delay. In this example, the LUNAME is identified as \*LOCAL, indicating the CRR recovery server is on the same system as the SFS file pool server. You should save the LUWID for use in [“Step 3: Contact the CRR Recovery Server Operator” on page 281](#) and [“Step 4: Determine If Heuristic Action is Necessary” on page 282](#).

Also of interest is the transaction tag, which is optionally provided by the application to aid in recovery from an error. If a transaction tag is displayed in the QUERY PREPARED command output, make a note of it in case you need to refer to it in [“Step 4: Determine If Heuristic Action is Necessary” on page 282](#). The transaction tag could contain a file name that describes SFS and CRR operator recovery procedures to be followed in case of an error.

You should also save the TASKID for use in [“Step 5: If Action Is Required, Commit or Roll Back” on page 282](#) and save the LUNAME and TPN for possible use in [“Step 6: Determine If Any Further Action is Required” on page 282](#).

### Step 3: Contact the CRR Recovery Server Operator

Now you know which CRR recovery server is responsible for completing the work involved in the delay, you can contact the CRR operator, who can then enter CRR queries to determine the system status of the problem. For more information on what the CRR operator does to evaluate the system status of a transaction, see [“Problem Management” on page 330](#).

### Step 4: Determine If Heuristic Action is Necessary

Normally, resynchronization will resume and commit or roll back the prepared transaction. However, if resynchronization will be delayed because of a serious error, or if the prepared-and-not-connected SFS resources are critical, heuristic action (manual intervention) may be necessary.

The SFS operator, CRR operator, and the user whose work is prepared-and-not-connected will need to make this decision based on output from several commands and other information about the circumstances of the error. The CRR operator enters the CRR QUERY LU and CRR QUERY LUWID commands to obtain information about the status of the error.

If heuristic action is not necessary, this task is complete. You may notify the user an error has occurred and processing will be temporarily delayed.

The CRR operator may have determined, in the previous step, there is other prepared-and-not-connected work on other systems that are part of this same transaction. The CRR operator might be able to enter the CRR RESYNC command to take heuristic action to complete this transaction on multiple systems. See [“Problem Management” on page 330](#) for more information about using the CRR RESYNC command.

If you determine manual intervention by the SFS operator is necessary, continue with the next step.

### Step 5: If Action Is Required, Commit or Roll Back

Having determined heuristic action is necessary, you must now decide whether to commit or roll back the prepared-and-not-connected work. You and the CRR recovery server operator should decide what to do based on established procedures and the information available from the system.

Because you have already entered a QUERY PREPARED command, you will know the task ID to use as input to the FORCE PREPARED command. After your discussion with the CRR recovery server operator, you will also have determined whether the work should be committed or backed out.

To manually commit or roll back the prepared-and-not-connected work, you will have to use the FORCE command with the PREPARED operand.

For a complete description, see [“FORCE” on page 539](#).

For example if you had determined the prepared-and-not-connected work was crucial and the logical unit of work currently locked by DBRISEE should be committed, you would use the task ID, 4, which was included in the output of the QUERY PREPARED command, and enter the following command:

```
force prepared 4 commit
```

This command would commit the logical unit of work that was prepared-and-not-connected. A record of this heuristic action is saved on the file pool logs. After the command processes, DBRISEE's locks are freed, and you can notify Andy he can continue with his work.

### Step 6: Determine If Any Further Action is Required

Now that you have heuristically resolved the prepared-and-not-connected work, when resynchronization resumes, the transaction will be completed.

In most instances, the task will now be complete. When resynchronization resumes, the record of this heuristic action for the LU name and TPN pair will be processed through the system.

However, if your heuristic decision conflicts with the resynchronization decision, it is likely the data will be inconsistent. For example, if your heuristic decision was to use the FORCE PREPARED command to commit the logical unit of work and resynchronization wants to roll back the logical unit of work, you have heuristic damage. If this is the case, the CRR recovery server operator will receive an error message indicating there is a possible inconsistency because of heuristic damage. The owner of the file in question must be notified so he can attempt to manually rebuild the file to reflect the results of the resynchronization decision and restore consistency of the data. The transaction tag displayed (if one is available from the application) in the QUERY PREPARED command's output may help you determine who owns the file.

In rare instances, when a serious error has occurred and resynchronization will not resume for this logical unit of work (for instance, because both CRR logs have been lost, or because a CRR recovery server operator entered a CRR ERASE LUWID), you may want to manually purge the record of the forced transaction associated with the LU name and TPN pair from the file pool logs. This would clean up the file pool logs, in case resynchronization never has to process this logical unit of work again. To purge the record of the forced transaction from the SFS logs, you would use the ERASE LUNAME command.



**CAUTION:** This ERASE LUNAME command should be used with extreme caution because it will erase the record of any heuristic force performed for this logical unit of work.

If you determine you can enter the ERASE LUNAME command on the file pool logs, use the following format:

```
erase luname *local .servp2
```

where

**\*local**

is obtained from the LUNAME field found in the output of the QUERY PREPARED command

**.servp2**

is obtained from the TPN field found in the output of the QUERY PREPARED command

The QUERY PREPARED command was entered in [“Step 2: Identify the Appropriate CRR Recovery Server”](#) on page 280.

For a complete description, see [“ERASE LUNAME”](#) on page 427.

At this point, the task is complete. All record of this coordinated transaction will be erased from the system when resynchronization resumes.

## SFS Log Name Table Changes

Additions to the file pool log name table occur automatically when new participants are recognized or when old participants change their log names or communications characteristics, obsoleting previous entries. Occasionally you may consider deleting obsolete entries to make room for additional space in the table.

A file pool log name table entry contains a pair of fully qualified LU names for each current or past CRR recovery server known to the file pool, along with the recovery server log name and recovery server TPN. The fully qualified names consist of a:

- *Local* fully qualified LU name, representing the local source of SNA network communications
- *Remote* fully qualified LU name, representing the target for SNA network communications

For local communications (within a single system, TSAF or CS collection) there are no fully qualified LU names. Therefore the display of the table entries indicate **\*LOCAL** for these fields.

Whenever an SFS user initiates communications with a file pool, there is an initial exchange of log names between the file pool and the recovery server at the originating application node. This ensures the file pool and the CRR recovery server each have consistent log data about the units of work that might require resynchronization. Both the file pool and the recovery server record their communication partner's log name in their own log name table, identified by LU names and TPNs as described above.

Normally there is only one log name table entry per participant. However, changing the items in the following list may cause residual, obsolete entries in the log name table:

- SNA network IDs (first part of the fully qualified LU name)
- Gateway names (second part of the fully qualified LU name)
- SFS file pool IDs
- CRR recovery server startup parameter (fully qualified) LU name which is the basis for formulating the recovery server's identifying TPN
- From TSAF or ISFC to SNA communications, or from SNA communications to TSAF or ISFC

Although new entries are created automatically when changes occur, obsolete entries are not deleted automatically. An accumulation of obsolete entries may cause insufficient free space in the table; therefore it is recommended you purge obsolete entries as changes are made.

In order to maintain the integrity of the log name tables and avoid CRR error conditions, it is very important changes such as those listed previously be done properly. To do it properly, use the following procedures:

- When changing identifiers such as file pool IDs and gateways (changing gateways affects fully qualified LU names for SNA network participants) be certain not to select names that already exist in the name space. For file pool IDs the name space is a TSAF or CS collection; for gateways it is the SNA network.
- Where the change is SFS originated, such as a change in the file pool identifier, use the QUERY LOGTABLE operator command to identify all recovery servers affected by the change, and notify the appropriate administrators to clean up their log name entries accordingly. Where the change is recovery server originated, the file pool administrator is notified by the CRR recovery server administrator. For more information, see [“QUERY LOGTABLE” on page 658](#).
- Avoid inconsistencies by making changes only after quiescing CRR activity. If the change is file pool originated, such as changing the file pool identifier, CRR activity is usually quiesced as a byproduct of the step below, involving the erasing of the obsolete log name table entry. When there is outstanding CRR work, the attempt to do this erase fails. When the outstanding work completes, the erase can be completed. It is probable that changes are CRR application user or recovery server originated, such as a change in communications path to the file pool. In this case the quiesce is accomplished by the CRR administrator by stopping the associated recovery server. If a change is made during usual CRR activity, you could encounter inconsistencies in log name entries.
- Clean up obsolete log name entries for the file pool using the ERASE LUNAME operator command. For more information, see [“ERASE LUNAME” on page 427](#).

**Note:** Although SNA network identifiers (first part of the fully qualified LU names) would not usually change, in this case you should erase log name entries **before** the network change takes place. These entries are those that include fully qualified LU names based on the old network identifiers.

See [“CRR Log Name Table Management” on page 321](#) for an example of changing the gateway used to communicate with a CRR participant. SFS commands for log name table query and erase are similar to those used for recovery server log name table maintenance, which are described in [“CRR Log Name Table Management” on page 321](#). The file pool is one example of a participant that could be affected by such a change.

For changing the file pool identifier, see [“Changing the File Pool ID” on page 67](#).

---

## Part 2. Administering Coordinated Resource Recovery (SFS only)

This part describes an overview of the administration and operation of the Coordinated Resource Recovery facility. This is explained in the following chapters:

- [Chapter 18, “CRR Overview,” on page 287](#)
- [Chapter 19, “CRR Administration,” on page 315](#)

<b>Note:</b>
This information only applies to SFS file pool servers. It does not apply to the Byte File System (BFS).



## Chapter 18. CRR Overview

The Coordinated Resource Recovery (CRR) facility ensures data integrity among participating resource managers and distributed applications.

### What Is CRR?

All updates to data repositories (hereafter called resources) within a transaction, are either done (committed) or not done (rolled back<sup>7</sup>). SFS file pools, starting with z/VM, are an example of a resource that participates in CRR. Other resources (or products) may also participate in CRR.

CRR coordinates the commit or backout of updates of protected resources whether the resources are within the same processor, or distributed among processors within a TSAF or CS collection or SNA network. Distributed application programs, in cooperation with CRR, use protected conversations to ensure distributed transaction resource integrity.

**Note:** TSAF or CS collections do not support protected conversations.

The CRR functions are:

- Coordination
- Resynchronization
- Logging

The coordination function, handled by the synchronization point manager (SPM), resides in CMS in the application program's virtual machine. The resynchronization function, handled by the resynchronization manager, and the logging function, handled by the logging manager, reside in a separate virtual machine called the CRR recovery server, which is also a part of CMS.

### Banking Example

A common banking transaction of transferring money from a savings account to a checking account serves as a good example of how CRR could be used. In this example, assume the savings and checking accounts are in different resources (or data repositories).

First, let us assume the resources are not participating in CRR. If there was a system error during the middle of a transaction that was transferring money from the savings to checking account, it is possible the money could have been debited from the savings account and never credited to the checking account. The result is a very unhappy customer.

Now, let us assume the resources are participating in CRR. If there was a system error during the middle of that same transaction, depending on the time of the error, CRR would ensure one of the following happens to both of the customer's accounts:

- Savings is debited and checking is credited, or
- Savings is not debited and checking is not credited.

But, CRR ensures that we do not have the case where the savings is debited and the checking is not credited.

<sup>7</sup> The term rollback (verb form roll back) is generally used in CMS. The synonymous term in Systems Application Architecture® (SAA) and SNA is backout (verb form back out). Because of the relationship between CRR and SAA and SNA, the SAA and SNA term backout does appear in CMS documentation along with rollback.

### SFS Example

This is an example of how SFS users could benefit from CRR. Assume you had an application that updates two files in the same SFS file pool. Then the SFS administrator moves one of the files to a different SFS file pool. CRR allows the application to still update the two files, even though they are now in different SFS file pools.

## What Are the CRR Tasks?

---

There are several categories of tasks data processing professionals could be involved in with CRR, they are:

- Administration and Operation
- Application development
- Product participation

### Administration and Operation

The CRR operator is the individual who has access to the CRR recovery server's console and can enter the CRR recovery server operator command and the file pool machine command. The CRR operator might manage a single CRR recovery server, or manage many CRR recovery servers from a central site using NetView®. Some of the tasks a CRR operator could perform are:

- Log management
- Problem management
- Resynchronization management
- Generate a CRR recovery server
- Audit operator commands
- Generate accounting information
- Monitor performance

You could also perform some administration tasks, such as enrolling a user as an administrator of a CRR recovery server. Probably, you would have this person act as administrator of all your CRR recovery servers and monitor their status using the QUERY FILEPOOL REPORT command or the QUERY FILEPOOL CRR command. For more information, see [“Set Up Remote User Machine Administration of CRR Recovery Server” on page 326.](#)

### Application Development

CRR provides routines for the application programmer to use in designing applications that take advantage of CRR. The SRRCMIT SAA resource recovery (also known as CPI resource recovery) routine or the DMSCOMM CSL routine can be used to commit changes and the SRRBACK SAA resource recovery routine or the DMSROLLB CSL routine can be used to back out (or roll back) changes. For more information, see [z/VM: CMS Application Development Guide](#), [z/VM: CMS Callable Services Reference](#), and [Common Programming Interface Resource Recovery Reference](#).

### Product Participation

The product developer, who is responsible for writing code to get an IBM or non-IBM resource to participate in CRR, is interested in understanding:

- How to write a resource adapter that registers the resource with the SPM.
- How to write resource adapter exits for precoordination, coordination, postcoordination, end of work unit, and backout required actions that are driven by the SPM.
- How the resource manager and CRR recovery server exchange log names and compare states during resynchronization initialization and resynchronization recovery.



For more information, see [z/VM: CMS Application Development Guide](#) and [z/VM: CMS Callable Services Reference](#).

The product developer should also be interested in the SNA LU 6.2 sync point architecture. CRR is z/VM's implementation of the SNA LU 6.2 sync point architecture. The SNA LU 6.2 sync point architecture describes the basic sync point architecture and several options. In the SNA LU 6.2 sync point architecture, CRR implements:

- Sync point base support (option set S.1)
- Enhancements and fixes to the sync point base support (option set S.4)

See *LU 6.2 Reference: Peer Protocols* for more information about the SNA LU 6.2 sync point architecture.

## Introduction to CRR Terminology

---

This section describes the terminology frequently used with CRR.

### What Are Distributed Resources and Protected Resources?

A *resource* is data controlled by a resource manager. For example, an SFS file pool is a resource controlled by the SFS resource manager (also called SFS file pool server). In z/VM, if the resource is on the same processor as the application program that is updating the resource, it is called a *local resource*. (SNA defines *local* resources differently, as resources that share the same log. z/VM has no SNA local resources.) If the resource is on a different processor (such as in a TSAF or CS collection or in a SNA network) than the application program, it is a *distributed resource*.

A *protected resource* identifies a resource that is participating in CRR. (SNA uses protected resource as an umbrella term that includes both protected resources and protected conversations.) If an application updates a protected resource, CRR ensures the updates are consistent with updates to other protected resources within the transaction.

### What Are Protected Conversations?

A *conversation* is an APPC connection between two programs that lets them communicate with each other.

A *protected conversation* is a conversation usually between two distributed application programs. The changes done by the distributed application that started the protected conversation and the changes done by the distributed application that accepted the protected conversation are all protected by CRR and are treated as a single coordinated transaction.

Applications can begin protected conversations using these interfaces:

- CPI Communications (also known as SAA communications interface) - The application program would enter the following CPI Communications calls:
  1. CMINIT - Initializes the conversation
  2. CMSSL - Sets the *sync\_level* to CM\_SYNC\_POINT
  3. CMALLC - Allocates the conversation with the partner

For more information, see [CPI Communications Reference](#).

- APPC/VM - The application program could use the CONNECT function and set the SYNCLVL equal to SYNCPT. For more information, see [z/VM: CP Programming Services](#).

CPI Communications is preferred, because it makes your application more portable.

#### Note:

1. CP supports protected conversations between two programs on the same processor.
2. AVS and VTAM support protected conversations between two programs in a SNA network.
3. TSAF and ISFC do *not* support protected conversations between two programs in a TSAF or CS collection.

4. Applications begin protected conversations to tell other distributed application programs to update protected resources, but applications generally do not begin protected conversations to actually update the protected resources. For updating protected resources, applications typically call a high-level product interface. The high-level product interface starts the conversations (usually with the sync level option set to NONE or CONFIRM) to the product's resource manager for updating the protected resource. An example of this kind of interface is the interface between resource adapters and resource managers as described in “CRR Functions” on page 294. But, there may be occasions where a resource adapter uses a protected conversation to communicate with a participating resource manager. For more information about how resource adapters and participating resource managers use protected conversations, see *z/VM: CMS Application Development Guide*.

**Protected Resources in Same Processor**

Figure 36 on page 290 illustrates an application program updating two protected resources. The application program and the protected resources all reside on VM1. In this example, the CRR recovery server logs (during sync point processing) the two protected resources on VM1 that are being updated. (Actually, the application calls a high-level product interface and the resource manager's resource adapter does the actual updating of the protected resources.) CRR ensures the application can update both protected resources with integrity.

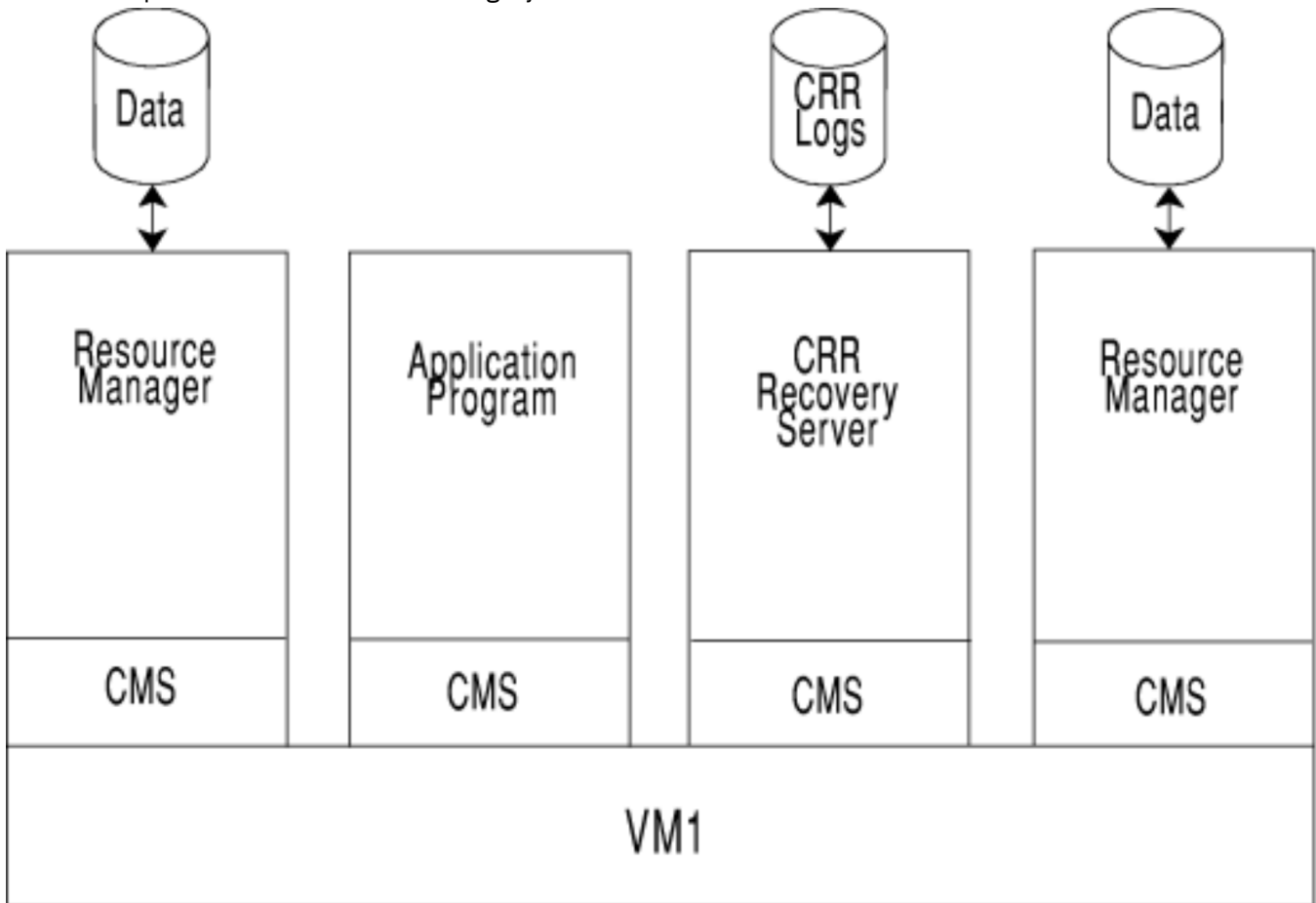


Figure 36. Protected Resources in Same Processor

**Protected Resources in a TSAF or CS Collection**

Figure 37 on page 291 illustrates an application program updating two protected resources in a TSAF or CS collection. The application program resides on VM1 and communicates through TSAF to update the protected resources that reside on VM2 and VM3. In this example, the CRR recovery server on VM1 logs (during sync point processing) the protected resources on VM2 and VM3 that are being updated. (Actually,

the application calls a high-level product interface and the resource manager's resource adapter does the actual updating of the protected resources.)

**Note:** The CRR recovery servers on VM2 and VM3 are not involved in this example, because there are no protected conversations started or accepted on VM2 or VM3. Also, there is no application program on VM2 or VM3 that is updating a protected resource, which would require CRR logging.

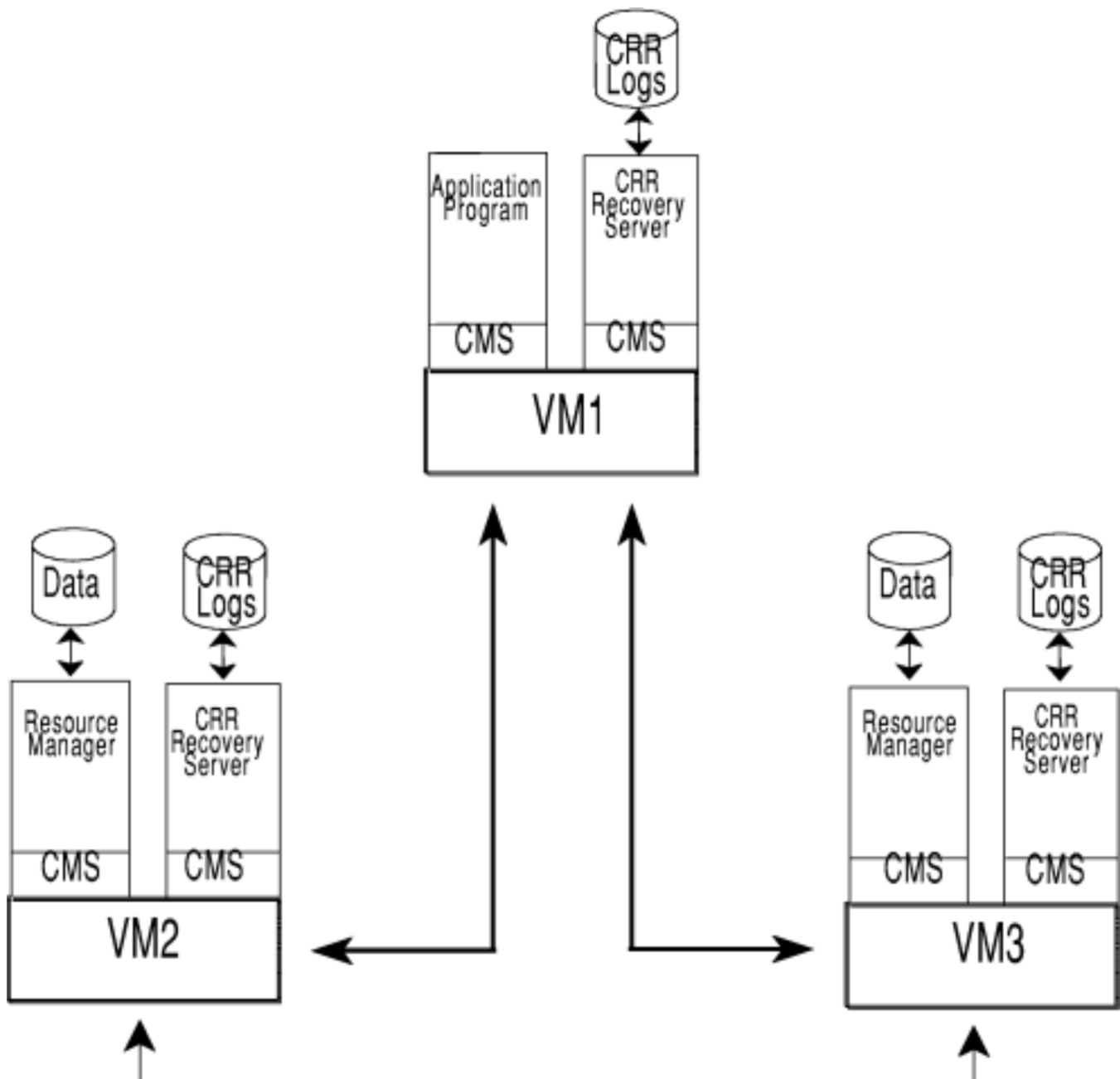


Figure 37. Protected Resources in a TSAF or CS Collection

### Protected Resources in a SNA Network

Figure 38 on page 292 illustrates an application updating two protected resources in a System Network Architecture (SNA) network. The application program resides on VM1 and communicates through GCS, AVS, and VTAM to update the protected resources that reside on VM2 and VM3. In this example, the CRR recovery server on VM1 logs (during sync point processing) the protected resources on VM2 and VM3 that are being updated. (Actually, the application calls a high-level interface and the resource manager's resource adapter does the actual updating of the protected resources.)

**Note:** The CRR recovery servers on VM2 and VM3 are not involved in this example, because there are no protected conversations started or accepted on VM2 or VM3. Also, there is no application program on VM2 or VM3 that is updating a protected resource, which would require CRR logging.

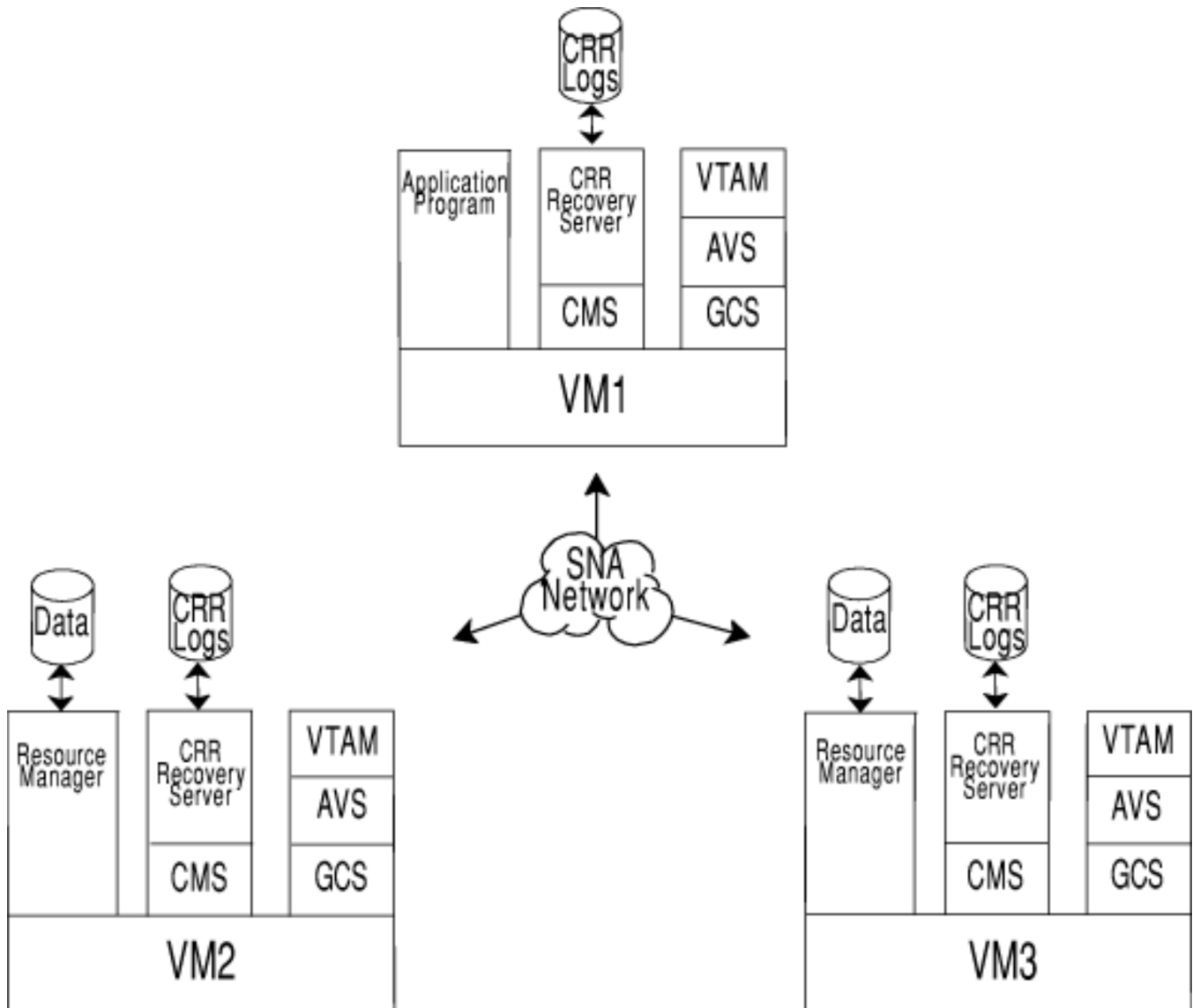


Figure 38. Protected Resources in SNA Network

## What Are Distributed Applications?

Distributed application programs have multiple parts that are on different virtual machines. The different virtual machines can be on the same or different systems. Distributed application programs, in cooperation with CRR, use protected conversations to ensure distributed transaction resource integrity.

Figure 39 on page 294 illustrates user's distributed application program that consists of three parts. Each part is distributed among three virtual machines. Part #1 communicates with part #2 by means of a protected conversation (PC #1 in the figure). Part #2 communicates with part #3 by means of a protected conversation (PC #2 in the figure). Also, there are two protected resources on both VM1 and VM2.

The sequence of events in this example are:

1. Distributed application program (part #1) updates (by means of the resource adapter) both protected resources on VM1.
2. Distributed application program (part #1) starts a protected conversation (shown as PC #1) to distributed application program (part #2) on VM1.

3. Distributed application program (part #2) updates (by means of the resource adapter) both protected resources on VM1.
4. Distributed application program (part #2) starts a protected conversation (shown as PC #2) to distributed application program (part #3) on VM2.
5. Distributed application program (part #3) updates (by means of the resource adapter) both protected resources on VM2.
6. Distributed application program (part #1) issues a commit.
7. The SPM in distributed application program's (part #1) virtual machine starts sync point processing by preparing the two protected resources on VM1 and tells distributed application program (part #2) to issue a commit.
8. Distributed application program (part #2) issues a commit.
9. The SPM in distributed application program's (part #2) virtual machine starts sync point processing by preparing the two protected resources on VM1 and tells distributed application program (part #3) to issue a commit.
10. Distributed application program (part #3) issues a commit.
11. The SPM in distributed application program's (part #3) virtual machine starts sync point processing by preparing the two protected resources on VM2.
12. CRR commits all the prepared work at all the protected resources and reports this result to all three parts of the distributed application program.

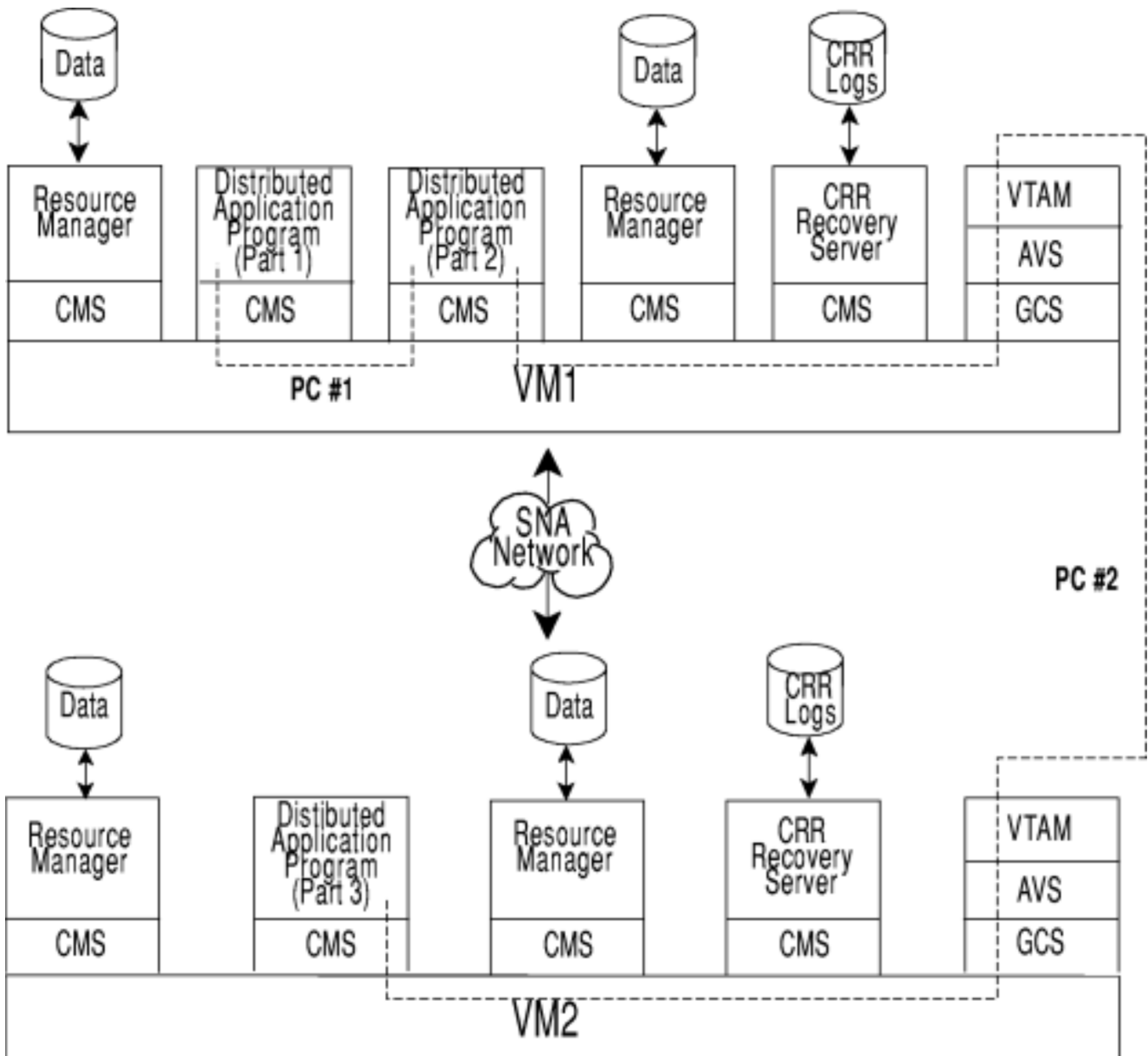


Figure 39. Protected Resources Using Protected Conversations in a SNA Network

## CRR Functions

Refer to [Figure 40 on page 295](#) to see how the CRR functions relate to each other.

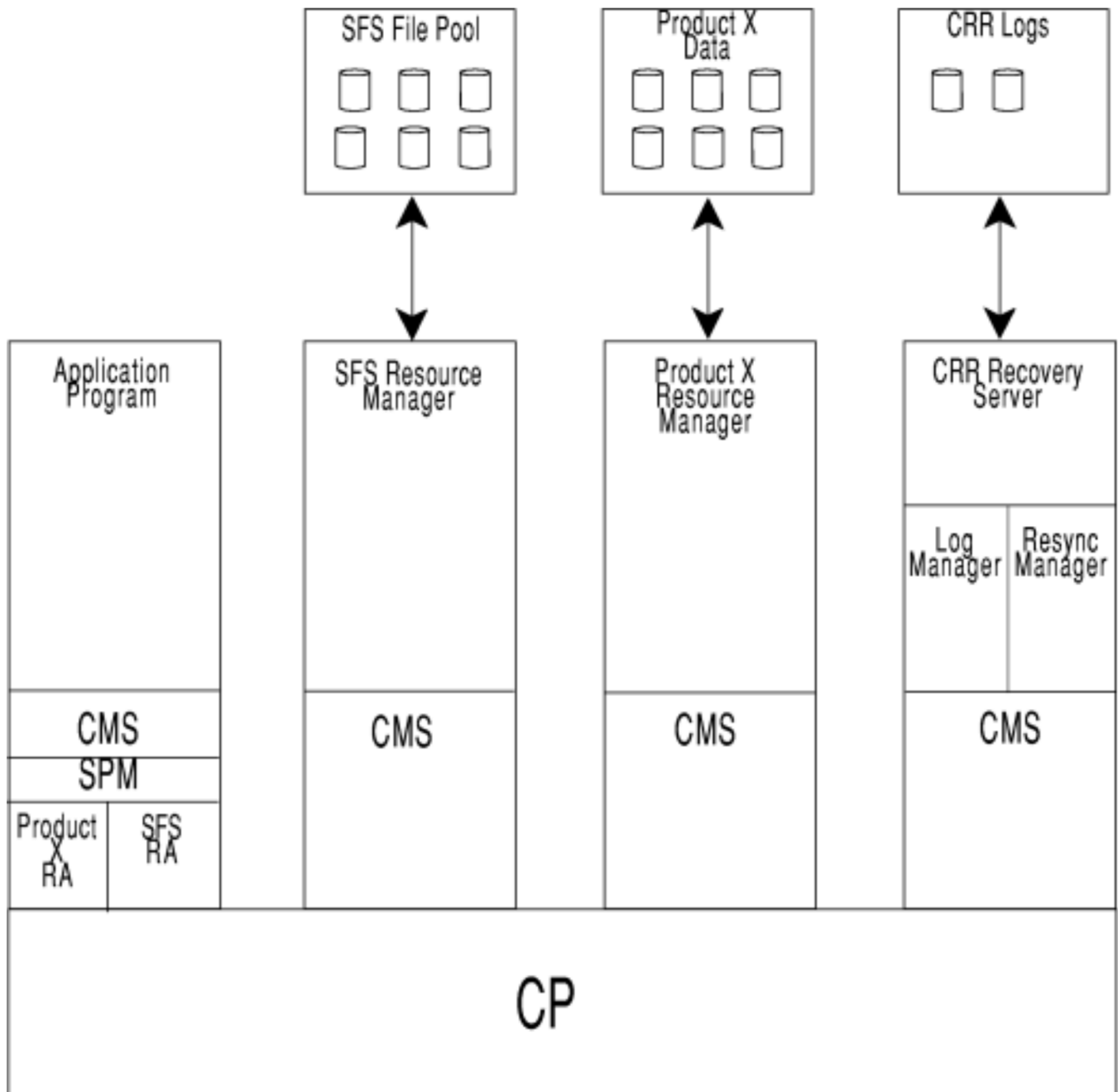


Figure 40. Coordinated Resource Recovery (CRR) Functions

This figure shows four virtual machines:

- Application program (runs on user machine CMS)
- SFS resource manager (CMS server machine)
- Product X resource manager (Part of product X)
- CRR recovery server (CMS server machine)

The user's virtual machine contains an application program. The application program wants to use the CRR facility to update two resources in the same CMS work unit. The two resources, in this example, are:

- SFS file pool managed by an SFS resource manager (or also called an SFS file pool server)
- Product X database managed by a Product X resource manager

The application program interfaces with the CMS synchronization point manager (SPM). The SPM is responsible for the coordination function (see [“Coordination Function”](#) on page 296). The SPM interfaces with the resource adapter, shown in the figure as RA, for each resource that participates in CRR. All user virtual machines include an SFS resource adapter within CMS, and in this example, product X has a resource adapter in the user's virtual machine too. The resource adapters register the resources with the SPM. The SPM drives exits to the resource adapters. The resource adapters interface with their corresponding resource managers, and the resource managers have to interface with the CRR recovery server and also support the two-phase commit protocol. For more information on the interfaces between the SPM and resource adapters, see [z/VM: CMS Application Development Guide](#).

The SFS resource manager and product X resource manager both must participate in CRR to allow the application to have coordinated updates of both resources.

**Note:** SFS always participates in CRR.

The CRR recovery server (see [“CRR Recovery Server”](#) on page 296) manages the two CRR log minidisks and contains the code for the CRR log management function (see [“Logging Function”](#) on page 306) and resynchronization management function (see [“Resynchronization Function”](#) on page 305). [Figure 40](#) on page 295 shows the CRR recovery server and SFS resource manager as separate virtual machines, but they could be the same virtual machine (not recommended).

## CRR Recovery Server

---

If you plan to use SFS (either locally or remotely), IBM recommends you install **one and only one** CRR recovery server on each system. Running without a CRR recovery server, which is known as *limp mode*, results in:

- Serious SFS performance degradation
- Only one protected resource on a CMS work unit can be updated
- Protected conversations not being allowed

In *limp mode*, you can still access SFS file pools, but you can only commit files within the *same* file pool with integrity.

The CRR recovery server maintains the CRR logs and performs the resynchronization processing in the event of a system error. IBM supplies a CRR recovery server, which is optionally generated during z/VM installation. The IBM-supplied CRR recovery server machine is VMSEVR and the IBM-supplied CRR file pool is VMSYSR. If you would prefer to generate your own CRR recovery server, rather than use the IBM-supplied CRR recovery server, then see [“Generate a CRR Recovery Server”](#) on page 317.

<b>Note:</b>
--------------

The rest of the chapter uses the word <i>server</i> . For a description of the possible combinations and types of file pool servers, see <a href="#">Chapter 1, “File Pool Administration Overview,”</a> on page 3.
---

## Coordination Function

---

The synchronization point manager (SPM) performs the function of coordinating the commit or backout of updates to protected resources and coordinates protected conversations between distributed application programs. An application that updates multiple resources in multiple places is assured that all the changes are committed, or if that was not possible, that the changes are rolled back. An application that communicates with another application through a protected conversation is assured that all changes made by both applications are committed or rolled back.

An application can update multiple resources and begin a protected conversation to another application. The other application may also update multiple resources. Then the original application can commit all of its updates and CRR tells the other application to commit its updates too. The point at which a commit is done is known as a sync point. CRR ensures all updates since the previous sync point are committed (or rolled back). An application then repeats the process with a new set of changes to



protected resources. The updates that occur between sync points are called transactions. This type of application is a transaction program.

An application can issue a commit or rollback by using one of the following:

- z/VM Resource Recovery routines (SRRCMIT or SRRBACK)
- CSL routines (DMSCOMM or DMSROLLB)
- Participating resource-specific verbs

Next, let us discuss the following coordination subtopics:

- [“Registration and Exits” on page 297](#)
- [“Sync Point Processing” on page 298](#))
- [“Sync Point Tree” on page 299](#))
- [“Allocation and Sync Point Trees” on page 302](#))
- [“Last Agent Optimization” on page 304](#))

## Registration and Exits

Before an application can update multiple protected resources, the protected resources must register with the SPM. Then, when the SPM goes through sync point processing, the SPM reaches certain exit points and transfers control to the resource adapters. [Figure 41 on page 298](#) shows the relationship between the SPM and a resource adapter. The resource adapter code calls the IBM-supplied CSL routines to register a resource manager with the SPM. When the SPM reaches various exit points, the SPM calls the product-supplied CSL routines. The product-supplied resource adapter CSL routines do their processing and then return control back to the SPM. IBM supplies the specifications for CMS-based products to write their own resource adapter exits. For more information, see [z/VM: CMS Application Development Guide](#).

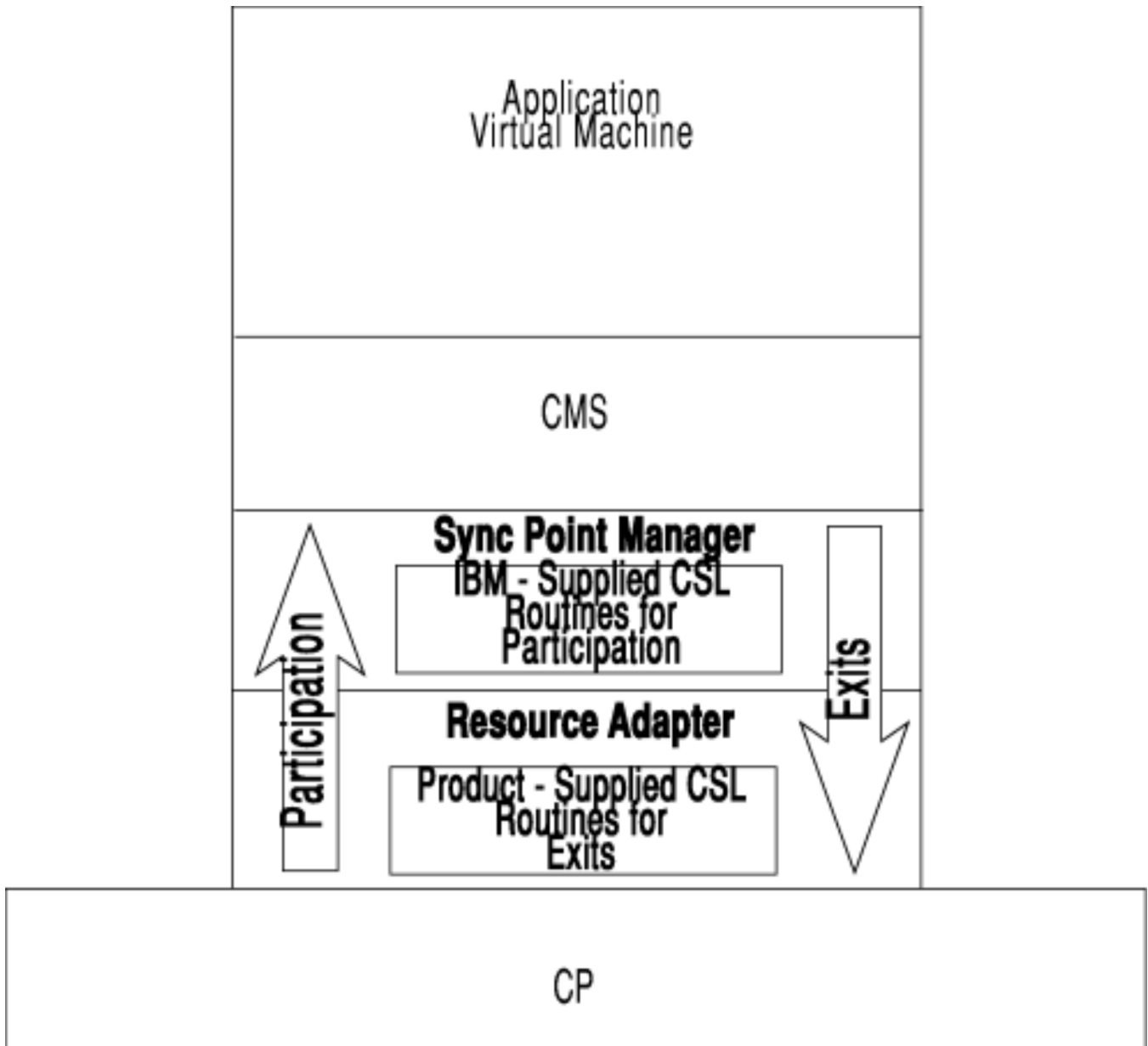


Figure 41. Registration and Exits

## Sync Point Processing

When an application issues a commit, the SPM starts sync point processing. Sync point processing consists of the SPM driving the participating resource adapters through various SPM exits. The following is a list of the exits:

- Sync Point Exits:
  - Precoordination—This exit asks the participating resource manager if it is ready for sync point processing.
  - Coordination—This exit calls the participating resource manager for sync point processing. This is the implementation of the two-phase commit protocol.
  - Postcoordination—This exit allows the participating resource manager to clean up after sync point processing.
- Additional Exits:

- End of work unit—This exit allows the participating resource manager to clean up when the work unit ends.
- Backout required—This exit tells the protected resource to put itself in a state such that backout is required.

The two-phase commit permits an application program to do distributed updates in a manner that appears "atomic" to the application. The distributed updates could be to distributed data or to distributed applications (by means of a protected conversation). The SPM coordinates the distributed updates and ensures all the updates are either committed or rolled back. The SPM returns the outcome of the sync point to the application program, so from the application program's perspective, the sync point processing appears as a single atomic step.

The usual **two-phase commit protocol** consists of these data flows:

1. First phase:

- a. **Prepare**—The SPM asks each of the protected resources and protected conversations, which are being coordinated by this SPM, if they can commit the logical unit of work.
- b. **Request commit**—Each protected resource and protected conversation responds back to the SPM whether it can commit the logical unit of work or not. If one of the protected resources or protected conversations cannot commit, it does a rollback and responds to the SPM with a backout. Then the SPM tells all the participants to roll back the changes. If the SPM gets a response from all the protected resources and protected conversations that says they all can commit, the second phase of the two-phase commit begins.

2. Second phase:

- a. **Committed**—The SPM tells the protected resources and protected conversations to make the changes permanent.
- b. **Forget**—The protected resources and protected conversations respond back to the SPM that the commit is completed and to erase the CRR log record for this completed logical unit of work.

At various times during the two-phase commit, the SPM records state information about the participants on the CRR logs. If there is an error during the second phase of the two-phase commit, the resynchronization function uses the information on the CRR logs to resynchronize the transaction. If resynchronization cannot complete the transaction, the CRR operator can manually complete the transaction. The CRR operator can view that state information of participants by using the [“CRR QUERY LU”](#) on page 376 and [“CRR QUERY LUWID”](#) on page 382 commands. See [“Problem Management”](#) on page 330 for information on using these commands.

## Sync Point Tree

The sync point tree is a method used to conceptually organize all the protected resources and protected conversations coordinated by the SPM in a transaction. The sync point tree consists of nodes that contain the protected resources or the targets of protected conversations. [Figure 42 on page 301](#) is an example of a sync point tree. VM1 is the initiator node (also called root or parent) and it is the start of the sync point.

The user application program on VM1 (also called root or parent):

- Updates the protected resources on agent nodes VM2 and VM4 (also called leaf).
- Starts a protected conversation with another user application program on cascaded initiator node VM3 (also called child).

The user application program on VM3:

- Updates the protected resources on cascaded agent nodes VM5 and VM6 (also called leaf).

In [Figure 42 on page 301](#), VM3 can have dual roles of cascaded initiator and agent, depending on how far this coordinated transaction (or LUWID) has progressed into sync point processing. Any node in the sync point tree, except the true initiator, can have the dual roles. If the sync point activity is flowing from VM3 down the sync point tree, VM3 is considered a cascaded initiator in respect to VM5 and VM6. If the sync

point activity is flowing from VM3 up the sync point tree, then VM3 is considered an agent in respect to VM1.

Usually, sync point trees are drawn that only show the protected conversations between the nodes and the protected resources are not shown. For example, [Figure 42 on page 301](#) could be drawn as [Figure 43 on page 302](#). Notice that [Figure 43 on page 302](#) only shows VM1 (parent) and the protected conversation to VM3 (child). The protected resources (leaf) are not shown. Hereafter, the general format will be used of representing sync point trees, as illustrated in [Figure 43 on page 302](#).

**Note:** In this example, only the CRR recovery servers on VM1 and VM3 play a role in the illustrated coordinated transaction. The protected conversation is between distributed application programs on VM1 and VM3, therefore the CRR recovery servers on VM1 and VM3 log the protected conversation. The CRR recovery server on VM1 also logs the protected resources on VM2 and VM4 that are being updated. The CRR recovery server on VM3 also logs the protected resources on VM5 and VM6 that are being updated.

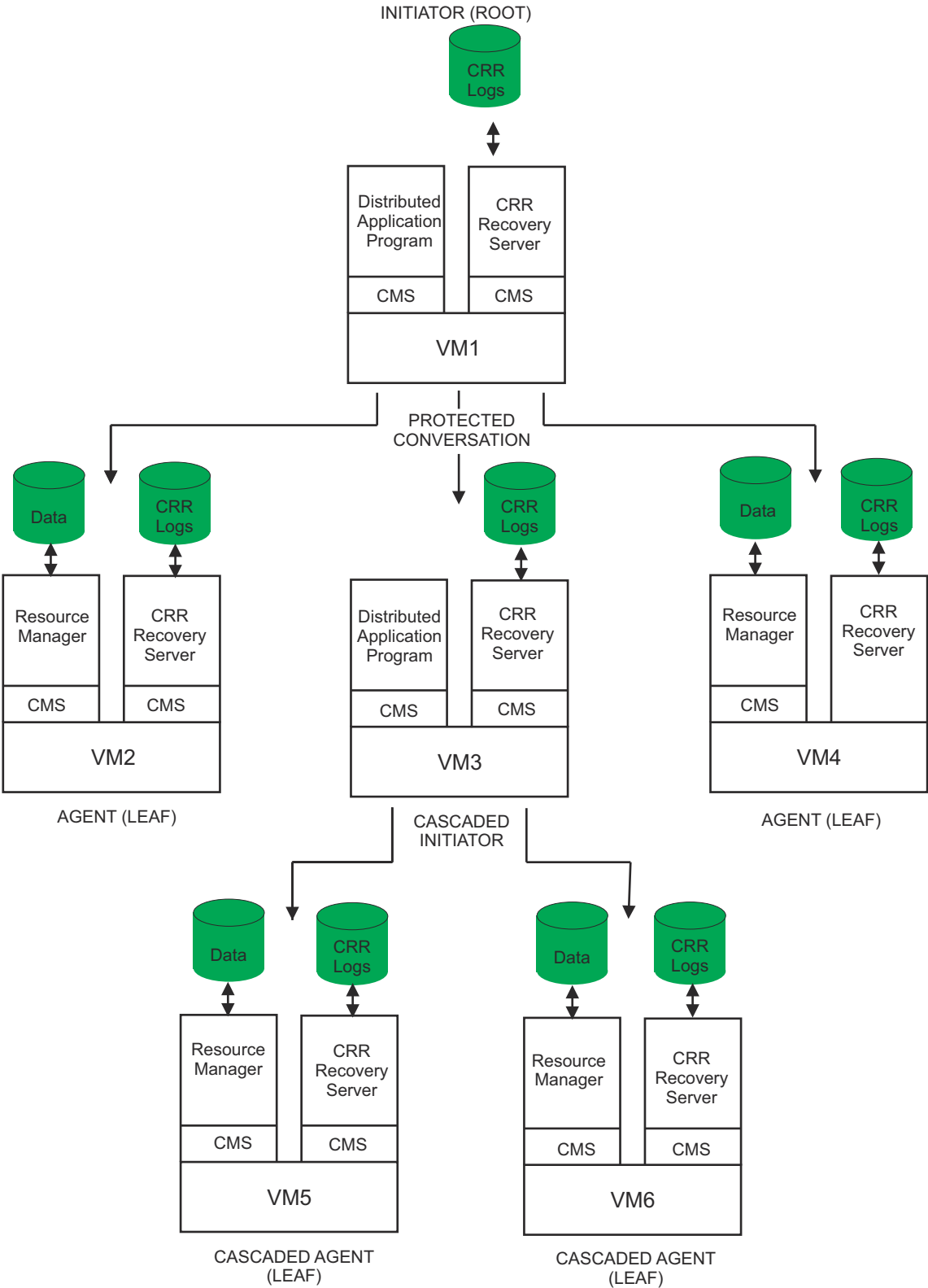


Figure 42. Sync Point Tree (Detailed Format)

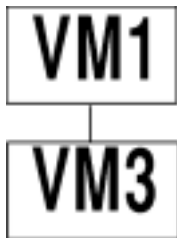


Figure 43. Sync Point Tree (General Format)

### Allocation and Sync Point Trees

The allocation tree provides a conceptual structure of the nodes allocating protected conversations within a coordinated transaction. The sync point tree provides a conceptual structure of the nodes initiating sync points within a coordinated transaction. The allocation tree and the sync point tree can be identical. This means the node that allocates the protected conversations is also the node that starts the sync point. There can be cases where the protected conversation is allocated in a node that is different from the node where the sync point originated (this is up to the designer of the distributed application). [Figure 44 on page 303](#) shows how an allocation tree could differ from a sync point tree for the same coordinated transaction. In this figure, the tree structure is simplified to show only the nodes involved with allocating protected conversations and initiating sync points. The top half of the figure shows an allocation tree and the bottom half of the figure shows the sync point tree for the same coordinated transaction.

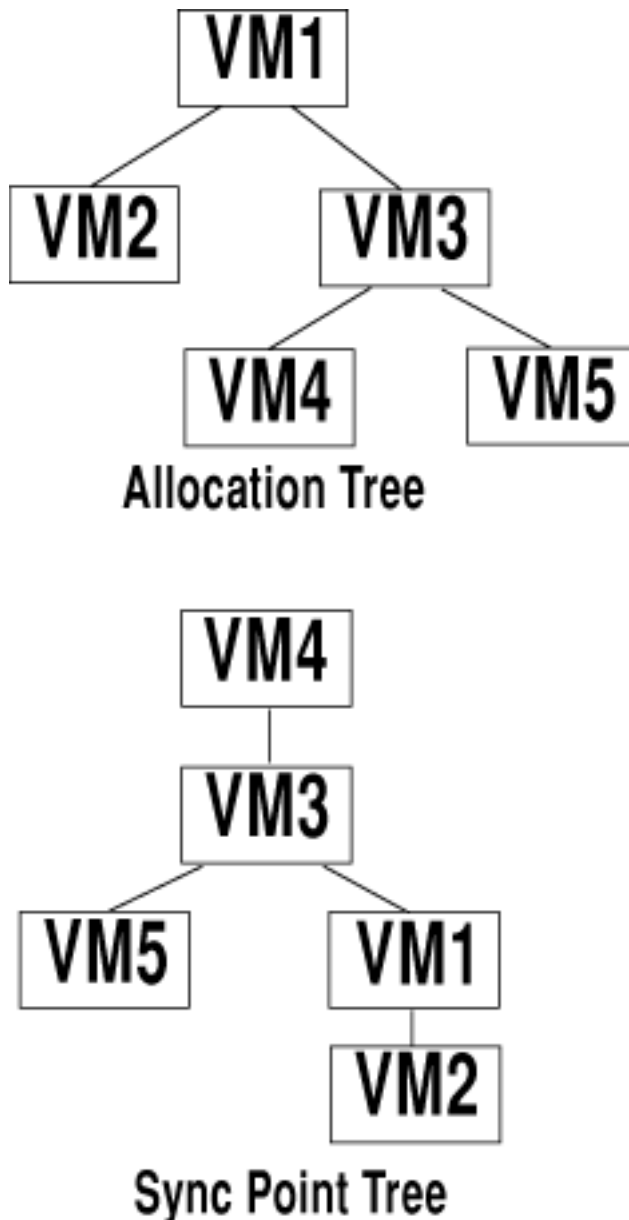


Figure 44. Allocation and Sync Point Trees

In the allocation tree, VM1 is allocating a protected conversation that causes the CRR recovery server at VM1 to provide the LUWID for the coordinated transaction. VM1 allocates a protected conversation to VM2 and VM3. VM3 then allocates a protected conversation to VM4 and VM5. But, now assume a transaction program at VM4 (not VM1) issues a commit verb to start a sync point. This results in a sync point tree that looks different from the allocation tree for the same coordinated transaction.

In the sync point tree, the node (VM4) that starts the sync point is at the top of the sync point tree. VM4 starts a sync point that flows to VM3. VM3 starts a sync point to VM5 and VM1. VM1 then starts a sync point to VM2.

Assuming in this sync point tree there is a communications error between VM3 and VM1 (CRR recovery server on VM3 tries to resynchronize with the CRR recovery server on VM1). VM1 and VM2 still have the LUWID associated with the coordinated transaction when the error between VM3 and VM1 occurred. Before the next sync point starts, the SPM breaks the communication links between VM4 and VM3, and between VM3 and VM5 so VM4, VM3, and VM5 lose the LUWID associated with this coordinated transaction that is in resynchronization. This ensures the LUWID at VM1 and VM2 is not used for a subsequent sync point at VM3, VM4, and VM5. This process of breaking the other parts of the sync point tree is called **break tree processing**. Break tree processing ensures a unique LUWID is used by each sync

point tree after a communication error. For more information about break tree processing, see [z/VM: CMS Application Development Guide](#).

### Last Agent Optimization

Last agent optimization is a technique that reduces the number of sync point flows in a sync point tree. The initiator picks one adjacent agent as a **last agent**. Then the initiator sends **prepare** to the agents that were not selected as a last agent. When these agents all respond with a **request commit** back to the initiator, the initiator sends the last agent a **request commit** rather than the usual **prepare**. This last agent is free to select one of its cascaded initiators to be the last agent and so on. The ultimate last agent is given the commit decision for the entire sync point tree.

**Note:** The SNA LU 6.2 sync point architecture requires CRR support being selected as a last agent. But, selecting a last agent is an option set of the LU 6.2 sync point architecture that is optional and is not supported by CRR.

Figure 45 on page 305 helps to explain last agent optimization. In this example, there are two non-z/VM systems (NON-VM1 and NON-VM3) that can select a last agent. NON-VM1 selects NON-VM3 as a last agent. Then, NON-VM3 selects VM5 as a last agent. VM5 accepts being selected as a last agent, but does not select a last agent. NON-VM1, NON-VM3, and VM5 all contain a distributed application that communicates by a protected conversation. VM2, VM4, VM6, and VM7 all have a protected resource managed by a resource manager (RM).

The following describes the sync point flows that could take place in the hypothetical scenario illustrated in Figure 45 on page 305 for last agent optimization:

1. NON-VM1 selects NON-VM3 as last agent.
2. NON-VM1 sends a **prepare** to VM2.
3. VM2 responds with a **request commit** to NON-VM1.
4. NON-VM1 sends a **request commit** to NON-VM3.
5. NON-VM3 selects VM5 as a last agent.
6. NON-VM3 sends a **prepare** to VM4.
7. VM4 responds with a **request commit** to NON-VM3.
8. NON-VM3 sends a **request commit** to VM5.
9. VM5 sends a **prepare** to VM6 and VM7.
10. VM6 and VM7 responds with a **request commit** to VM5.
11. VM5 sends a **committed** to VM6 and VM7 (VM5 is the ultimate last agent and decides to commit for the entire sync point tree).
12. VM6 and VM7 responds with a **forget** to VM5.  
At this point, the subtree containing VM5, VM6, and VM7 is now committed.
13. VM5 sends a **committed** to NON-VM3.
14. NON-VM3 sends a **committed** to VM4.
15. VM4 responds with a **forget** to NON-VM3.  
At this point, the subtree containing NON-VM3, VM4, VM5, VM6, and VM7 is now committed.
16. NON-VM3 sends a **committed** to NON-VM1.
17. NON-VM1 sends a **committed** to VM2.
18. VM2 responds with a **forget** to NON-VM1.

At this point, the entire sync point tree is committed.



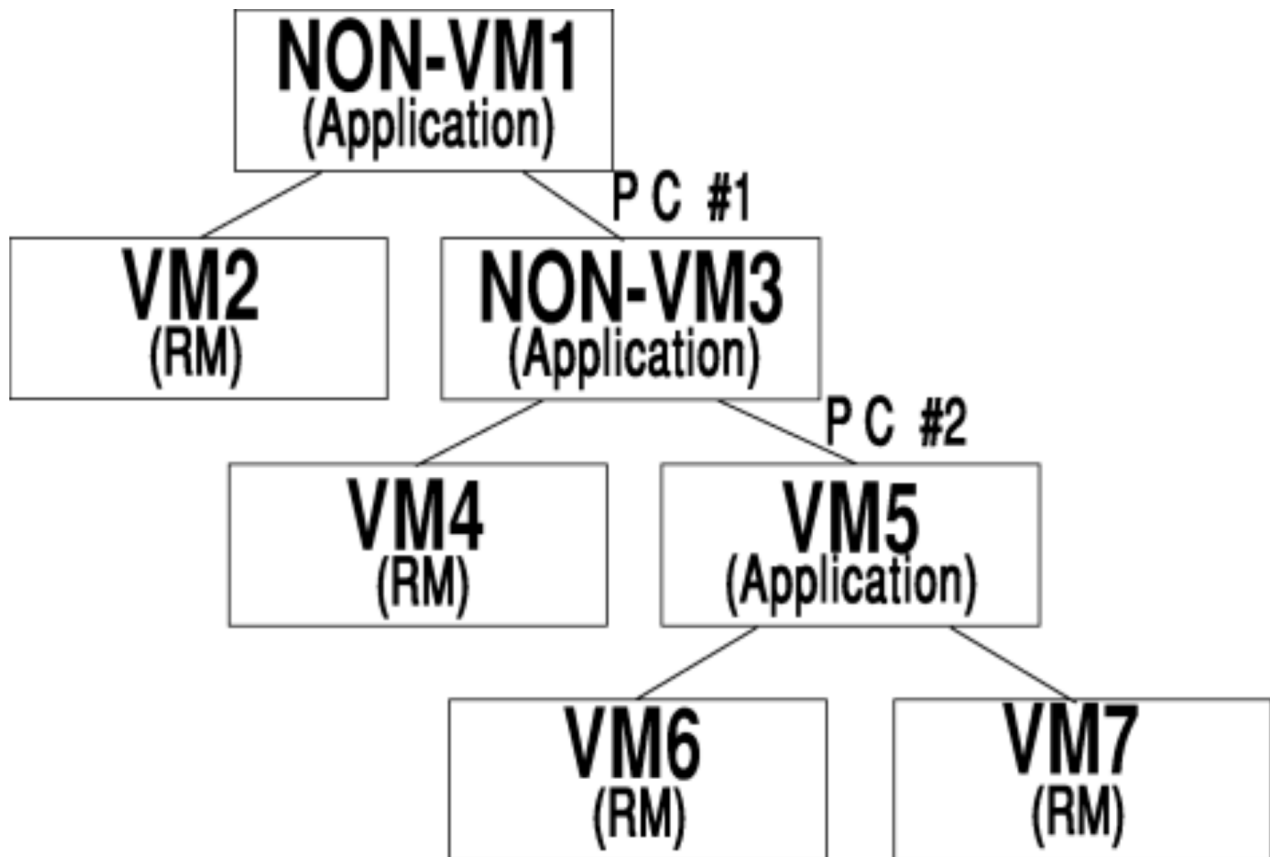


Figure 45. Last Agent Optimization

## Resynchronization Function

The purpose of the resynchronization function is to complete updates when there has been an error, such as a telecommunications line going down, during sync point processing before all updates within a transaction have been committed or rolled back. Resynchronization tries to ensure all protected resources in the transaction are either committed or rolled back, therefore maintaining the data's integrity. Resynchronization uses data logged by the SPM and by the resynchronization manager to determine what actions the resynchronization function needs to take. This data is recorded on dual CRR logs to protect against losing the CRR log data. In the case of protected resources, resynchronization tries to communicate with the unavailable protected resource manager. In the case of protected conversations, resynchronization tries to communicate with the other CRR recovery server, which resides on the same system as the protected conversation partner. The other CRR recovery server then tries to communicate with the unavailable protected resource manager.

If resynchronization cannot initially complete the updates, resynchronization issues a timed wait. After a specified time interval, which is called the timed wait interval, the resynchronization function leaves the timed wait and automatically attempts to complete the updates. This is called the **automatic periodic retry of resynchronization**.

If the CRR operator knows the protected resources will not be available for a certain period of time, the CRR operator can enter the CRR SUSPEND command. The CRR SUSPEND command stops the automatic periodic retry of resynchronization. This saves computer resources from being unnecessarily consumed while trying to resynchronize a protected resource that is not available. The CRR operator can use the CRR RESUME command to restart the automatic periodic retry of resynchronization.

For more information, see [“Stop and Restart Automatic Periodic Retry of Resynchronization”](#) on page 324 and [“Bypass and Change Timed Wait Interval”](#) on page 325.

The resynchronization function also reports an irrecoverable error if the resynchronization function cannot complete the transaction with integrity.

## Logging Function

When all protected resources and protected conversation partners become available, the resynchronization function completes the coordinated transaction. Resynchronization may not be able to complete the updates for a coordinated transaction because the protected resources or protected conversation partner may not become available for a long period of time. If resynchronization cannot complete the updates for this coordinated transaction, CRR provides the CRR RESYNC command for a CRR operator to manually complete the resynchronization.

Resources that participate in CRR should provide commands to complete the coordinated transaction (with respect to its resources). For example, SFS provides the following commands for the SFS operator to manually complete a coordinated transaction:

- QUERY PREPARED—Displays information about SFS prepared work and forced work in a coordinated transaction.
- FORCE PREPARED—Forces SFS prepared work to be committed or rolled back.
- ERASE LUNAME—Erases the history of previously forced SFS prepared work from the SFS logs.

See [Chapter 17, “Participation in CRR \(SFS only\),”](#) on page 279 and [“Problem Management”](#) on page 330 for more information about operator intervention to complete resynchronization.

## Logging Function

---

The purpose of the logging function is to maintain the CRR log minidisks, which are required for CRR. The CRR log manager, which runs in the CRR recovery server, performs the logging function. CRR uses dual CRR log minidisks on nonvolatile storage to protect against losing CRR log data because of a DASD error and also to improve the availability of the CRR recovery server. The two CRR log minidisks must be on different DASD volumes to decrease the chance of losing both CRR log minidisks. The CRR recovery server will not terminate because of I/O errors on the minidisks unless the CRR recovery server is not able to read from or write to at least one CRR log minidisks. One CRR log minidisk is a copy of the other CRR log minidisk. If one CRR log is destroyed, CRR can continue using the other CRR log. CRR sync point activity, which is needed for resynchronization, is supplied by the SPM to the CRR log manager. The CRR log manager writes the sync point data to the CRR logs. The resynchronization manager also records data on the CRR logs. The CRR logs are attached to the CRR recovery server. The CRR operator can define the CRR logs to the system, obtain information about the CRR logs, display information on the CRR logs as it relates to a specific logical unit of work, and erase CRR log information. For more information, see [Chapter 19, “CRR Administration,”](#) on page 315.

**Note:** The CRR logs do not contain user data (with the exception of the transaction tag data), they contain sync point data that describes the participants in the sync point. The participating resource manager logs data sufficient for it to recover any updates of any in-progress coordinated transactions.

## CRR Logs

---

The dual CRR log minidisks contain the CRR logs, which consist of:

- Log name table
- Log ring

The log name table uses about 33% and the log ring uses about 67% of the 4K blocks allocated to each CRR log minidisk.

## Log Name Table

The log name table contains information that describes every participating resource manager and CRR recovery server (for protected conversations) on other systems this CRR recovery server has communicated with by an exchange of log names. The log name table contains the following information about each resource manager:

- Log name (usually a time stamp)
- Fully qualified LU name

- Transaction Program Name (TPN)

The log name table does not contain TPN information about protected conversations.

If the log name table does not contain information about a protected resource or protected conversation partner that is about to participate in a sync point, there is an initial exchange of log names and the protected resource or protected conversation partner is added to the log name table. The initial exchange of log names is skipped if all the participants of the sync point are listed in the log name table. If the protected resources or protected conversation partners change their characteristics (such as gateway or TPN), then at the next sync point the log name table is updated with a new entry for the protected resources or protected conversation partner. The old entry, which contains the original LU name or TPN, should be erased. See [“SFS Log Name Table Changes”](#) on page 283 for further information.

CRR has operator commands that supply the CRR operator information about the log name table. For example, the CRR operator can issue:

- CRR QUERY LOG–Determine percentage of log name table used
- CRR QUERY LOGTABLE–Display contents of log name table
- CRR ERASE–Erase unwanted entries from the log name table

## Log Ring

The CRR log manager writes SPM records to the log ring and to a linked list that is in the CRR recovery server's virtual storage. At the start of a sync point, the SPM writes an SPM pending record. During sync point processing, the SPM updates the SPM records. Each SPM record contains information that describes the sync point activity, with respect to this node in the sync point tree, at that time for a particular sync point. At the completion of a sync point, the SPM record for the completed sync point is updated with a state of forget in the log ring and is deleted from the linked list in virtual storage. There can be many sync points occurring at the same time. The information in the linked list is what is displayed when you enter the CRR QUERY LU or CRR QUERY LUWID command.

At each CRR log checkpoint, the CRR log manager writes the contents of the linked list in virtual storage to the log ring and this marks the start of a new log ring. If the CRR recovery server crashes, when it comes back up, the SPM records are read from the log ring into the linked list in virtual storage. As more SPM records are written to the log ring, the log ring eventually wraps around and writes over old SPM records.

## Product Participation in CRR

---

Each resource has a resource manager (sometimes known as a server) and a resource adapter. The resource adapter resides in the application's virtual machine. The CRR facility provides CSL routines for other IBM and non-IBM resources to participate in CRR. Any resource that wants to participate in CRR must ensure that its:

- Resource adapter registers the product with the SPM. IBM supplies the CSL routines for the interface from the resource adapter to the SPM.
- Resource adapter handles the various exits that are driven by the SPM. IBM supplies the interface specifications for writing your own CSL routines for the various exits.
- Resource manager supports the two-phase commit protocol (see [“Sync Point Processing”](#) on page 298) and interfaces with the CRR recovery server.

An example of a resource that participates in CRR is SFS. See [Chapter 17, “Participation in CRR \(SFS only\)”](#) on page 279 for more information.

**Note:** Protected conversations are generally intended for communication between two application programs. But, there may be occasions where a resource adapter uses a protected conversation to communicate with a participating resource manager.

For information about getting other resources to participate in CRR, including information about how resource adapters and participating resource managers use protected conversations, see [z/VM: CMS Application Development Guide](#) and [z/VM: CMS Callable Services Reference](#).

## Data Restoration Considerations

---

Because CRR allows applications to update multiple protected resources with integrity, you (the CRR administrator or operator) and the application programmer (see *z/VM: CMS Application Development Guide*) should consider the possible implications of updating protected resources that do not have the same data restoration capabilities. SFS supplies the capabilities to back up user data and then to restore it to the backup level (see Chapter 7, “Recovery Procedures,” on page 101); but SFS does not provide **forward recovery** for user data, which is the process that restores data from the backup level to the point of the media error. Therefore, if one of the protected resources being updated has forward recovery, but the others do not, you would not be able to restore the protected resources to the same level. One protected resource could be restored to the point of the media error and the other protected resources could only be restored to the backup level.

You might also consider taking steps to ensure multiple protected resources can at least be restored to the same backup level in the event of a media error. This might involve shutting down all the protected resources at the same time, doing a back up of all the protected resources, and then restarting all the protected resources. Otherwise, you could have different protected resources at different levels of backup and their data restoration will not be to the same backup level.

## CRR Management in the Distributed Environment

---

IBM anticipates CRR will sometimes be used in a distributed environment where all the processors will be controlled from a central site. The central site is where the data processing skills exist for network and systems operations. The central site is also known as a data center or glass house. This means the CRR operator must be able to enter a command to any one of the CRR recovery servers and all the CRR recovery servers must be able to package together multiple line output and route their multiple line output up to the central site to the CRR operator's console. The CRR operator, at the central site, should enter the following command for each CRR recovery server:

```
prop set outid on userid msgprefix
```

**userid**

is the user ID of the CRR recovery server.

**msgprefix**

is the message prefix, such as CRR to indicate that the terminal output is coming from a CRR recovery server.

NetView is the product used for managing distributed systems. NetView command lists (CLISTs) must be installed to allow NetView to package and route the command output to the proper operator or database administrator at the central site. IBM has provided some sample CLISTs; see [Appendix E, “Sample NetView CLISTs for Distributed CRR Management,”](#) on page 707 for more information about these CLISTs.

[Figure 46 on page 309](#) shows how the CRR recovery server can route terminal output to the CRR operator at the central site.

First, the CRR recovery server uses Single Console Image Facility (SCIF) to designate the virtual machine for the system operator as the secondary user (operator virtual machine using the programmable operator facility). This means that when the CRR recovery server is running disconnected, all the CRR recovery server terminal output is sent to the secondary user.

Next, the Programmable Operator Facility routes the CRR recovery server terminal output from the secondary user to NetView. See [z/VM: CP Planning and Administration](#) for more information about the Programmable Operator Facility.

NetView, which uses the CLISTs you installed, routes the CRR recovery server terminal output to the CRR operator's console.

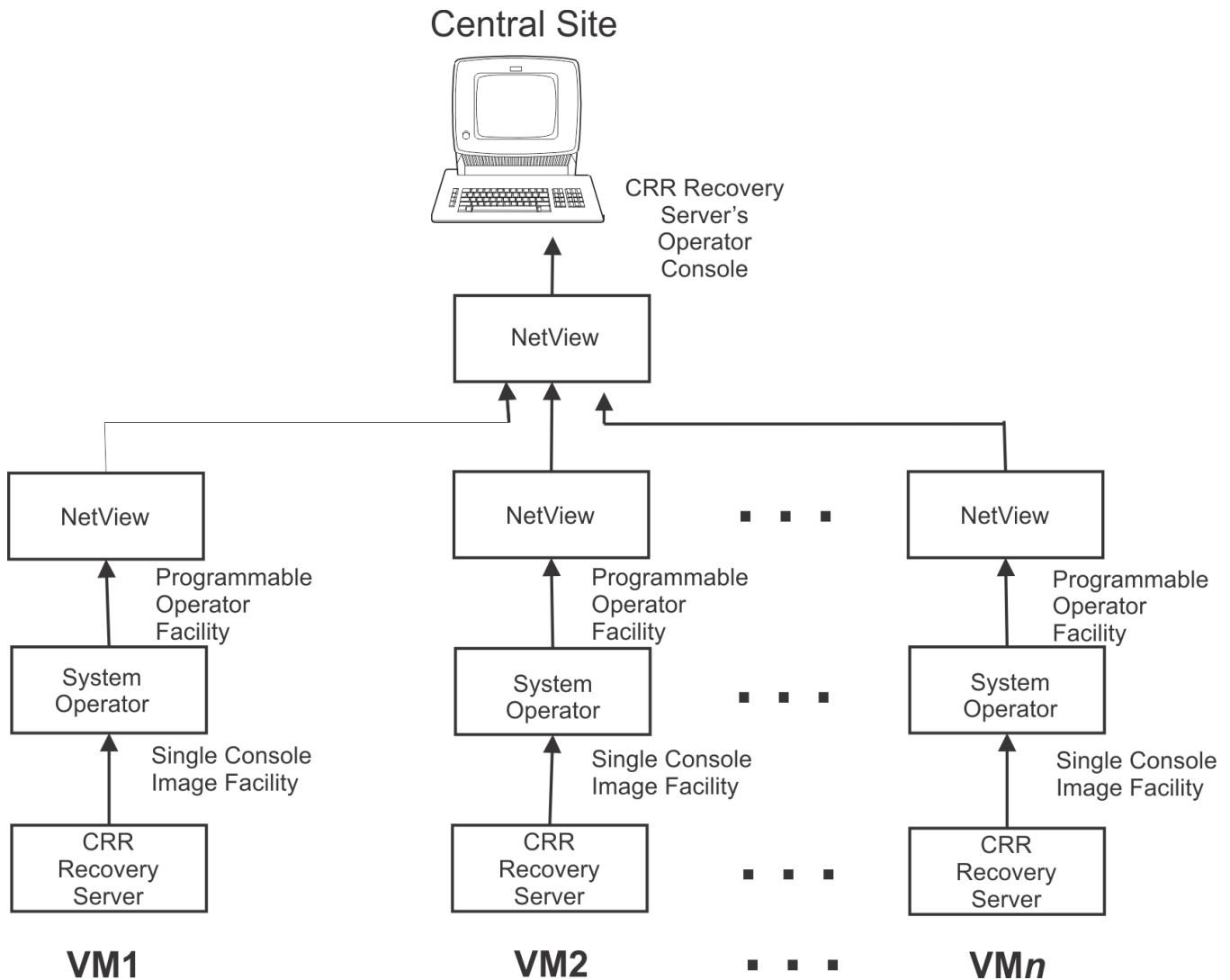


Figure 46. Distributed Management of CRR

## What Is a Transaction Program?

Figure 47 on page 310 and Figure 48 on page 310 are simplified examples of a distributed application. In this example, the one part of the application program updates protected\_resource\_#1 and protected\_resource\_#2 and the other part of the distributed application updates protected\_resource\_#3 and protected\_resource\_#4. The two parts of the distributed application communicate by a protected conversation. The application program then repeats the process with a new set of changes to protected resources. A transaction, in this example, is defined as the processing performed by an application program to change a set of protected resources from one consistent state to another consistent state. Therefore, in Figure 47 on page 310 and Figure 48 on page 310 the two parts of the distributed application program are executing  $x$  transactions, where one transaction consists of:

- Update protected\_resource\_#1 (by TPN A)
- Update protected\_resource\_#2 (by TPN A)
- Update protected\_resource\_#3 (by TPN B)
- Update protected\_resource\_#4 (by TPN B)

## What Is an LUWID?

```
[1]    START TPN A
[2]    GET CMS WORK UNIT NUMBER AND PUSH ONTO STACK
[3]    INITIATE PROTECTED CONVERSATION TO TPN B

[4]    DO UNTIL ALL TRANSACTIONS DONE
[5]        UPDATE PROTECTED_RESOURCE_#1
[6]        UPDATE PROTECTED_RESOURCE_#2
[7]        SEND DATA OVER PROTECTED CONVERSATION TO TPN B
[8]        COMMIT
[9]    END

[10]   END PROTECTED CONVERSATION TO TPN B
[11]   END TPN A
```

Figure 47. Simplified Structure for a Distributed Application Program (TPN A)

```
[1]    START TPN B
[2]    CMS WORK UNIT NUMBER SUPPLIED FROM PROTECTED CONVERSATION SUPPORT
[3]    ACCEPT PROTECTED CONVERSATION FROM TPN A

[4]    DO UNTIL ALL TRANSACTIONS DONE
[5]        RECEIVE DATA AND COMMIT NOTIFICATION OVER PROTECTED
        CONVERSATION FROM TPN A
[6]        UPDATE PROTECTED_RESOURCE_#3
[7]        UPDATE PROTECTED_RESOURCE_#4
[8]        COMMIT
[9]    END

[10]   END TPN B
```

Figure 48. Simplified Structure for a Distributed Application Program (TPN B)

Any application program that executes transactions is a transaction program. A transaction coordinated by CRR is sometimes called a coordinated transaction. The transaction program name (TPN) identifies the transaction program. When allocating a conversation link between SNA network nodes, it is necessary for the source program (initiator of the conversation) to specify the LU name and TPN of the target program (receiver of the conversation) to be called at the partner LU. The TPN is not limited to identifying only transaction programs, it identifies any program you want to begin a conversation with, such as a resource manager program.

**Note:** The resource adapter starts conversations with its resource manager for updating the protected resource, by means of the application calling a high-level interface.

This example uses protected conversations, which demonstrates the use of TPN to identify transaction programs, not resource manager programs. In this example, TPN A (Figure 47 on page 310) is the source transaction program and TPN B (Figure 48 on page 310) is the target transaction program.

## What Is an LUWID?

When a transaction program starts a protected conversation, the CRR recovery server assigns a CRR logical unit of work identifier (LUWID) to the transaction. If there were no protected conversations involved, when the transaction program issues its first commit, the SPM requests an LUWID from the CRR recovery server. The LUWID consists of:

- Fully qualified LU name from the LUNAME startup parameter
- Instance number
- Sequence number, which the SPM increments by one after each sync point

The LUWID is used to identify the CRR logical unit of work for this coordinated transaction. The SPM:

- Writes the LUWID to the CRR recovery server log.

- Passes the LUWID to all the participating resource adapters, which are responsible for passing the LUWID to the participating resource manager, which writes the LUWID to the resource log.
- Passes the LUWID to the SPM of the protected conversation partners. Each SPM, in the entire sync point tree, propagates the LUWID to the next protected conversation partner's SPM.

Even if the protected resources are distributed to other systems or if there are multiple levels of protected conversations, they all are associated with the same LUWID. The LUWID is the "common thread" that ties all the distributed parts of a transaction together. The CRR operator commands use the LUWID extensively.

A CRR logical unit of work consists of one or more instances of the logical unit of work. An LUWID instance represents the work done by another user ID (each application program) for this LUWID. In the example shown in [Figure 47 on page 310](#) and [Figure 48 on page 310](#), the LUWID has two instances:

- Instance #1—represents the work involved with TPN A to update protected resources #1 and #2 and also points to the second instance of this LUWID.
- Instance #2—represents the work involved with TPN B to update protected resources #3 and #4 and also points to the first instance of this LUWID.

The work involved at each protected resource is the resource logical unit of work. Therefore, a CRR logical unit of work (equivalent to a coordinated transaction or LUWID) consists of one or more LUWID instances, each of which may consist of one or more resource logical units of work.

**Note:** If SFS has a logical unit of work it could not complete, SFS calls it prepared-and-not-connected work and CRR calls it in-doubt work.

The CRR operator can enter several formats of the CRR command on the LUWID instance, which is specified with a token value. The CRR operator can also enter the CRR RESYNC command on a subset of an LUWID instance (resource logical unit of work), which is specified with an index value.

## What About LUWIDs and CMS Work Units?

Multiple CMS work units on different systems have the same LUWID during a coordinated transaction. A transaction program obtains a CMS work unit (or uses the default CMS work unit) and uses the same CMS work unit for all the transactions. For each transaction, the previous LUWID is incremented by one and is associated with the CMS work unit. An LUWID is an expression of a part of a CMS work unit for a TPN. CRR ensures all TPNs have the same LUWID value.

If TPN A has a protected conversation to TPN B and TPN B has a protected conversation to TPN C, then:

- TPN A, TPN B, and TPN C each have their CMS work units
- TPN A, TPN B, and TPN C all have the same LUWID associated with their CMS work units.

For more information about CMS work units, see [z/VM: CMS Application Development Guide](#).

## CRR Functional Flow

---

This section provides a generalized functional flow of CRR. This information may be helpful to a programmer who has the responsibility of designing code to get an IBM or non-IBM resource to participate in CRR. For more information about CRR participation, see [z/VM: CMS Application Development Guide](#).

In this z/VM example, assume an application:

1. Updates two protected resources
2. Starts a protected conversation with distributed application
3. Issues a commit

Also, assume the protected resources are SFS file pools and the participating resource managers are the SFS file pool servers.

The following describes the CRR functional flow for this application:



1. Application accesses, by means of the SFS resource adapter, the protected resources. The application can also set a transaction tag, by using the Set Transaction Tag CSL routine (DMSSETAG), that could supply useful recovery information about the coordinated transaction in case there was an error. The transaction tag data is written to the CRR logs and SFS writes it to the SFS logs too. See [z/VM: CMS Callable Services Reference](#) for more information about the DMSSETAG CSL routine.
2. SFS resource adapter:
  - a. Registers the protected resources with the SPM that resides in the application's virtual machine. See [z/VM: CMS Callable Services Reference](#) and [z/VM: CMS Application Development Guide](#) for more information about the resource adapter registration CSL routine (DMSREG) and the other participation interfaces.
  - b. Obtains information about the CRR recovery server that is on the same system as this application and passes the information to the participating resource managers.
3. Participating resource managers determine if they must do an initial exchange of log names (ELN) with the CRR recovery server. For more information on the initial ELN, also known as resynchronization initialization, see [z/VM: CMS Application Development Guide](#).
4. Application starts a protected conversation with a distributed application.
5. SPM obtains an LUWID, which identifies this coordinated transaction, from the CRR recovery server.
6. The protected conversation is registered with the SPM that resides in the application's virtual machine. Also, the protected conversation is registered with the SPM that resides in the partner distributed application (target of the protected conversation).
7. Application issues a commit, which is the start of sync point processing that consists of these SPM exits:
  - Precoordination exit—SPM asks if the resource adapter of the participating resource manager is ready for sync point processing.
  - Coordination exit—Calls the participating resource manager for sync point processing, which is an implementation of the two-phase commit protocol.
  - Postcoordination exit—Allows the participating resource manager to clean up after sync point processing.
8. During the precoordination exit, the participating resource manager is asked if it is ready for sync point processing. If it is not, control is returned to the application to correct the situation and retry the commit.
9. At the start of the coordination exit processing, the CRR recovery server, at the SPM's request, writes a CRR log record that describes the participants in this sync point and identifies its LUWID.

The SPM then goes through the coordination exit (two-phase commit of sync point processing) with all protected resources and protected conversations.

SFS resource adapters, during the first phase of the two-phase commit, pass the LUWID and transaction tag to the participating resource managers.

Other participating resource adapters must pass the LUWID to their participating resource managers prior to the end of the first phase of the two-phase commit.

**Note:** Passing the transaction tag is optional, but very beneficial.

Each participating resource manager writes a resource-specific log record that describes the data to be committed.

If any protected resource or protected conversation cannot complete the first phase of the two-phase commit, all updates to all protected resources in this LUWID are rolled back to their original value.

If all protected resources can commit, the second phase of the two-phase commit tells all the participating resource managers to commit the data.

If there is an error during the two-phase commit, the resynchronization function attempts to complete the coordinated transaction.



If the application had called the Set Sync Point Options CSL routine (DMSSSPTO) to set WAIT=NO, the application makes one attempt at resynchronization. If the attempt fails, the application does not wait for the automatic periodic retry of resynchronization, which occurs asynchronous to the application. If the application had set WAIT=YES, the application waits for resynchronization processing to complete.

At the start of resynchronization, the CRR recovery server does an ELN and a compare states with the participating resource manager. This process is called resynchronization recovery. The compare states provides the action (for example, commit or backout) for the participating resource manager to follow to recover the protected resource that failed in the LUWID instance.

For protected conversations, the CRR recovery server does an ELN and compare states with the CRR recovery server at the node that is the target of the protected conversation. Then the target CRR recovery server does whatever processing necessary to recover all protected resources associated with the failed LUWID instance. For more information on the ELN and compare states, also known as resynchronization recovery, see *z/VM: CMS Application Development Guide*.

After resynchronization completes, all protected resources involved in the failed LUWID, should be in a consistent state (for example, all the protected resources are either committed or backed out).

If resynchronization processing cannot complete the LUWID, the LUWID instance issues the automatic periodic retry of resynchronization, which consists of cycles of timed waits and resynchronization attempts.

If recovery is required sooner than the automatic periodic retry of resynchronization can provide it, manual intervention is needed. Manual intervention involves the CRR operator or the participating resource manager operator or both:

- a. Determining whether the protected resource should be committed or backed out. This is a heuristic decision that involves using the CRR QUERY LU and CRR QUERY LUWID operator commands and resource-specific commands such as SFS's QUERY PREPARED operator command.
- b. Forcing the protected resource to be committed or backed out. The force can be done by the CRR operator using the CRR RESYNC command or by the participating resource manager operator using resource-specific commands such as SFS's FORCE PREPARED operator command.

For more information about manual intervention, see [Chapter 17, "Participation in CRR \(SFS only\),"](#) on page 279 and ["Problem Management"](#) on page 330.

10. At the completion of the coordination exit, the SPM drives the participating resource adapters through the postcoordination exit to allow the participating resource managers to clean up.



## Chapter 19. CRR Administration

The CRR recovery server operator is the person who has access to the CRR recovery server machine console (or the secondary user console, if the Single Console Image Facility (SCIF) is being used) and can issue the CRR recovery server operator commands. The CRR recovery server operator could be one person managing the CRR recovery server for one system or the operator could be at a central site managing many distributed CRR recovery servers by means of NetView. This chapter describes the administration and operation duties of the CRR recovery server operator.

See [Table 25 on page 315](#) for a list of administration and operation activities a CRR operator may have to perform.

*Table 25. CRR Administration and Operation Activities*

<b>CRR Task</b>	<b>Location</b>
Postinstallation steps	<a href="#">Chapter 3, “Post-installation Activities,” on page 37</a>
Start a CRR recovery server	<a href="#">“Starting Multiple User Mode Processing” on page 55</a>
Monitor CRR recovery server processing	<a href="#">“Monitoring Server Operation” on page 56</a>
Stop a CRR recovery server	<a href="#">“Stopping Server Processing” on page 65</a>
Changing file pool ID	<a href="#">“Changing the File Pool ID” on page 67</a>
Generate a CRR recovery server	<a href="#">“Generate a CRR Recovery Server” on page 317</a>
Designate an alternate CRR recovery server	<a href="#">“Designate an Alternate CRR Recovery Server” on page 318</a>
Remove designation of alternate CRR recovery server	<a href="#">“Remove Designation of Alternate CRR Recovery Server” on page 320</a>
CRR Log Name Table Management	<a href="#">“CRR Log Name Table Management” on page 321</a>
Stop and restart automatic periodic retry of resynchronization	<a href="#">“Stop and Restart Automatic Periodic Retry of Resynchronization” on page 324</a>
Bypass and change timed wait interval	<a href="#">“Bypass and Change Timed Wait Interval” on page 325</a>
Set up remote user machine administration of CRR recovery server	<a href="#">“Set Up Remote User Machine Administration of CRR Recovery Server” on page 326</a>
Audit use of operator commands that intervene in CRR activity	<a href="#">“Auditing Security” on page 143</a>
Generate accounting information	<a href="#">Chapter 12, “Accounting,” on page 223</a>
Managing CRR performance	<a href="#">“Managing CRR Performance” on page 327</a>

Table 25. CRR Administration and Operation Activities (continued)

CRR Task	Location
Replace minidisks in CRR file pool	<a href="#">“Replace Minidisks in CRR File Pool” on page 327</a>
Problem management	<a href="#">“Problem Management” on page 330</a>

The CRR commands can be classified as a dedicated maintenance mode command (see [Table 26 on page 316](#)), CRR operator command (see [Table 27 on page 316](#)), and administration commands (see [Table 28 on page 316](#)). The CRR operator command is really one command, the CRR command, with various operands to perform all the desired functions.

**Note:** Many SFS commands also apply to CRR recovery servers, see [Table 5 on page 19](#), [Table 6 on page 22](#), and [Table 7 on page 26](#).

Table 26. CRR Dedicated Maintenance Mode Command

Command	Function
FILESERV CRRLOG	Reconfigures the CRR log minidisks
FILESERV DEFCCRLOG	Defines the CRR log minidisks in the CRR recovery server's POOLDEF file

Table 27. CRR Operator Command

Command	Function
CRR ERASE LU	Erases specified LU name and TPN entries from the CRR log name table
CRR ERASE LUWID	Erases CRR log records for a specified LUWID instance, which prevents any further CRR recovery server activity on this LUWID instance
CRR QUERY LOG	Displays the status of the CRR log minidisks
CRR QUERY LOGTABLE	Displays LU names and TPNs in the CRR log name table
CRR QUERY LU	Displays status of logical units of work known to this CRR recovery server and associated with the specified LU name
CRR QUERY LUWID	Displays status of sync point processing and resynchronization processing for an LUWID instance known to this CRR recovery server
CRR RESUME	Restarts the automatic periodic retry of resynchronization for a specified LUWID that was suspended by the CRR SUSPEND command and also bypasses the timed wait interval
CRR RESYNC	Provides a heuristic response for an unavailable protected resource or protected conversation so resynchronization can continue
CRR SUSPEND	Stops the automatic periodic retry of resynchronization for a specified LUWID until the CRR operator enters the CRR RESUME command

Table 28. CRR Administration Commands

Command	Function
ENROLL ADMINISTRATOR	Gives file pool administration authority to another user for monitoring the status of the CRR recovery server
DELETE ADMINISTRATOR	Revokes file pool administration authority from another user

Table 28. CRR Administration Commands (continued)

Command	Function
QUERY FILEPOOL AGENT	Displays information about the users or internal processes for which the CRR file pool server is currently doing work
QUERY FILEPOOL COUNTER	Displays information about the CRR file pool server processing against a file pool
QUERY FILEPOOL CRR	Displays information about CRR counters
QUERY FILEPOOL MINIDISK	Displays all minidisk information
QUERY FILEPOOL OVERVIEW	Displays overview information about a specified file pool
QUERY FILEPOOL REPORT	Displays information about CRR recovery server processing and about the CRR file pool
QUERY FILEPOOL STATUS	Displays information about CRR recovery server processing and about the CRR file pool

## Generate a CRR Recovery Server

If you plan to use SFS (either locally or remotely), IBM recommends you install **one and only one** CRR recovery server on each system. The CRR recovery server's functions reside in an SFS file pool server, therefore you could have the same server performing both SFS functions and CRR functions. IBM does **not recommend** using the same server for both SFS and CRR functions, except when an existing CRR recovery server is not available. Then you can temporarily designate an existing SFS file pool server to also act as an alternate CRR recovery server. See [“Designate an Alternate CRR Recovery Server”](#) on page 318 for more information.

IBM does recommend you have separate SFS file pool servers and CRR recovery servers because of these reasons:

- Limp mode avoidance—a dedicated CRR recovery server prevents limp mode. Limp mode is when an z/VM system runs without a CRR recovery server, which can result in a serious degradation in performance.
- Tuning flexibility—different tuning values can be set for SFS and CRR depending on the scenario. It is also easier to choose appropriate startup parameters for their DMSPARMS files if the SFS and CRR functions are on different servers.
- Contention avoidance—applications that need SFS and not CRR or that need CRR and not SFS can avoid contention of server resources for I/O, storage, and processor when the SFS and CRR functions are on different servers.
- Performance problem identification—separate servers makes it easier to isolate resources used for SFS and CRR.
- Better RAS—abend of one server does not affect the other server. Also, needed service or changes to the startup parameters can be made with less impact.
- Local modification flexibility—special execs can be customized for SFS or CRR.

If you do not already have a CRR recovery server, you can create one by doing one of the following:

- You can return to the z/VM installation process as described in *z/VM: Installation Guide* and now do the steps to postinstall-load the IBM-supplied servers and file pools. One of these servers is the CRR server, VMSEVR, and the CRR file pool VMSYSR. Then return to Chapter 3, “Post-installation Activities,” on page 37 to tailor the IBM-supplied VMSYSR file pool. This assumes you had not deleted the system directory entries for the IBM-supplied servers and file pools.
- Generate your own CRR recovery server and file pool. See [Chapter 15, “Generating a File Pool and Server,”](#) on page 247.

## Designating an Alternate CRR Recovery Server

If you do already have a CRR recovery server (which may have been installed during z/VM installation or you may have generated yourself) and for some reason you want to replace it:

1. Remove VMSERVR (or whatever user ID you supplied in the USER control statement in “[Step 4: Define a Server Machine](#)” on page 256) and VMSYSR (or whatever name you supplied with the FILEPOOLID startup parameter in “[Step 7: Create a Startup Parameter File](#)” on page 263) from your system as the CRR recovery server and CRR file pool. You could consider removing VMSERVR from the z/VM system directory. Also, see Chapter 16, “[Deleting a File Pool](#),” on page 277.
2. Generate your own CRR recovery server and file pool, see [Chapter 15, “Generating a File Pool and Server,”](#) on page 247.

## Designate an Alternate CRR Recovery Server

When a CRR recovery server is unavailable, an existing SFS file pool server may be designated to act as an alternate CRR recovery server.

**Note:** In this situation, you temporarily have one server acting as both an SFS file pool server and a CRR recovery server, which normally is not recommended.

The alternate CRR recovery server can use the same CRR logs as the original CRR recovery server. To designate an SFS file pool server as an alternate CRR recovery server:

1. Log on the original CRR recovery server machine, and if in multiple user mode, enter a STOP or STOP BACKUP command. This allows the original CRR recovery server to complete any active sync points before stopping.
2. Process the SETUP EXEC to access minidisks needed by the original CRR recovery server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

3. Change the CRR and LUNAME *luname* startup parameters to NOCRR and NOLUNAME in the original CRR recovery server's DMSPARMS file. If the optional RESYNCINTERVAL startup parameter was specified, delete it.
4. Remove the definition of the CRR logs from the original CRR recovery server's POOLDEF file:

```
fileserv defcrrlog delete
```

5. Using standard operating procedures for your installation, update the z/VM system directory for the CRR recovery server to specify a write access password for the CRR log minidisks, if they do not already have one.

For example, if you have the IBM Directory Maintenance program, and the CRR log minidisks are at virtual device numbers 306 and 307, you would enter:

```
dirm mdisk 306 w writepass  
dirm mdisk 307 w writepass
```

where:

***writepass***

is the write access password you are assigning to the CRR log minidisks. Save *writepass* for use in step “9” on page 319.

6. Detach the CRR log minidisks from the original CRR recovery server:

```
det 306  
det 307
```

7. Log on the existing SFS file pool server machine.

If you set up the existing SFS file pool server machine for automatic starting, as instructed in [Chapter 15, “Generating a File Pool and Server,”](#) on page 247, you will have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC will issue a FILESERV START command.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

If you forget to enter the above command, stop the existing SFS file pool server by entering one of these commands:

```
stop backup
stop nobackup
```

Either one of these commands allows the existing SFS file pool server to complete any active logical units of work before stopping.

8. Process the SETUP EXEC to access minidisks needed by the existing SFS file pool server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

9. Link to the CRR log minidisks that were just detached from the original CRR recovery server in step “6” on page 318:

```
link to userid 306 as vdev1 w writepass
link to userid 307 as vdev2 w writepass
```

where:

**userid**

is the user ID of the original CRR recovery server.

**306 and 307**

are the virtual number of the CRR log minidisks just detached from the original CRR recovery server.

**vdev1 and vdev2**

are the virtual device numbers you want to assign to the CRR log minidisks for the alternate CRR recovery server (you could use 306 for *vdev1* and 307 for *vdev2*).

**writepass**

is the write access password for the CRR log minidisks you saved in step “5” on page 318.

**Note:** This link to the CRR log minidisks is only good for the present session. If z/VM goes down or if you log off, the link is gone. Consider having your system administrator add a link to the server's z/VM system directory in case the designation of the alternate CRR recovery server will last for an extended period of time. If you do this, do not forget to have the system administrator delete the links to the CRR log minidisks when this alternate CRR recovery server is no longer needed.

10. Define the CRR log minidisks to the existing SFS file pool server's POOLDEF file.

When entering the FILESERV DEFCCRLOG command, specify the virtual device numbers of the CRR log minidisks. If, for example, the CRR log minidisks are at virtual device numbers 306 and 307, you would enter:

```
fileserv defcrrlog 306 307
```

11. Add the CRR and LUNAME *luname* startup parameters to the existing SFS file pool server's DMSPARMS file. If the optional RESYNCINTERVAL startup parameter was specified in the original CRR recovery server's DMSPARMS file, add it here.

**Note:** You should specify the same *luname* as was specified in the original CRR recovery server's LUNAME startup parameter.

12. Start the existing SFS file pool server as the alternate CRR recovery server:

```
fileserv start
```

When the original CRR recovery server becomes available, change the alternate CRR recovery server back to an SFS file pool server see [“Remove Designation of Alternate CRR Recovery Server”](#) on page 320.

## Remove Designation of Alternate CRR Recovery Server

Now that the original CRR recovery server is available for use, you should remove the designation of the existing SFS file pool server as the alternate CRR recovery server. This results in the recommended case of having a separate SFS file pool server and CRR recovery server.

1. Log on the alternate CRR recovery server machine.

If you set up the alternate CRR recovery server machine for automatic starting, as instructed in Chapter 15, “Generating a File Pool and Server,” on page 247, you will have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC will issue a FILESERV START command.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

If you forget to enter the above command, stop the alternate CRR recovery server by entering one of these commands:

```
stop backup
stop nobackup
```

Either one of these commands allows the alternate CRR recovery server to complete any active sync points and SFS logical units of work before stopping.

2. Process the SETUP EXEC to access minidisks needed by the alternate CRR recovery server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

3. Change the CRR and LUNAME *luname* startup parameters to NOCRR and NOLUNAME in the alternate CRR recovery server's DMSPARMS file. If the optional RESYNCINTERVAL startup parameter was specified, delete it.
4. Remove the definition of the CRR logs from the alternate CRR recovery server's POOLDEF file:

```
fileserv defcrrlog delete
```

5. Detach the CRR log minidisks from the alternate CRR recovery server:

```
det 306
det 307
```

6. If links to the CRR log minidisks were added to the alternate CRR recovery server's z/VM system directory, ask your system administrator to delete the links.
7. Start the existing SFS file pool server, which is no longer the alternate CRR recovery server:

```
fileserv start
```

8. Log on the original CRR recovery server machine.

If you set up the original CRR recovery server machine for automatic starting, as instructed in Chapter 15, “Generating a File Pool and Server,” on page 247, you will have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC will issue a FILESERV START command.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

If you forget to enter the above command, stop the original CRR recovery server by entering this command:

```
stop nobackup
```



This command allows the original CRR recovery server to complete any active SFS logical units of work.

9. Process the SETUP EXEC to access minidisks needed by the original CRR recovery server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

10. Define the CRR log minidisks to the original CRR recovery server's POOLDEF file.

When entering the FILESERV DEFCCRLOG command, specify the virtual device numbers of the CRR log minidisks. If, for example, the CRR log minidisks are at virtual device numbers 306 and 307, you would enter:

```
fileserv defcrrlog 306 307
```

11. Add the CRR and LUNAME *luname* startup parameters to the original CRR recovery server's DMSPARMS file. If the optional RESYNCINTERVAL startup parameter was previously specified in the original CRR recovery server's DMSPARMS file, add it back here.

**Note:** You should specify the same *luname* as was previously specified in the original CRR recovery server's LUNAME startup parameter.

12. Start the original CRR recovery server:

```
fileserv start
```

## CRR Log Name Table Management

Additions to the CRR log name table occur automatically when new participants are recognized or when old participants change their log names or communications characteristics, obsoleting previous entries. Occasionally you may consider deleting obsolete entries to make room for additional space in the table.

A CRR log name table entry contains each current or past participants' log name, along with a pair of fully qualified LU names where a:

- *Local* fully qualified LU name, represents the local source of SNA network communications
- *Remote* fully qualified LU name, represents the target for SNA network communications

For local communications (within a single system, TSAF or CS collection) there are no fully qualified LU names. Therefore the display of the table entries indicates **\*LOCAL** for the fully qualified LU name fields.

Because there may be more than one participating resource for an LU, fully qualified LU names are not sufficient to uniquely identify them. In this case the TPN field in the log name table completes the identification of the participant.

Whenever a participating resource or protected conversation connection is initiated, there is an initial exchange of log names between the participant and the CRR recovery server at the originating application node. This ensures the CRR participants have consistent log data about the units of work that might require resynchronization. Each CRR recovery server and participant records their communication partner's log name in their own log name table, identified by LU names and TPNs as described above.

Normally there is only one log name table entry per participant. However, changing the items in the following list may cause residual, obsolete entries in the log name table:

- SNA network IDs (first part of the fully qualified LU name)
- Gateway names (second part of the fully qualified LU name)
- Resource TPNs (SFS file pool ID, for example)
- CRR recovery server start up parameter (fully qualified) LU name which is the basis for formulating the recovery server's identifying TPN

**Note:** When changing the recovery server LUNAME startup parameter it is necessary to either do a FILESERV GENERATE or FILESERV CRRLOG to accomplish the change. For more information on these

commands, see [“FILESERV GENERATE”](#) on page 513 and [“FILESERV CRRLOG”](#) on page 502. This assumes a quiesce of CRR work and resetting of all log name table information. `FILESERV GENERATE` would not be appropriate if the recovery server file pool is also used to store user files.

- From TSAF or ISFC to SNA communications, or from SNA communications to TSAF or ISFC

Although new entries are created automatically when changes occur, obsolete entries are not deleted automatically. An accumulation of obsolete entries may cause insufficient free space in the table, therefore it is recommended you purge obsolete entries as changes are made.

In order to maintain the integrity of the log name tables and avoid CRR error conditions, it is very important that changes such as those listed above be done properly. Following are general guidelines for these change procedures:

- When changing identifiers such as TPNs and gateways (changing gateways affects fully qualified LU names for SNA network participants) be certain not to select names that already exist in the name space. For TPNs the name space is TSAF or CS collection; for gateways it is the SNA network.
- Use `CRR QUERY LOGTABLE` command to identify all participants affected by the change and notify the appropriate administrators to clean up log name log entries accordingly. For more information, see [“CRR QUERY LOGTABLE”](#) on page 373.
- Avoid inconsistencies by making changes only after quiescing CRR activity. Normally this is accomplished by stopping the recovery server that controls the changed element. This quiescing is to prevent inconsistencies in log name entries.
- In the case where there is a network identifier change that may affect participating resources, contact resource managers and be certain to hold up on such changes until all affected resource manager log name tables have been purged of entries that reflect the obsolete network identifier (in the fully qualified LU names).
- After all participants have cleaned up their log name tables, bring up the recovery server again and clean up obsolete log name entries using the `CRR ERASE LU` command. For more information, see [“CRR ERASE LU”](#) on page 368.

For example, if there is a change to one of the gateway names supported by the recovery server (assuming it is not the one that affects the `LUNAME` startup parameter for the recovery server file pool):

1. Make certain the new gateway name is unique in the SNA network.
2. Use the `CRR QUERY LOGTABLE` command with the `ALL` parameter. From this determine and note all cases where the local fully qualified LU name includes the gateway name to be changed.
3. Quiesce CRR activities for the recovery server by issuing the `STOP` command. For more information, see [“STOP”](#) on page 666.
4. Using the information noted in step 2, notify those CRR participants affected by the change to make appropriate changes to their Log Name Tables.
5. Change the gateway name using the `AGW DEACTIVATE GATEWAY`, `AGW ACTIVATE GATEWAY` commands (see *z/VM: Connectivity* for more information).
6. Determine all participants have completed their log name table clean up for the current change.
7. Bring up the recovery server again.
8. Use the `CRR ERASE LU` command to erase all entries noted in step 2. The LU names used in the erase are those remote fully qualified LU names that are associated with the local fully qualified LU names noted in step 2. Where there is also a TPN for the entry, the optional TPN should also be specified in the `ERASE`.

**Note:** In the unusual case where a remote fully qualified LU name used in this step is also associated with a local fully qualified LU name that is not affected by the current change, an extra entry will be erased. However, it will be recreated automatically by CRR log name exchange processing when required.

Even if you properly manage obsolete CRR log name entries, it is possible the table may grow normally with valid entries sufficient to approach capacity.

You can determine the percentage of CRR log name table space used by entering:

```
crr query log
```

See [“CRR QUERY LOG” on page 371](#) for more information.

The following describes some conditions and messages you could receive when entries are added to the CRR log name table:

- When the CRR log name table reaches 95% or more of its physical size and a new entry is added to the CRR log name table, you receive the following warning message:

```
DMS3337W CRR log name table is nn % full
```

- When the CRR log name table reaches 100% of its physical size, the new entry is not added to the CRR log name table and you receive the following error message:

```
DMS3337E CRR log name table is full
```

Before the CRR log name table becomes 100% full, the CRR operator should either:

- Increase the size of the CRR log minidisks, or
- Free space in the CRR log name table.

## Increase Size of CRR Log Minidisks

The CRR log name table uses about 33.3% of the 4KB blocks in the CRR log minidisks. If you increase the size of the CRR log minidisks, you are effectively increasing the size of the CRR log name table. For more information, see [“Replace Both CRR Log Minidisks” on page 328](#).

## Free Space in the CRR Log Name Table

The CRR operator can free space in the CRR log name table by following these steps:

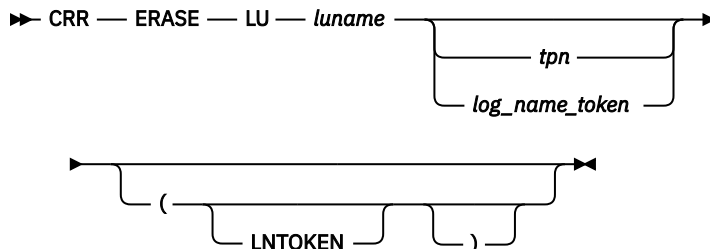
1. Display the contents of the CRR log name table by entering:

```
crr query logtable all
```

Because the CRR log name table may contain many entries, you may want to limit the scope of the CRR QUERY LOGTABLE command by using the BEFORE option to display only those entries that were not referenced recently. These entries are likely candidates to be erased from the CRR log name table.

See [“CRR QUERY LOGTABLE” on page 373](#) for more information.

2. Examine the output of the CRR QUERY LOGTABLE command (see the Last Referenced field) to determine which entries can be erased for protected resources and protected conversations no longer being used. You might want to write down the fully qualified LU name and TPN pairs you are going to erase from the CRR log name table. If the TPN contains unprintable characters, you would want to write down the fully qualified LU name and log name token pairs instead. You will need this information for step [“3” on page 323](#).
3. Erase unwanted entries from the CRR log name table by entering:



where *luname*, *tpn*, and *log\_name\_token* are what you saved from step [“2” on page 323](#). You would enter this command for each fully qualified LU name and TPN pair you want to erase. See [“CRR ERASE LU” on page 368](#) for more information.

# Stop and Restart Automatic Periodic Retry of Resynchronization

If a coordinated transaction does not gain access (during the second phase of the two-phase commit) to a protected resource it is trying to update or to a protected conversation partner, CRR goes into resynchronization processing. Resynchronization tries to establish communication with the unavailable protected resource or protected conversation partner to complete the transaction. If resynchronization does not communicate with the protected resource or protected conversation partner, resynchronization issues a timed wait and this starts the automatic periodic retry of resynchronization. After a time period (timed wait interval) has elapsed, resynchronization retries to establish communication with the unavailable protected resource or protected conversation partner. If resynchronization is again unsuccessful, resynchronization goes back into a timed wait. Again, after the timed wait interval has elapsed, resynchronization leaves the timed wait and again attempts to establish communication with the unavailable protected resource or protected conversation partner. The automatic periodic retry of resynchronization continues until resynchronization can complete or until operator intervention. The frequency of resynchronization leaving the timed wait and trying to communicate with the unavailable protected resource or protected conversation partner is based on the RESYNCINTERVAL startup parameter. See “Startup Parameter Descriptions” on page 342 for more information. The RESYNCINTERVAL startup parameter's default is 600 seconds (10 minutes).

## Stop Automatic Periodic Retry of Resynchronization

If you discover the protected resource or protected conversation partner will be unavailable for an extended period of time, you may want to stop the automatic periodic retry of resynchronization. This prevents resynchronization from automatically attempting to communicate with the unavailable protected resource or protected conversation partner and therefore saves system resources. The CRR operator can stop the automatic periodic retry of resynchronization by following these steps:

1. Obtain *token* from the:

- Resynchronization messages displayed at the CRR recovery server operator's console for this CRR recovery server, or
- Output of the CRR QUERY LU command.

Assume the value obtained for *token* is 2f. Remember to record this *token*, because you need to use it again to restart the automatic periodic retry of resynchronization.

2. Enter:

```
crr suspend 2f
```

Now, the automatic periodic retry of resynchronization for this LUWID instance has been stopped and its status has changed from timed wait to suspend.

## Restart Automatic Periodic Retry of Resynchronization

If this LUWID instance is in a timed wait (and not in suspended status) and you discover the protected resource or protected conversation partner will soon be available, you may want to restart the automatic periodic retry of resynchronization so resynchronization can complete the coordinated transaction when the protected resource or protected conversation partner is available. The CRR operator can restart the automatic periodic retry of resynchronization by following these steps:

1. Obtain the *token* used when you stopped the automatic periodic retry of resynchronization.

2. Enter:

```
crr resume 2f
```

Now, the automatic periodic retry of resynchronization for this LUWID instance has been restarted and attempts to complete the coordinated transaction.

## Bypass and Change Timed Wait Interval

This section describes the following timed wait intervals:

- Bypass Timed Wait Interval
- Change Timed Wait Interval

### Bypass Timed Wait Interval

If this LUWID instance is in a timed wait (and not in suspend status) and you discover the protected resource or protected conversation partner will be available before the next automatic resynchronization attempt, you may want to bypass the remaining time in the timed wait interval. The CRR operator can bypass the timed wait interval by following these steps:

1. Obtain *token* from the:

- Resynchronization messages displayed at the CRR recovery server operator's console for this CRR recovery server, or
- Output of the CRR QUERY LU command.

Assume the value obtained for *token* is 2f.

2. Enter:

```
crr resume 2f
```

This causes an immediate resynchronization attempt regardless of the amount of time left in the timed wait interval.

If the CRR RESUME command is unsuccessful, the LUWID instance returns to the original timed wait interval and the automatic periodic retry of resynchronization continues.

If the CRR RESUME command is successful, but the resynchronization attempt is not, the LUWID instance enters a new timed wait interval with the clock reset to the value of the RESYNCINTERVAL startup parameter and the automatic periodic retry of resynchronization continues.

If the CRR RESUME command is successful, and the resynchronization attempt is successful for this LUWID instance, the coordinated transaction is completed and the automatic periodic retry of resynchronization is ended.

### Change Timed Wait Interval

If you decide the default timed wait interval of 10 minutes before the next retry at resynchronization is not suitable, you can change the timed wait interval by following these steps:

1. Log on the CRR recovery server machine.

If you set up the server machine for automatic starting, as instructed in Chapter 15, “Generating a File Pool and Server,” on page 247, you will have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC will issue a FILESERV START command.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

If you forget to enter the above command, stop the CRR recovery server by entering this command:

```
stop nobackup
```

This command allows the CRR recovery server to complete any active sync points before stopping.

2. Process the SETUP EXEC to access minidisks needed by the server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

3. Change the RESYNCINTERVAL startup parameter in the DMSPARMS file for the CRR recovery server to the number of seconds you desire. For more information, see [RESYNCINTERVAL](#).
4. Enter the FILESERV START command to restart CRR recovery server processing:

```
fileserv start
```

## Set Up Remote User Machine Administration of CRR Recovery Server

---

If you have multiple systems within a TSAF collection, CS collection, or a SNA network, you may have multiple CRR recovery servers if one was generated on each system. It might be beneficial to have one person enter user machine administration commands for the multiple CRR recovery servers. The only task the CRR administrator would have to perform would be to monitor the status of the CRR recovery servers.

The only user machine commands applicable to a CRR recovery server are:

- Administration:
  - DELETE ADMINISTRATOR
  - ENROLL ADMINISTRATOR
  - QUERY FILEPOOL AGENT
  - QUERY FILEPOOL COUNTER
  - QUERY FILEPOOL CRR
  - QUERY FILEPOOL MINIDISK
  - QUERY FILEPOOL OVERVIEW
  - QUERY FILEPOOL REPORT
  - QUERY FILEPOOL STATUS
- End Use (see [z/VM: CMS Commands and Utilities Reference](#) for more information about this command):
  - QUERY FILEPOOL CONNECT

Follow these steps to set up remote user machine administration of a CRR recovery server:

1. Enroll someone to serve as the CRR administrator. You would enroll the same person for all the CRR recovery servers. For detailed information on enrolling an administrator, see [“Enrolling an Administrator”](#) on page 141.
2. Change each of the CRR recovery server's file pool IDs from VMSYSR to a file pool ID that does not have a VMSYS prefix. File pool IDs with a VMSYS prefix are identified (IUCV \*IDENT) by the server as a local resource, which prevents remote CRR administration by a user machine. But, file pool IDs that do not have the VMSYS prefix are identified (IUCV \*IDENT) by the server as a global resource and therefore capable of remote CRR administration by a user machine. For detailed information on changing the file pool ID, see [“Changing the File Pool ID”](#) on page 67.

**Note:** When you update the DMSPARMS file with the new *filepoolid* value for the FILEPOOLID startup parameter, remember to also specify the REMOTE startup parameter. If you do not specify the REMOTE startup parameter, the default is the LOCAL startup parameter. The LOCAL startup parameter prevents the server from accepting file pool ID connections from outside the processor, even though the server has identified the file pool ID as a global resource.

3. See [Chapter 13, “Setting Up a File Pool for Remote Use,”](#) on page 233 for additional information regarding remote usage.

## Managing CRR Performance

For a discussion of tasks related to performance management of CRR recovery servers, see [z/VM: Performance](#).

### Replace Minidisks in CRR File Pool

If you are interested in recovery procedures for SFS file pools, see [Chapter 7, “Recovery Procedures,”](#) on [page 101](#). IBM recommends you do control data backup and restore (use the BACKUP and RESTORE startup parameters) for SFS file pools, see [“Backing Up a File Pool - Overview”](#) on [page 101](#).

IBM recommends you do **not** do control data backup and restore (use the NOBACKUP and NORESTORE startup parameters) for CRR file pools. In CRR file pools, there is no user-created file pool catalog data (storage group 1) and user data (storage group 2) to protect. The file pool catalog and user minidisks are minimal size and are only required for server startup and shutdown. In the unlikely event of a error, the minidisks can easily be recreated by using the FILESERV GENERATE command, which is an easy administrative task. You would only have to recreate a minidisk if the error is with the control or catalog minidisks. If you did control data back up, you would have to periodically process a control data back up and save them just in case you ever got the error. Therefore, not doing a control data backup and restore is administratively simpler and cheaper.

CRR file pools, are mainly interested in maintaining the integrity of the CRR log minidisks, not the other minidisks. The CRR logs contain extremely important sync point data necessary for resynchronization processing. The integrity of the CRR logs is maintained by having two copies of the CRR logs on different DASD volumes, not by doing a backup and restore.

**Note:** Do not back up the CRR log minidisks, because if you restore them, you are replacing current sync point data with old sync point data.

See [Table 29 on page 327](#) for a list of minidisks that a CRR operator may have to replace in a CRR file pool.

*Table 29. CRR File Pool Minidisk Replacement Tasks*

Minidisk to be Replaced	Location
Replace one CRR log minidisk	<a href="#">“Replace One CRR Log Minidisk” on page 328</a>
Replace both CRR log minidisks	<a href="#">“Replace Both CRR Log Minidisks” on page 328</a>
Replace one file pool log minidisk	<a href="#">“Replacing One File Pool Log Minidisk” on page 131</a>
Replace both file pool log minidisks	<a href="#">“Replacing Both File Pool Log Minidisks” on page 133</a>
Replace the control minidisk	<a href="#">“Replacing File Pool Minidisks” on page 129</a>
Replace the storage group 1 (catalog) minidisk	<a href="#">“Replacing File Pool Minidisks” on page 129</a>
Replace the storage group 2- <i>n</i> (user) minidisks	<a href="#">“Replacing File Pool Minidisks” on page 129</a>
Replace the work minidisk	<a href="#">“Replacing File Pool Minidisks” on page 129</a>

## Replace One CRR Log Minidisk

CRR records synchronization (sync) point activity on the CRR log in case resynchronization needs the CRR log information to complete a sync point. Because the CRR log data is very important, CRR uses dual logging. This means CRR log data is written to two CRR log minidisks. These two CRR log minidisks must be the same size and type. (Where both the devices are FB-512 devices, you can consider them to be the same device type, even if they are different DASD models). The two CRR log minidisks must be on different DASD volumes. If the DASD that contains one of the CRR log minidisks is destroyed, CRR still has the other CRR log minidisk available for use on another DASD volume. If both CRR log minidisks were on the same DASD volume, both CRR log minidisks would be destroyed and the CRR recovery server would system malfunction.

If one of the CRR log minidisks is "damaged" because of an irrecoverable DASD error or because the CRR log has been corrupted by accidentally being overwritten, then to replace the damaged CRR log minidisk:

1. Replace the damaged CRR log minidisk with a new minidisk of the same size and type as the existing CRR log minidisk.

**Note:** For best performance, the CRR logs should be on separate channels and control units.

2. Update the MDISK control statement for the CRR logs in the server's z/VM system directory entry. **You must not change the virtual device numbers when updating the MDISK statements.**
3. Enter the CMS FORMAT command on the new CRR log minidisk, using a block size of 4096.

**Note:** Dual CRR log processing copies the good CRR log data to the new empty CRR log. If you do not format the new CRR log minidisk, the results are unpredictable and you could end up with two damaged CRR logs.

4. Enter the CMS RESERVE command on the new CRR log minidisk using CRR1 or CRR2 for the label (as appropriate).
5. If the virtual number of the new CRR log minidisk changed, enter the FILESERV DEFCRRLOG command to define the new CRR log minidisk in the server's POOLDEF file.

The FILESERV DEFCRRLOG command must be entered from the server machine console.

When entering the FILESERV DEFCRRLOG command, specify the virtual device numbers of the CRR log minidisks. If, for instance, the CRR log minidisks are at virtual device numbers 306 and 307, you would enter:

```
fileserv defcrrlog 306 307
```

**Note:** You need to enter the FILESERV DEFCRRLOG command only if you had changed the virtual device numbers from what was previously used.

6. Enter the FILESERV START command to restart CRR recovery server processing:

```
fileserv start
```

## Replace Both CRR Log Minidisks

This section describes how to replace both CRR log minidisks. You might want to replace both CRR log minidisks because you:

- Want to increase or decrease the space allocated.
- Need to replace both CRR log minidisks lost because of errors (such as media errors).
- Want to move both CRR log minidisks to another physical device.
- Need to reformat both CRR log minidisks because they have been corrupted (perhaps the DASD areas were accidentally overwritten).

To replace both CRR log minidisks, follow these steps:

1. Log on the CRR recovery server machine.



If you set up the server machine for automatic starting, as instructed in [Chapter 15, “Generating a File Pool and Server,”](#) on page 247, you will have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC will issue a FILESERV START command.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

If you forget to enter the above command, the FILESERV START command may be unsuccessful if the CRR log minidisks are corrupted.

- Force the server to complete all sync points.

**Do not skip this step.** To complete all the sync points, you must allow multiple user mode processing to stop normally. Reconnect to the server machine if you are not currently logged on to it and enter this operator command:

```
stop nobackup
```

Do **not** use a STOP IMMEDIATE command because any CRR log records of sync points that are in process are erased when the new CRR log minidisks are formatted. This prevents the sync point from ever completing.

- Process the SETUP EXEC to access minidisks needed by the server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

- Verify the FORMAT startup parameter is in effect.

Check the DMSPARMS file for the NOFORMAT startup parameter. If it is specified, delete it. You do not need to explicitly specify FORMAT, as it is the default.

- Log off the CRR recovery server machine.
- Using standard operating procedures for your installation, update the z/VM system directory entry for the server machine.

Change the MDISK control statements for the CRR log minidisks to indicate the new size and location. Remember the CRR log minidisks must be of identical size, so they should be defined on the same DASD device type. (Where both the devices are FB-512 devices, you can consider them to be the same device type, even if they are different DASD models). The maximum size of a CRR log minidisk is a DASD volume.

**Note:** This step may not be necessary for all cases. If, for example, you are following these instructions because both CRR logs were accidentally overwritten, it is not necessary to update the CRR log minidisk definitions.

If you are changing the physical location of the CRR log minidisks, you should review the log placement considerations in [“Where to Place the Logs”](#) on page 253.

In addition to changing the CRR log sizes and, perhaps, their placement, you can also change their virtual device numbers. If you change the virtual device numbers, remember the new numbers. They are needed in a later step.

- Log back on to the CRR recovery server machine.

If you set up the server machine for automatic starting, as instructed in [Chapter 15, “Generating a File Pool and Server,”](#) on page 247, you will have to avoid executing the PROFILE EXEC when CMS is initializing. Otherwise, the PROFILE EXEC will issue a FILESERV START command.

To avoid having the PROFILE EXEC process, enter this command immediately after IPL CMS:

```
access (noprof
```

If you forget to enter the above command, stop the server by entering this operator command:

```
stop nobackup
```

This command allows the server to complete any active logical units of work before stopping.

8. Process the SETUP EXEC to access minidisks needed by the server machine (For more information, see [SETUP EXEC](#)):

```
setup
```

9. Enter the FILESERV DEFCRRLOG command to define the CRR log minidisks in the CRR recovery server's POOLDEF file.

The FILESERV DEFCRRLOG command must be entered from the server machine console.

When entering the FILESERV DEFCRRLOG command, specify the virtual device numbers of the new CRR log minidisks. If, for instance, you defined new larger CRR log minidisks at virtual device numbers 306 and 307, you would enter:

```
fileserv defcrrlog 306 307
```

**Note:** You need to enter the FILESERV DEFCRRLOG command only if you had changed the virtual device numbers from what was previously used.

10. Enter the FILESERV CRRLOG command to format the CRR log minidisks.

The FILESERV CRRLOG command must be entered from the server machine console.

When entering the FILESERV CRRLOG command, specify the virtual device numbers of the new CRR log minidisks. If, for instance, you defined new larger CRR log minidisks at virtual device numbers 306 and 307, you would enter:

```
fileserv crrlog 306 307
```

If the command is unsuccessful, enter it again after correcting the problem. For a complete description, see “[FILESERV CRRLOG](#)” on page 502.

11. Enter the FILESERV START command to resume multiple user mode processing:

```
fileserv start
```

## Problem Management

---

The ultimate goal of CRR problem management is to find all resources in an LUWID, determine the status of the resources, and then take action to make all the resources consistent.

This section describes the steps a CRR operator has to follow in the rare case of CRR not being able to complete a transaction. See [Figure 49 on page 331](#) for the example transaction in this CRR problem management scenario. In this scenario:

- Application program on VM1 updates the protected resources on VM2 and VM4.
- Application program on VM1 initiates a protected conversation with the application program on VM3.
- Application program on VM3 updates the protected resources on VM5 and VM6.

In this scenario, let's assume:

- The error occurs during the second phase of the two-phase commit.
- The error is the loss of the protected conversation between VM1 and VM3.
- The protected resources at VM2 and VM4 are committed.
- VM3, VM5, and VM6 are in-doubt.

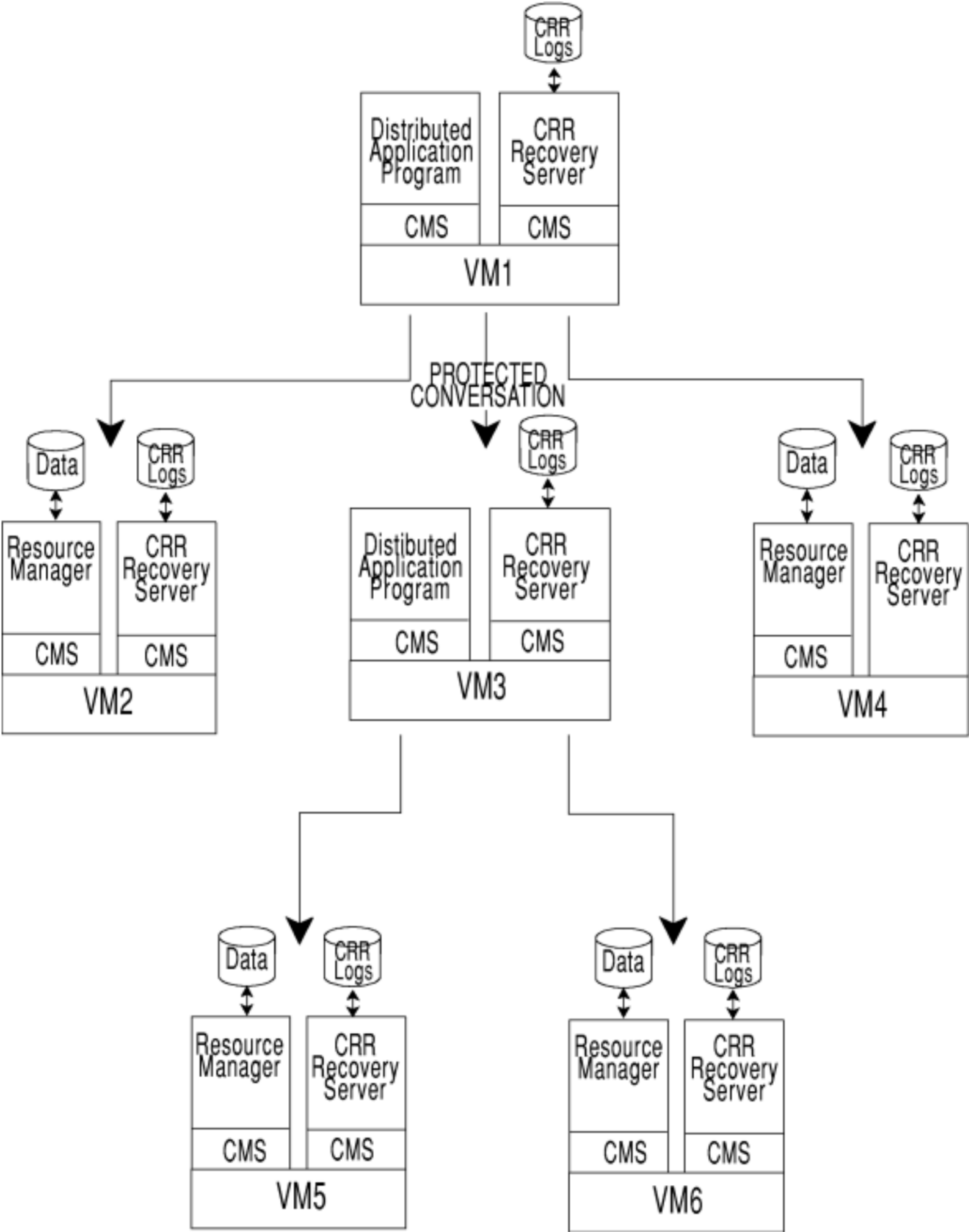


Figure 49. CRR Problem Management Scenario

**Note:**

1. Always examine the transaction tag before proceeding with any problem management steps. An application can write a transaction tag on to the CRR logs by using the Set Transaction Tag CSL routine (DMSSETAG) as described in *z/VM: CMS Callable Services Reference*. The transaction tags may identify a file that contains recovery procedures in the event of a error. Reading the recovery procedures could save the CRR operator a lot of work.
2. When you enter a CRR QUERY LU or CRR QUERY LUWID command, from the CRR recovery server console, the console's screen may fill up with output. If you let the CRR recovery server console's screen fill up with output and you do not receive the message "Operator command processing complete", you have tied up the CRR recovery server and the coordinated transactions cannot access the CRR recovery server. The CRR operator should scroll through the console's output, print the contents of the CRR recovery server console, and then examine the printed output. Scrolling through the console output frees up the CRR recovery server and makes it available for coordinated transaction processing. Assuming the server's console has been spooled using the SPOOL CONSOLE START command, you could print the contents of the console by using the SPOOL CONSOLE CLOSE command and then printing the reader file.

You could also avoid filling up the CRR recovery server console by using SCIF and issuing commands from a secondary user console.

### Step 1: Notification of a Problem

The CRR operator can be notified of a problem through various methods, such as:

- Phone call from an administrator of a protected resource, who received a phone call from a user who is complaining that some data is locked because of pending resynchronization activity. For details of this scenario, see [Chapter 17, "Participation in CRR \(SFS only\)," on page 279](#).
- Resynchronization messages displayed on the CRR operator's console indicating resynchronization processing has started, but has not been able to complete.
- Resynchronization messages displayed on the CRR operator's console indicating heuristic damage has occurred.

In this example, when the protected conversation is lost and CRR goes into resynchronization processing, the CRR operator receives the following message:

```
DMS3305I LUWID VM1.SERV1.A10DE4DF4402D.0001 with token 0000005F
requires a resynchronization on mm/dd/yy at hh:mm:ss with:
LU VM3 with index 2
Transaction tag: THIS IS THE SOURCE SIDE OF THE USER APPLICATION.
```

If resynchronization processing is not able to complete the coordinated transaction, the CRR operator receives the following message from the CRR recovery server at VM1:

```
DMS3309I The resynchronization of LUWID VM1.SERV1.A10DE4DF4402D.0001
with token 0000005F is being suspended on mm/dd/yy at hh:mm:ss
Resynchronization was started on mm/dd/yy at hh:mm:ss for the LUWID.
The resynchronization awaits the availability of:
LU VM3 with index 2
```

and then resynchronization enters a timed wait. After a timed wait interval, which is established by the RESYNCINTERVAL startup parameter, resynchronization leaves the timed wait and attempts to complete the coordinated transaction. (For more information, see RESYNCINTERVAL.) In the example, the default timed wait interval of 10 minutes. After 10 minutes, resynchronization leaves the timed wait and attempts to complete the transaction. If the protected conversation to VM3 is still not available and therefore resynchronization could not complete the transaction on this attempt, the CRR operator again receives message DMS3309I and resynchronization returns to the timed wait. If message DMS3309I continues to be displayed on the CRR operator's console, the CRR operator should investigate the cause of the delay, which results in the continuous automatic periodic retry of resynchronization for the coordinated transaction.

## Step 2: Obtain Token for LUWID at VM1

The CRR operator can obtain the token from:

- Output from the resynchronization messages
- CRR QUERY LU command.

In the scenario, the CRR QUERY LU command to obtain the token. The CRR operator, at the CRR recovery server on VM1, enters:

```
crr query lu all
```

The output could look like this:

```
Time: hh:mm:ss                CRR QUERY LU - CRRSERV1
Date: mm/dd/yy                LUNAME - VM1.SERV1

LUWID                          Token
VM1.SERV1.A10DE4DF402D.0001    0000005F
Name                            Process
COLEMAN                        RESYNCHRONIZATION
Syncpoint Role                  Syncpoint State      Status
INITIATOR                       COMMITTED             TIMED WAIT
Transaction Tag
THIS IS THE SOURCE SIDE OF THE USER APPLICATION.
```

In the output of the CRR QUERY LU command, look for an LUWID that has:

- Process equal to RESYNCHRONIZATION, and
- Status equal to TIMED WAIT.

In this example, the LUWID instance that meets the above criteria has a token of 5F. Record the LUWID and token value for use in the following steps.

### Note:

1. The output of the CRR QUERY LU command could contain many screens of LUWID data.
2. Rather than using the token obtained from the output of the CRR QUERY LU command, you could also obtain the token from DMS3305I or DMS3309I to use as input into the CRR QUERY LUWID command in the next step.

## Step 3: Determine Status of Resources at VM1

Using the token saved from the previous step, enter:

```
crr query luwid 5F
```

The output could look like this:

```

Time: hh:mm:ss          CRR QUERY LUWID - CRRSERV1
Date: mm/dd/yy         LUNAME - VM1.SERV1

LUWID
VM1.SERV1.A10DE4DF402D.0001
Name
COLEMAN
Syncpoint Role          Syncpoint State
INITIATOR               COMMITTED
Transaction Tag
THIS IS SOURCE OF USER APPLICATION.

Resources
VM2    SERV2
  Recovery TPN
  RECSERV2
  Recovery Token
  00003ABB
  Resync Role           Resync State
  FORGET                COMMITTED
VM3    SERV3
  Recovery TPN
  .2
  '06F2'X
  Recovery Token
  A15CCDEB35D91000
  Resync Role           Resync State
  RESYNC NEEDED        RESYNC NEEDED
VM4    SERV4
  Recovery TPN
  RECSERV4
  Recovery Token
  00002722
  Resync Role           Resync State
  FORGET                COMMITTED

Token
0000005F
Process
RESYNCHRONIZATION
Status
TIMED WAIT

Index
1
Access Userid
COLEMAN

Index
2
Access Userid
COLEMAN

Index
3
Access Userid
COLEMAN

```

Look for a resource in the output that has a role of RESYNC NEEDED. This tells the CRR operator that at VM3, resynchronization is in process. But the output does not tell the CRR operator what resources resynchronization is trying to update. The CRR operator has to enter a CRR QUERY LUWID command at VM3 to determine what is happening at VM3. Record the LU name from the output associated with the resource that has the role of RESYNC NEEDED. This is the LU name at VM3.

Also note that the resources at VM2 and VM4 have been committed, because the resynchronization states are COMMITTED.

### Step 4: Obtain Token for LUWID at VM3

The CRR operator cannot obtain a token (which may or may not be the same token as obtained from VM1) for this LUWID at VM3 from a resynchronization error message, because the CRR recovery server at VM3 did not enter a resynchronization message. Therefore, at VM3, the CRR operator should enter:

```
crr query lu all
```

The output could look like this:

```

Time: hh:mm:ss                CRR QUERY LU - CRRSERV3
Date: mm/dd/yy                LUNAME - VM3.SERV3

LUWID                          Token
VM1.SERV1.A10DE4DF402D.0001    00000020
Name                            Process
COMER                           SYNCPOINT
Syncpoint Role                  Syncpoint State      Status
AGENT                            IN-DOUBT
Transaction Tag
THIS IS THE TARGET SIDE OF THE USER APPLICATION.

Initiator Name
VM1      SERV1
  Recovery TPN
  .2
  '06F2'X
  Recovery Token                  Index
  A15CCDEB35D91000              2
  Resync Role                    Access Userid
  NOT-IN-RESYNC                 COMER
  Resync State
  NOT-IN-RESYNC

```

Examine the output for the matching LUWID (saved in step 2) and record the token (in this example, it is 20) associated with the LUWID for use in the next step.

## Step 5: Determine Status of Resources at VM3

Next the CRR operator should determine the status of the resources for this LUWID at VM3 to see what is happening to the resources at VM5 and VM6.

The CRR operator, using the token obtained in the previous step, should enter:

```
crr query luwid 20
```

The output could look like this:

```

Time: hh:mm:ss                CRR QUERY LUWID - CRRSERV3
Date: mm/dd/yy                LUNAME - VM3.SERV3

LUWID                          Token
VM1.SERV1.A10DE4DF402D.0001    00000020
Name                            Process
COMER                           SYNCPOINT
Syncpoint Role                  Syncpoint State      Status
AGENT                            IN-DOUBT
Transaction Tag
THIS IS THE TARGET SIDE OF THE USER APPLICATION.

Initiator Name
VM1      SERV1
  Recovery TPN
  .2
  '06F2'X
  Recovery Token                  Index
  A15CCDEB35D91000              1
  Resync Role                    Access Userid
  NOT-IN-RESYNC                 COMER
  Resync State
  NOT-IN-RESYNC

Resources
VM5      SERV5
  Recovery TPN
  RECSERV5
  Recovery Token                  Index
  0000CAF2                       2
  Resync Role                    Access Userid
  NOT-IN-RESYNC                 COMER
  Resync State
  NOT-IN-RESYNC

VM6      SERV6
  Recovery TPN
  RECSERV6
  Recovery Token                  Index
  0000319B                       3
  Resync Role                    Access Userid
  NOT-IN-RESYNC                 COMER
  Resync State
  NOT-IN-RESYNC

```

The sync point state is IN-DOUBT, which means VM3 is waiting to hear from the initiator at VM1 for direction to either commit or roll back the resources at VM5 and VM6.

The resynchronization state is NOT-IN-RESYNC, which means the resources are not aware they need resynchronization.

### Step 6: Determine Heuristic Action

At this point, the CRR operator has determined from the preceding steps that the:

- Resource at VM2 has committed
- Resource at VM4 has committed
- Cascaded initiator at VM3 is in-doubt
- Resource at VM5 is in-doubt
- Resource at VM6 is in-doubt

Therefore, half the resources for this LUWID have committed so the CRR operator could decide to commit the resources at VM5 and VM6 if the protected conversation will be down for a long time.

**Note:** If possible, let resynchronization processing complete the coordinated transaction. Only take heuristic action as a last resort.

### Step 7: Take Heuristic Action

If the CRR operator is taking the heuristic action, the CRR operator, at VM3, should enter:

```
crr resync 20 1 commit purge
```

The 20 is the token for this LUWID instance at VM3. The 1 is the index from the previous output. The index indicates the CRR RESYNC command is responding for the initiator at VM1 to drive resynchronization down to VM5 and VM6 (parental direction). The heuristic action is to commit. PURGE is used to erase CRR log records of this LUWID at VM3. For more information, see [“CRR RESYNC” on page 390](#).

This results in the resources at VM5 and VM6 being committed so all the resources in this LUWID are in a consistent state of committed.

The CRR operator, at VM1, should answer for VM3 by issuing:

```
crr resync 5F 2 commit purge
```

This reflects what was started at VM3 to prevent resynchronization from starting at VM1 and manually completing resynchronization.

This results in the resources at VM2 and VM4 being committed so all the resources in this LUWID are in a consistent state of committed.

**Note:** The preceding problem management scenario involved a CRR operator taking the heuristic action, but there are other scenarios that involve heuristic action being taken by the operator of a participating resource manager.

For example, assume:

- Communication links between VM3 and VM5 and between VM3 and VM6 are down.
- Participating resource managers on VM5 and VM6 are SFS file pool servers.

In this scenario, the CRR operator would inform the SFS operator to do a commit (or backout, if appropriate) of the prepared-and-not-connected SFS work on VM5 and VM6. This would involve the SFS operator in issuing the QUERY PREPARED and FORCE PREPARED commands on VM5 and VM6. For more information on the steps the SFS operator takes to manually complete a coordinated transaction, see [Chapter 17, “Participation in CRR \(SFS only\),” on page 279](#).



---

## Part 3. Administration Reference

This part contains reference information on the server startup parameters and the administration and server commands. This information is explained in the following chapters:

- [Chapter 20, “File Pool Server Startup Parameters,” on page 339](#)
- [Chapter 21, “File Pool Administration and Server Commands,” on page 357](#)



---

## Chapter 20. File Pool Server Startup Parameters

You can specify parameters to be used by the file pool server FILESERV command processing. These parameters, referred to as *startup parameters*, are described in this chapter.

A file pool server has the potential for performing any of the following server services:

- SFS server repository services
- BFS server repository services
- CRR recovery server services
- FIFO server services

However, startup parameters are provided for establishing whether a file pool server will act as:

- a CRR recovery server
- a FIFO server

A file pool server can do these specialized, non-repository server services or it can allow for another file pool server to do them.

A file pool server can perform SFS or BFS repository services based on the file spaces created (enrolled), not based on startup parameters. However, other operational characteristics of the repository file pool server are established by its startup parameters. Many of these parameters also affect the operation of FIFO and CRR recovery services.

A file pool server may be dedicated to one of the services listed above or it can involve combinations of them. SFS repository file pool servers can be combined with BFS repository file pool servers simply by enrolling BFS file systems, for example. However, the CRR recovery server and FIFO server services must be established (or defaulted) by startup parameters.

The CRR recovery server is used by SFS operations, but not by BFS operations. The FIFO server is used by BFS operations but not by SFS operations.

---

### How to Specify Startup Parameters

To specify startup parameters for the FILESERV commands, place them in a file having fixed-length 80-byte records. The file name must be the user ID of the server machine (or *serverid*). The file type must be DMSPARMS. The startup parameters can be in any order in the file.

While the DMSPARMS file can reside on any minidisk in the server machine's search order, it is best to keep it on the A-disk, which should be accessed in read/write mode. If the server does not find the DMSPARMS file on file mode A, it will use the first *serverid* DMSPARMS file it finds in the standard CMS search order. If it does not find a DMSPARMS file, it uses the default values, which are shown in the startup parameters syntax diagram.

For your convenience, a file named SERVER DMSPARMS is included with z/VM. It resides on the MAINT 193 disk. SERVER DMSPARMS contains a list of the server startup parameters in a form similar to that in [Figure 50 on page 340](#).

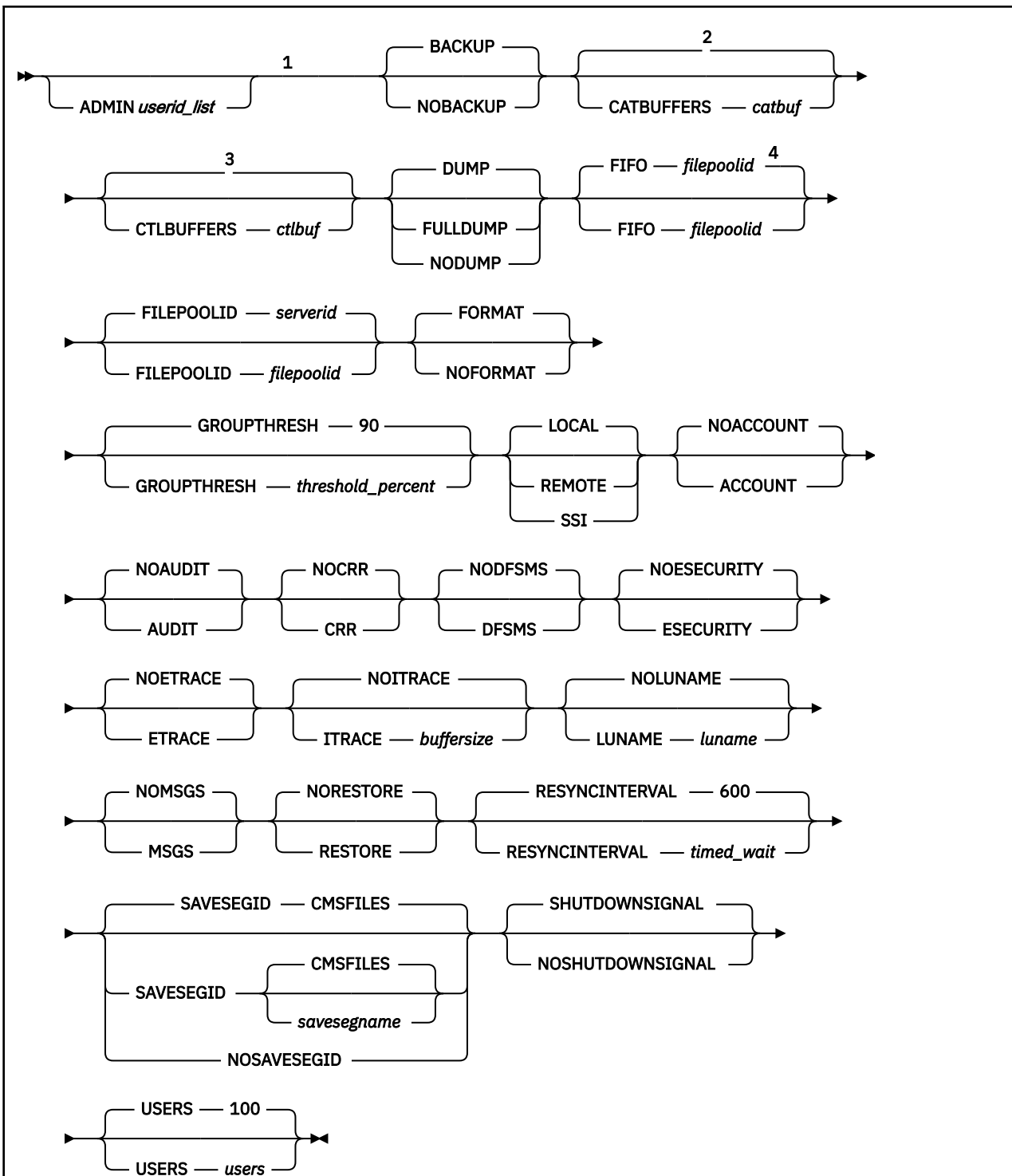
```
ADMIN userid list
BACKUP | NOBACKUP
CATBUFFERS catbuf
CTLBUFFERS ctlbuf
DUMP | FULLDUMP | NODUMP
FIFO filepoolid
FILEPOOLID filepoolid
FORMAT | NOFORMAT
GROUPTHRESH threshold_percent
LOCAL | REMOTE | SSI
NOACCOUNT | ACCOUNT
NOAUDIT | AUDIT
NOCRR | CRR
NODFSMS | DFSMS
NOESECURITY | ESECURITY
NOETRACE | ETRACE
NOITRACE | ITRACE buffersize
NOLUNAME | LUNAME luname
NOMSGS | MSGS
NORESTORE | RESTORE
RESYNCINTERVAL timed_wait
SAVESEGID savesegname | NOSAVESEGID
SHUTDOWN SIGNAL | NOSHUTDOWN SIGNAL
USERS users
```

Figure 50. Example Content of SERVER DMSPARMS File

When creating or modifying a DMSPARMS file, follow these rules:

- Each startup parameter must be in a separate record (line) of the file.
- The server will ignore any record that begins with an asterisk (\*) or that contains all blanks in positions 1 to 72.
- Do not have any values extend into or past column 72.
- Mixed case is allowed (but alphabetic characters are automatically converted to uppercase before they are used).

The following illustrates the syntax of each of the startup parameters to be used when you are creating or modifying a DMSPARMS file.



Notes:

- 1 Each startup parameter must be in a separate record (line) of the file.
- 2 Function of the USERS startup parameter value.
- 3 Function of the USERS startup parameter value.
- 4 The default for *filepoolid* will be the default name specified with the FILEPOOLID parameter.

## Startup Parameter Descriptions

---

This section describes all FILESERV startup parameters. Except where noted, a startup parameter applies to both multiple user mode and dedicated maintenance mode processing. Startup parameters that do not apply to dedicated maintenance mode operation are ignored in that mode.

### **ADMIN *userid\_list***

ADMIN identifies one or more virtual machine user IDs that are granted file pool administration authority when the server is started.

If not specified, no one is granted file pool administration authority when a server is started. In this case, the only way to grant someone file pool administrator authority is by issuing the GRANT ADMIN operator command when the server is running in multiple user mode. For more information, see [“GRANT ADMIN” on page 541](#).

#### **Note:**

1. Administration authority may also be granted to server machines that interact with the file pool. For example, if DFSMS/VM is to manage the file pool, additional ADMIN parameters will need to be added. See [z/VM: DFSMS/VM Customization](#) for more information.
2. Administration authority implies superuser status when you are using the OpenExtensions functions.

If this file pool server is to be a FIFO server for one or more other file pool servers, you must establish this file pool server as a file pool administrator for each of those other file pools. This is accomplished by including the virtual machine IDs for the FIFO file pool server in the *userid\_list* for this ADMIN startup parameter for each of the file pool servers the FIFO server serves. A file pool designates another file pool server as its FIFO server through the FIFO startup parameter.

If necessary, additional startup parameter file records can be used to contain file pool administrator user IDs. The ADMIN keyword must appear first on each of the additional records.

ADMIN is applicable to all types of file pool servers. It is ignored by dedicated maintenance mode processing.

### **BACKUP**

#### **NOBACKUP**

BACKUP indicates the backup is being handled by file pool repository facilities and automatic backup will occur when repository log usage reaches its threshold. It also means the server maintains the repository logs so they reflect control data changes made since the most recent control data backup. When BACKUP is specified, a current control data backup is always required.

The BACKUP startup parameter requires a DDNAME=BACKUP record exists in the *filepoolid* POOLDEF file. If the startup parameter BACKUP is in effect and a DDNAME=BACKUP record does not exist in the POOLDEF file, a warning message will be displayed. (The DDNAME=BACKUP record is the default backup destination. This destination can be redirected). Refer to the FILESERV DEFBACKUP command on how to define a file pool backup file. For more information, see [“FILESERV DEFBACKUP” on page 506](#).

BACKUP must be in effect for you to successfully back up the control data using either the BACKUP operator command, STOP operator command, the FILEPOOL CONTROL BACKUP administrator command, or the FILESERV BACKUP dedicated maintenance mode command. When BACKUP is in effect, BACKUP is also the default on the STOP operator command.

**Note:** When BACKUP is in effect, some dedicated maintenance mode commands will obsolete the current control data backup. This document tells you when it is necessary to back up the control data again. If you ignore those instructions, the server will realize there is no current control data backup and will prevent you from starting multiple user mode operation until a backup is made. In this case, use the FILESERV BACKUP command to back up the control data.

NOBACKUP indicates the server should not automatically back up the control data. When NOBACKUP is in effect, you cannot back up the control data. The RESTORE startup parameter is not allowed if the NOBACKUP startup parameter is specified.

The default is BACKUP.

If you have a dedicated repository file pool server or a server that is acting as both a repository file pool server and CRR recovery server (*not recommended*), or both a repository server and a FIFO server, it is recommended you use BACKUP and RESTORE startup parameters. If you have a dedicated CRR recovery server, it is recommended you use NOBACKUP and NORESTORE startup parameters.

**Note:**

1. FILESERV processing issues a FILEDEF command with a *ddname* of POOLDEF to define the *filepoolid* POOLDEF file used in backing up the control data.
2. To switch from NOBACKUP to BACKUP processing, see [“Enabling Control Data Backup Processing” on page 104](#). To switch from BACKUP to NOBACKUP, see [“Disabling Control Data Backup Processing” on page 107](#). If you change the startup parameter without following the instructions in the appropriate section, subsequent FILESERV commands will display error messages and server initialization processing will not be successful.
3. When BACKUP is in effect, the repository log minidisks must be large enough to contain the logging generated between control data backups. When NOBACKUP is in effect, the repository log minidisks can be much smaller. For a discussion of how to calculate the repository log size, see [“Replacing Both File Pool Log Minidisks” on page 133](#).

**CATBUFFERS *catbuf***

*catbuf* indicates the number of 4096-byte virtual storage buffers the server should acquire for use in reading and writing the file pool catalogs. The value *catbuf* can range from 10 to 32767.

The default is a function of the USERS startup parameter value. The CATBUFFERS default value is:

$$(mmmm/2) + 10$$

where *mmmm* is the value specified (or allowed to default) for the USERS startup parameter.

In dedicated maintenance mode, this startup parameter is computed as described above, even though the USERS startup parameter value otherwise has no meaning in dedicated maintenance mode.

For repository file pool servers, it is recommended you use the CATBUFFERS default value, which is based on the USERS startup parameter.

For dedicated CRR recovery servers or dedicated FIFO servers, the recommended value for *catbuf* is 10 (the minimum allowable value). If you use the default, there will be a lot of unneeded storage allocated for catalog buffers. Dedicated CRR recovery servers and FIFO servers only have minimal use for catalog buffers compared to repository file pool servers. Therefore, you should specify the value for *catbuf* rather than using the default for CATBUFFERS.

If you should use the same server for both repository services and CRR recovery services (not recommended) or for both repository services and FIFO services, combine the two values for *catbuf* that would have been used if SFS and CRR or FIFO services were in separate servers.

**CTLBUFFERS *ctlbuf***

*ctlbuf* indicates the number of 512-byte buffers the server should acquire for use in reading and writing the file pool control minidisk. The value *ctlbuf* can range from 10 to 32767.

The default is a function of the USERS startup parameter value. The CTLBUFFERS default value is:

$$10 \times mmmmm$$

where *mmmm* is the value specified (or allowed to default) for the USERS startup parameter.

In dedicated maintenance mode, this startup parameter is computed as described above, even though the USERS startup parameter value otherwise has no meaning in dedicated maintenance mode.

For repository file pool servers, it is recommended you use the CTLBUFFERS default value, which is based on the USERS startup parameter.

For dedicated CRR recovery servers or FIFO servers, the recommended value for *ctlbuf* is 10 (the minimum allowable value). If you use the default, there will be a lot of unneeded storage allocated for control buffers. Dedicated CRR recovery servers and FIFO servers only have minimal use for control buffers compared to repository servers. Therefore, you should specify the value for *ctlbuf* rather than using the default for CTLBUFFERS.

If you should use the same server for both repository and CRR services (not recommended) or both repository services and FIFO services, then combine the two values for *ctlbuf* that would have been used if the services were in separate servers.

### **DUMP**

### **FULLDUMP**

### **NODUMP**

DUMP indicates a partial virtual machine storage dump (not including code areas) should occur if a server error is detected. If the server is running in an XC mode virtual machine, some of the SFS data spaces may be dumped according to the following:

- If the server was in access-register mode at the time of the error, each data space accessible by way of an access register is dumped.
- If the server was not in access-register mode, no SFS data spaces will be dumped.

FULLDUMP indicates a complete virtual machine storage dump (including code areas) should occur if a server error is detected. If the server is running in an XC mode virtual machine, all data spaces created by the server are also dumped.

NODUMP indicates no virtual machine storage dump should occur if a server error is detected.

The default is DUMP.

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

Unless the designated support group for your installation asks you to specify NODUMP or FULLDUMP, you should let the server default to DUMP.

All dumps are in VMDUMP format and are written to a server machine spool file. These dumps can be read using the z/VM VM Dump Tool.

### **FILEPOOLID *filepoolid***

*filepoolid* is the one-to-eight character file pool ID name for the file pool. The server uses *filepoolid* as the:

- File name for the POOLDEF file, which contains the file pool definition information.
- Transaction program name (TPN) for the file pool. The TPN is sometimes called the resource ID.

Users refer to the file pool by the file pool ID name specified in the FILEPOOLID startup parameter. The file pool ID identifies the TPN you communicate with from a user machine to issue administration commands. The first character of the name must be a character from A through Z and the remaining positions of the name must be from A through Z or 0 through 9. The names SYSTEM, ALLOW, and ANY are not valid. Also, be sure the file pool ID you specify is not the same as a user ID on the system.

If you omit the FILEPOOLID startup parameter, the server uses its own virtual machine ID (also known as the server ID) as the file pool ID.

This startup parameter is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

### **Note:**

1. The FILEPOOLID startup parameter is not used for IUCV processing in dedicated maintenance mode. In dedicated maintenance mode, the startup parameter determines the file name for the POOLDEF file that contains file pool definition information.
2. If this is a repository file pool server and the file pool ID has a VMSYS prefix (such as VMSYS or VMSYSU), then:



- Server always identifies (using IUCV \*IDENT) the file pool ID TPN as a local resource. (The IUCV \*IDENT statement in the server's z/VM system directory entry can specify LOCAL or GLOBAL.)
- The LOCAL and REMOTE startup parameters are ignored, because the file pool ID is always identified as local.

If this is a repository file pool server and the file pool ID does not have a *VMSYS* prefix, then:

- Server always identifies (using IUCV \*IDENT) the file pool ID TPN as a global resource. (The IUCV \*IDENT statement in the server's z/VM system directory entry must specify GLOBAL.)
  - The LOCAL and REMOTE startup parameters determine whether a user outside the system (TSAF, ISFC, or SNA) can connect to the file pool.
3. Renaming a repository file pool managed by DFSMS/VM has special implications. For example, if DFSMS file migration is being used, you must also rename the DFSMS Migration Level 1 directory which supports that file pool. See *z/VM: DFSMS/VM Customization* for more information.
  4. If this is a CRR recovery server, the server always identifies itself (here the reference is not to the file pool ID TPN, but to the TPNs that are reserved for CRR services and discussed in the CRR startup parameter) as a global resource. (The IUCV \*IDENT statement in the server's z/VM system directory entry must specify GLOBAL).

**Note:** The **.luname** TPN associated with the CRR recovery server is always global.

If the file pool ID has a *VMSYS* prefix (such as *VMSYSR*), the server identifies (using IUCV \*IDENT) the file pool ID TPN as a local resource. The LOCAL and REMOTE startup parameters are ignored, because the file pool ID is always identified as local. In this case, you can only do CRR user machine administration from the same system.

If the file pool ID does not have a *VMSYS* prefix, the server identifies (using IUCV \*IDENT) the file pool ID TPN as a global resource. The LOCAL and REMOTE startup parameters determine whether a user outside the system (TSAF, ISFC, or SNA) can connect to the file pool. In this case, if the REMOTE startup parameter was selected, you can do remote CRR user machine administration.

5. The z/VM installation process creates file pools with the following file pool IDs:
  - *VMSYS*
  - *VMPSFS*
  - *VMSYSU*
  - *VMSYSR*

If other file pools are already defined on your processor, you should choose a name not already in use on that processor. If your processors are connected in a TSAF or CS collection, the file pool ID you select should be unique in the collection. The collection enforces this uniqueness in all cases except for file pool IDs beginning with *VMSYS*. So if you choose a name already in use, you will not be able to successfully start the server for multiple user access to that file pool.

A server always identifies file pools having IDs beginning with *VMSYS* as LOCAL resources, so it is possible to have duplicate file pool IDs beginning with *VMSYS* in a TSAF or CS collection (such as the IBM-supplied file pools *VMSYS*, *VMSYSU*, and *VMSYSR*).

### **FIFO filepoolid**

*filepoolid* is a one-to-eight character file pool ID name of the server that will act as the associated FIFO server. If you omit the FIFO startup parameter, a server uses its own file pool ID as the FIFO server file pool ID.

A file pool server can designate another file pool server to handle its FIFO operations by using the FIFO startup parameter. This allows for off-loading work to another server, presumably one that has little or no repository work to do. The FIFO startup parameter is pertinent only to BFS services, therefore it is ignored for SFS specific services and can be defaulted when only SFS services are used. If it is defaulted, no server-supported FIFO operations will be off-loaded. In this case the startup parameter for FIFO *filepoolid* defaults to the filepoolid established by the FILEPOOLID startup parameter. Off-loading of FIFO work is not visible to applications that use the server functions.

To understand better this optional splitting of FIFO operations, it is appropriate to understand the nature of the FIFO objects, how they are different from ordinary file objects, and how the file pool server relates to them as a repository manager. A repository file pool server provides two functions for regular files:

1. definitional or control information for the object, such as its name, type, parent directory (path), and
2. actual file data.

In the case of an ordinary file, the file data is stored on DASD and has specific recovery characteristics. On the other hand, FIFO objects involve similar definitional information, but the data is transitory and is not stored on DASD. That is, the data for a FIFO is kept in memory and disappears when the FIFO is closed. In this sense this FIFO data can be considered non-repository data, while the control for the FIFO (for example, its type, name) is considered part of the file pool repository.

With this in mind, it is easier to understand the nature of the functional split between the file pool server repository where the FIFO is defined and the file pool server where its transitory read/write operations are handled, as supported by the FIFO startup parameter. The repository function for the FIFO is not affected by the off-load; only the transitory, non-repository functions of the FIFOs are off-loaded.

FIFO service work for a file pool server potentially involves fairly high volumes of short messages from FIFO read and write operations. Such operations could potentially consume considerable resources in the file pool server where the FIFOs are defined. If a high volume of such activity is anticipated for the BFS applications that use the file pool, it is recommended you establish another server for off-loading the FIFO read/write activity for the file pool repository. Any file pool server can be designated a FIFO server, although it must be resident in the same system as the file pool that designates it.

When you specify a separate file pool for doing FIFO services, you must also set up administrative authority (with the respective ADMIN start up parameters) such that the FIFO server has administrative authority to the repository file pool that it is servicing, and, (vice versa) the repository file pool server has administrative authority for the FIFO server. Even when the FIFO read and write work is off-loaded to another server, it is occasionally necessary to communicate between the file pool server where the FIFOs are defined (repository) and the file pool server to which read and write operations are off loaded (FIFO server). For this reason, a separate FIFO server must have administrator authority for the file pool for which it is doing FIFO services. When you specify another file pool server in this FIFO startup parameter, you must be certain the ADMIN startup parameter for the same server includes the FIFO file pool server's virtual machine ID to establish administrator authority for the FIFO server.

A file pool server may act as a FIFO server for any number of other file pools. A system may have any number of FIFO servers. A FIFO server may also support file pool repository services (BFS or SFS) of its own. The FIFO parameter allows you to control this load balancing to meet your individual requirements.

Although the FIFO server must be in the same system as the file pool that defines it, VM user machines may be in other systems. Allowance for remote file pool users is a normal feature of file pool servers. The FIFO server simply retains this file pool service characteristic.

### **FORMAT**

### **NOFORMAT**

FORMAT indicates certain FILESERV commands should enter CMS FORMAT and RESERVE commands for various minidisks, as follows:

- FILESERV GENERATE—issues FORMAT and RESERVE for the control minidisk, repository log minidisks, CRR log minidisks (if this is a CRR recovery server), and all minidisks allocated to storage groups.
- FILESERV MINIDISK—issues FORMAT and RESERVE for the storage group minidisks being added.
- FILESERV LOG—issues FORMAT and RESERVE for both repository log minidisks.
- FILESERV CRRLOG—issues FORMAT and RESERVE for both CRR log minidisks.

The FORMAT or NOFORMAT startup parameters are ignored by all other FILESERV commands. You will not see the prompts that FORMAT and RESERVE commands usually issue; the FILESERV commands respond to the prompts internally.

NOFORMAT indicates FILESERV GENERATE, FILESERV MINIDISK, FILESERV LOG, or FILESERV CRRLOG should not issue CMS FORMAT and RESERVE commands for the indicated minidisks. If new file pool minidisks are being used by these commands, you must ensure CMS FORMAT and RESERVE commands are entered for them. (See #4 for instructions on using the CMS FORMAT and RESERVE commands.)

**Note:** A file pool control minidisk requires a block size of 512, and repository log minidisks, storage group minidisks, and repository log minidisks require a block size of 4096.

The default is FORMAT.

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

### **GROUPTHRESH *threshold\_percent***

Indicates a threshold percentage. When *threshold\_percent* percent of the physical space allocated to any storage group is used, the server displays a warning message on its operator console. The message identifies the storage group that is becoming full. This startup parameter is provided so you can know in advance when more DASD storage will be needed.

The value *threshold\_percent* can range from 1 to 99. If you do not specify a GROUPTHRESH value, the default for *threshold\_percent* is 90, which means the threshold message is displayed when a storage group is 90% full.

This startup parameter is applicable to repository file pool servers, CRR recovery servers, and FIFO servers. For dedicated CRR recovery servers and for dedicated FIFO servers, use the default value.

### **LOCAL**

### **REMOTE**

### **SSI**

If the FILEPOOLID startup parameter's *filepoolid* has a VMSYS prefix (for example, VMSYS, VMSYSU, or VMSYSR), the LOCAL, REMOTE, and SSI startup parameters have no significance. VMSYS file pool ID TPNs are always identified as a local resource, therefore this startup parameter is always a logical LOCAL regardless of whether you specify LOCAL or REMOTE or SSI.

For file pool IDs that do not have a VMSYS prefix, LOCAL indicates the server should not accept a file pool ID connection from outside the processor in which it is running.

For file pool IDs that do not have a VMSYS prefix, REMOTE indicates the server should accept a file pool ID connection from outside the processor in which it is running.

For file pool IDs that do not have a VMSYS prefix, SSI indicates the server should accept a file pool ID connection from outside the processor in which it is running, but only if the request is from another member of the same single system image cluster.

The default is LOCAL.

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

**Note:** The TPNs that identify a CRR recovery server (see the startup parameter description, CRR) are always set up to accept remote connections (from participating resource managers and other CRR recovery servers, not from SFS repository file pool users).

### **NOACCOUNT**

### **ACCOUNT**

ACCOUNT indicates the server should generate accounting records when processing in multiple user mode. (The server never generates accounting records in dedicated maintenance mode—so this startup parameter is ignored in dedicated maintenance mode.) The accounting records are placed in the z/VM system accounting file.

NOACCOUNT indicates no accounting records should be generated in multiple user mode.

The default is NOACCOUNT. See [Chapter 12, “Accounting,” on page 223](#) for more information about accounting.

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

### **NOAUDIT AUDIT**

NOAUDIT indicates server security audit processing should not be performed in multiple user mode. (The server never audits security in dedicated maintenance mode—this startup parameter is ignored in dedicated maintenance mode.)

AUDIT indicates security audit processing should be performed in multiple user mode. Security audit processing, in this case, generates a record in an output file for:

- Unsuccessful repository access attempts
- Successful repository access attempts because special authority (for example, administrator access to objects owned by other users)
- Use of the CRR recovery server operator command:
  - CRR ERASE LU
  - CRR ERASE LUWID
  - CRR RESYNC
- Use of these repository file pool server commands:
  - ERASE LUNAME
  - FORCE PREPARED

This audit is equivalent to one started by an AUDIT ON PARTIAL operator command. The audit output is written to the file identified by the *ddname* AUDIT. If an error occurs while opening the output file, server processing stops.

The default is NOAUDIT.

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

### **Note:**

1. Each one of these startup parameters does not apply to, and is ignored by, dedicated maintenance mode processing.
2. The *ddname*=AUDIT file has the following characteristics:

```
RECFM=VB  LRECL=384  BLOCK=4096
```

3. See [“Auditing Security” on page 143](#) for more about auditing file pool and CRR log security. To change the level of auditing (partial or complete) or to selectively audit CRR recovery server operator commands after multiple user mode processing is started, use the operator command, [“AUDIT” on page 362](#).

### **NOCRR CRR**

NOCRR indicates the server is not going to be the designated CRR recovery server.

CRR indicates the server is going to be the designated CRR recovery server. It also can be used to designate the CRR recovery server either during file pool generation or it can make an already generated server the designated CRR recovery server.

When the server detects the CRR startup parameter is in effect, the server automatically issues a \*IDENT for each of the different transaction program names (TPNs) reserved for the following CRR functions:

- CRR logging. (The TPN is identified as a local resource.)

- Communication between this CRR recovery server and AVS. (The TPN is identified as a local resource.)
- Communication between this CRR recovery server and another CRR recovery server on another system for resynchronization. (The TPN is identified as a system resource.)
- Communication between this CRR recovery server and a participating resource manager (such as an SFS repository file pool server). (The TPN is identified as a global resource.) This TPN is derived from the CRR recovery server's LUNAME startup parameter (does not include *netid*).

**Note:** If the *luname* value specified in the LUNAME startup parameter is eight characters (maximum allowable length), the first character in *luname* is truncated to allow for the dot prefix. For example, if you specify:

```
LUNAME mynodeid.myluname
```

the mapping to the \*IDENT value is 'yluname', not 'myluname'.

The four different TPNs are used by the server to identify itself as the CRR recovery server machine. These TPNs are reserved for CRR services and must not be used by another user.

Only one server on a system can specify the CRR startup parameter because one and only one CRR recovery server can be running on a system. An IBM-supplied CRR recovery server can be optionally installed during the z/VM installation process or you can generate your own CRR recovery server in a post installation procedure.

The default is NOCRR.

Each one of these startup parameters is applicable to SFS repository file pool servers and CRR recovery servers. When a repository file pool server is only involved in BFS file space activity, there is no active CRR recovery server involvement. However, if no CRR recovery server exists in the z/VM system, then performance can be degraded in those cases where SFS services are used in conjunction with administering BFS file spaces.

**Note:**

1. If the CRR startup parameter is specified, the server opens (connects) the CRR logs. If this open is unsuccessful, an error message indicating this is issued to the server console and processing is stopped.
2. If you already have a CRR recovery server running and you attempt to bring up another server with the CRR startup parameter, error messages are issued to the server console and processing is stopped.

**NODFSMS**

**DFSMS**

NODFSMS indicates DFSMS/VM is not managing the file pool. This is the default parameter.

DFSMS indicates DFSMS/VM is managing the file pool. Therefore, the DFSMS/VM exit routine will be called at certain times.

**Note:** The VMSYS file pool *must* be up for DFSMS/VM to function correctly.

If the CSL routine handles the DFSMS/VM exits does not exist, a return code indicating the exit should not be called again will be returned.

Specifying the DFSMS parameter indicates you wish DFSMS/VM to manage the file pool. This has implications for the file pool that you should be aware of, including:

- Backup
- Moving a user to a different storage group
- Enrolling users

The use of DFSMS/VM to manage a file pool should be part of an overall repository storage management strategy. For more information about DFSMS/VM and how it can help you manage

repository file pools, see [z/VM: DFSMS/VM Planning Guide](#) and [z/VM: DFSMS/VM Storage Administration](#).

If you specify DFSMS and DFSMS/VM is not installed, you will receive the following error messages:

```
DMS2030I Initialization error in exit routine SMSDFSMS
Return code = -7, reason code = 0
```

This parameter is ignored in dedicated maintenance mode.

For dedicated CRR recovery servers or dedicated FIFO servers, you should always specify, or let the value default, to NODFSMS. Note that you must not specify DFSMS for the VMSYS file pool. If you do, the server will stop and you will receive the following error message:

```
DMSSSH2030I Initialization error in exit routine SMSDFSMS
Return code = 12, reason code = 100
```

Additional system setup is required to install DFSMS. For more information, see [z/VM: DFSMS/VM Customization](#).

**Note:** If the DFSMS parameter is specified, FILESERV processing uses RTNLOAD to load FSMINT and VMLIB, libraries containing CSL routines reserved for IBM use. These libraries must be present on a file mode accessed by the repository file pool server. If either is not found, message

```
DMS1136E Unable to gain access to library libname
```

is issued and FILESERV processing continues as if NODFSMS had been specified. If you wish to run with DFSMS enabled, take corrective action by first stopping the server using the STOP NOBACKUP operator command. Ensure the CSL library identified in the message is present on an accessed file mode, and restart the server using the FILESERV START command.

### NOSECURITY

### ESECURITY

NOSECURITY indicates file pool access is not controlled by an external security manager (ESM).

ESECURITY indicates file pool access is controlled by an ESM. The server looks for a file called DMSESM PROFILE, which defines the interface with the ESM. This file contains three types of records:

1. The name of the CSL routine the server calls during initialization and termination of the system. You can also specify in this record if you want to allow the ESM to defer a security check back to the usual SFS authorizations.

**Note:** The initialization and termination CSL routine can interface directly with the ESM or through a portion of the ESM code, called the security adapter, loaded into the server machine. The initialization and termination CSL routine specified in the supplied DMSESM PROFILE calls a module named RPIUCMS to enable the ESM. However, RPIUCMS MODULE is not supplied with z/VM. If you plan to use the RPIUCMS interface to the ESM, you must provide the RPIUCMS MODULE. (The ESM might supply its own RPIUCMS MODULE.) If RPIUCMS is missing or cannot be invoked, server processing ends with an error.

2. A list of the types of calls the ESM is to review, including the name of the CSL routine used for each type of call. Four types of calls can be directed to the ESM:
  - SFS object authorization check
  - SFS command authorization check
  - SFS operator command authorization check
  - ESM program check
3. A list of specific file pool requests, if any, the ESM must review.

You can tailor the DMSESM PROFILE to meet the needs of your installation. You can also customize the ESM interface by modifying the supplied CSL routines or writing your own.

Prior to specifying ESECURITY, you should review the MAXCONN parameter of the OPTION control statement in the z/VM system directory entry for the server machine. (See [z/VM: CP Planning and](#)

*Administration* for details.) Depending on how many connections the ESM uses, you might want to add some connections to MAXCONN.

ESMs are not used in dedicated maintenance mode. The ESECURITY startup parameter is ignored if specified for a dedicated maintenance mode command.

The default is NOESECURITY.

The ESECURITY startup parameter is applicable to all file pool servers, but it is recommended that dedicated CRR recovery and dedicated FIFO servers use the default of NOESECURITY.

For more information about external security managers, see [“Using an External Security Manager” on page 161](#).

## **NOETRACE**

### **ETTRACE**

NOETRACE indicates server external trace processing should not be performed.

ETTRACE indicates server external trace processing should be performed for all server processing beginning with initialization. If ETRACE processing cannot be started, the server displays an error message and ends.

The default is NOETRACE.

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

### **Note:**

1. You should use the ETRACE facility only when requested by the designated support group for your installation.
2. Refer to the operator command, [“ETTRACE” on page 429](#) for a description of server external trace processing.
3. If the server is started in multiple user mode, the ETRACE parameter causes tracing of server functions at the most complete level of detail. The only reason you should specify ETRACE for a multiple user mode startup (FILESERV START) is if you want to trace server initialization. Otherwise, you should use the ETRACE operator command in multiple user mode to start and stop external tracing.

If you do specify ETRACE in multiple user mode, you can use the ETRACE OFF operator command to stop the complete tracing, and then enter ETRACE ON to resume a selective trace. (ETTRACE ON prompts you for specific subcomponents and functions.)

4. If the server is started in dedicated maintenance mode, prompts will be issued to allow the type of subsequent external trace processing to be specified. Refer to the ETRACE operator command for a description of the valid prompt responses.

## **NOITRACE**

### **ITRACE *buffersize***

NOITRACE indicates server internal trace processing should not be performed.

ITRACE *buffersize* indicates server internal trace processing should be performed for all server processing beginning with initialization.

The buffer size is the amount of virtual storage in the server machine to be used for internal tracing. The *buffersize* value represents the number of bytes to be used for the buffer in 1024-byte (1KB) units. The server rounds the amount to the next highest multiple of 4096 (4KB) bytes. For example, a *buffersize* value of 1, 2, 3, or 4 will result in a 4096-byte buffer while a value of 5, 6, 7, or 8 will result in a 8192-byte (8KB) buffer. If you specify a buffer size value, it must be equal to, or greater than 1.

If a buffer size value is not specified, a default value of 16 is used. This results in a 16384-byte (16KB) buffer. The maximum buffer size is 2097148KB. If the ITRACE buffer cannot be acquired, the server displays a message and stops.

The default is NOITRACE.



## File Pool Server Startup Parameters

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

### Note:

1. You should use the ITRACE facility only when requested by the designated support group for your installation.
2. Refer to the operator command, [“ITRACE” on page 542](#) for a description of server internal trace processing.
3. This startup parameter does not apply to and is ignored by dedicated maintenance mode processing.

### NOLUNAME

#### LUNAME *luname*

LUNAME *luname* supplies the fully qualified LU name to this CRR recovery server. Specify NOLUNAME when the file pool server does not act as a CRR recovery server. The LUNAME startup parameter is required at FILESERV GENERATE if the CRR startup parameter is specified, and it is also required when FILESERV CRRLOG is entered. This startup parameter is ignored at all other times.

The default is NOLUNAME.

The LUNAME startup parameter is required for CRR recovery servers. It is recommended that the other file pool servers use the default of NOLUNAME.

### Note:

1. The fully qualified LU name must be in the form:



### where:

- *netid* is an optional SNA network ID of 0 to 8 characters in length followed by a period. (The period is optional if *netid* is omitted.) A *netid* is required if this CRR recovery server is part of an SNA network.
- *luname* is an LU name 1 to 8 characters in length.

The *netid* or *luname* must consist of EBCDIC uppercase letters (A-Z), numerics (0-9), \$, #, or @. The first character must be nonnumeric.

2. **This fully qualified LU name MUST be unique in the SNA network.** In other words, you must not have two recovery servers with the same name. This fully qualified LU name will be used internally whenever the CRR recovery server has to exchange log names with another CRR recovery server or participating resource manager, and when generating unique logical unit of work IDs (LUWIDs). The fully qualified LU name specified must be a legitimate LU name for your system.

If the environment changes, the LU name may have to be changed. For example, assume your system was local and you did not specify an SNA network ID. If your system became part of an SNA network, you would have to specify an SNA network ID in the LU name.

The LU name can only be changed by supplying a new one when running either FILESERV GENERATE or FILESERV CRRLOG. The LU name must adhere to the format of SNA network name definitions. See [z/VM: Connectivity](#).

3. **This LU name also derives one of the CRR recovery server's reserved TPNs.** When the LU name is eight characters in length, the leftmost character is replaced with a dot (.) in the formation of the CRR recovery server's derived TPN. If the remaining seven characters are not unique in the scope of the TSAF or CS collection, you will get message DMS3365E when the CRR recovery server attempts to identify this reserved TPN using the \*IDENT function at CRR recovery server initialization time. If this occurs, one of the CRR recovery servers (the one already initialized or the one encountering the DMS3365E message) must change its LU name to one that will permit the unique TPN derivation. The procedure for a CRR recovery server to make this LU name change



is described above in note 2. See the description of the startup parameter, CRR for a list of the reserved CRR TPNs.

Rules for creating LU names if the CRR recovery server resides on a system that is:

- Part of an SNA network:
    - The *netid* must be the same as an SNA network ID value defined to VTAM for this processor.
    - The *luname* must be the same as an LU name defined to VTAM that represents one of the AVS gateways for this processor.
    - The *netid.luname* must be unique among CRR recovery servers and their non-z/VM LU 6.2 sync point equivalents.
  - Part of a TSAF or CS collection:
    - The *netid* can be omitted (but you may specify the SNA network ID if you know in the future your system will be part of an SNA network).
    - The *luname* must be the same as the node ID associated with this processor.
    - The *luname* must be unique among CRR recovery servers.
- Note:** If you can connect to another system within your TSAF or CS collection that has AVS, you have access to the SNA network. In this case, you should follow the rules for a system that is part of an SNA network.
- Local only (not part of a TSAF or CS collection or an SNA network):
    - The *netid* can be omitted (but you may specify the SNA network ID if you know in the future your system will be part of an SNA network).
    - The *luname* can be any valid name.

For information on changing LU names, see [“Remove Designation of Alternate CRR Recovery Server” on page 320.](#)

## NOMSGS

### MSG

NOMSGS indicates informational server startup and processing messages should not be written to the server machine console. Messages such as those that display the contents of the *filepoolid* POOLDEF file are not written when the NOMSGS parameter is in effect.

MSG indicates the informational messages should be displayed.

The default is NOMSGS.

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

## NORESTORE

### RESTORE

NORESTORE indicates the file pool control data should not be restored during startup processing.

RESTORE indicates the control data should be restored before usual server operation is initiated. When RESTORE is in effect, you must enter a FILEDEF command for *ddname* RESTORE before entering the FILESERV START command. The file identified by *ddname* RESTORE must contain a backup of the control data. That is, the file identified by *ddname* RESTORE should be the same file identified by *ddname* BACKUP in a previous control data back up. The file can be either a disk or tape file. If *ddname* RESTORE is not defined, the server displays an error message and stops.

When FILESERV START is entered, the server replaces the control data with the backup copy in the file identified by *ddname* RESTORE. After restore processing completes successfully, the server immediately initiates usual multiple user mode operation.

The default is NORESTORE.

### Note:

1. This startup parameter does not apply to and is ignored by dedicated maintenance mode processing.
2. When the RESTORE startup parameter is specified:
  - a. FILESERV command processing issues a FILEDEF command for DDNAME=POOLDEF to define the *filepoolid* POOLDEF file.
  - b. FILESERV START processing updates the *filepoolid* POOLDEF file.
  - c. FILESERV START processing creates and usually erases a temporary file named \$\$\$AVED \$POOLDEF using the same file mode as that of the *filepoolid* POOLDEF file. If \$\$\$AVED \$POOLDEF already exists on that file mode, FILESERV START processing displays an error message and stops.
3. If the FILESERV START command ends unusually and the RESTORE parameter was specified:
  - a. If message DMS3292I was displayed, the restore process completed, but the log recovery process did not. In this case, it is not necessary to do the restore over. When you next start multiple user mode processing with the NORESTORE startup parameter, the server will automatically do the log recovery.
  - b. If message DMS3292I was not displayed, the restore process did not complete and needs to be rerun. In this case, if the \$\$\$AVED \$POOLDEF file exists with the same file mode as that of the *filepoolid* POOLDEF file, erase the *filepoolid* POOLDEF file. Then rename \$\$\$AVED \$POOLDEF to *filepoolid* POOLDEF. For example, if the file pool ID is VMSYSU, and the POOLDEF file resides on file mode A, you would enter:

```
erase vmsysu pooldef a
rename $$$aved $pooldef a vmsysu pooldef a
```

4. The RESTORE startup parameter can be specified only if the BACKUP startup parameter is also in effect. If the RESTORE and NOBACKUP startup parameters are specified, an error message is displayed, and FILESERV command processing stops.
5. The RESTORE startup parameter is only used to recover the file pool by restoring control data using the last control data backup. For usual startup, this is not required and the NORESTORE startup parameter should be specified.
6. The only backup file that will restore successfully is the last backup file created. If you try to restore any earlier control data backup, the restore will not succeed. An error message will be displayed, and server processing will stop.
7. This note applies only to repository file pool servers and should be ignored in the case of dedicated CRR recovery servers and dedicated FIFO servers. For best performance when restoring the control data, you should also specify a very large catalog buffer pool. To increase the size of the catalog buffer pool, specify the CATBUFFERS startup parameter.

```
CATBUFFERS 5000
```

If the server machine does not have enough virtual storage available to it, you may need to reduce the CATBUFFERS value.

If you do specify a large catalog buffer pool, you should plan to stop multiple user mode processing soon after the restore completes to reset CATBUFFERS to its usual value.

8. It is convenient to have a startup parameter file used only for restores.

For more instructions on restoring the control data, see [“Restoring Control Data” on page 110](#).

### **RESYNCINTERVAL *timed\_wait***

This startup parameter sets the amount of time (called the timed wait interval) spent in a wait state before the next attempt at resynchronization. Resynchronization attempts to connect to either a CRR recovery server (for protected conversations) or a participating resource manager (such as an SFS repository file pool server) to try to complete an LUWID instance. The cycling through timed waits and attempts at resynchronization is called the automatic periodic retry of resynchronization.

**Note:** An LUWID instance is a subset of a coordinated transaction (also called a CRR logical unit of work). The coordinated transaction is identified by the LUWID. The LUWID instance can consist of one or more resource logical units of work.

Also, the RESYNCINTERVAL startup parameter determines the frequency the CRR recovery server operator is reminded of LUWID instances waiting to be resynchronized.

*timed\_wait* is the number of seconds the CRR recovery server should wait between attempts at completing resynchronization.

If the RESYNCINTERVAL startup parameter is not specified for the CRR recovery server, the default value is 600 seconds (10 minutes).

This startup parameter is applicable to CRR recovery servers.

### **SAVESEGID *savesegname***

#### **NOSAVESEGID**

This startup parameter identifies the location of the server executable code. If omitted or if SAVESEGID is specified without a *savesegname*, CMS executes the server code residing in a physical saved segment named CMSFILES.

If SAVESEGID is specified with *savesegname*, *savesegname* must identify a segment that contains server executable code. If NOSAVESEGID is specified, the server's executable modules are loaded as nucleus extensions.

When you have multiple file pool servers, it is more desirable to specify SAVESEGID. Specifying SAVESEGID allows multiple servers to share one copy of server executable code.

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

### **SHUTDOWN SIGNAL**

#### **NOSHUTDOWN SIGNAL**

SHUTDOWN SIGNAL indicates the server is enabled to receive a shutdown signal from CP. When a CP SHUTDOWN, FORCE, or SIGNAL command is issued, CP sends a shutdown signal to all enabled file pool servers. When an enabled server receives the shutdown signal from CP, the SFS STOP operator command (with no operands) is automatically issued. At the completion of the STOP command, SFS notifies CP the shutdown has completed. If this processing does not complete within a CP-maintained time interval, CP shuts down and SFS is terminated.

NOSHUTDOWN SIGNAL indicates the server is not enabled to receive the shutdown signal.

The default is SHUTDOWN SIGNAL.

Each one of these startup parameters is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

### **USERS *users***

For repository file pool servers, *users* is your best estimate of the number of logged-on repository users expected during peak system activity. A repository user is someone who accesses objects in either SFS or BFS file spaces in a file pool repository server. (The value should be the same as the USERS value used in calculating the MAXCONN value for the server machine as described in [Chapter 15, "Generating a File Pool and Server,"](#) on page 247.)

For CRR recovery servers, *users* is your best estimate of the number of logged-on CRR users you anticipate exploiting CRR by doing sync point processing (two-phase commit). The default of 100 should be sufficient for many installations.

<b>Note:</b>
It is strongly recommended you do not use the same server for both file pool repository and CRR services. However, if you do so, combine the two values for <i>users</i> that would have been used if the file pool repository and CRR services were assigned were to separate servers.

For FIFO servers, *users* is determined as follows:

## File Pool Server Startup Parameters

- If FIFO service is done by the repository server (FIFO startup parameter defaulted), there are no adjustments to *users* for the FIFO service.
- If the current startup is for a file pool server that is doing FIFO services for one or more other file pool servers (ones that specified in their FIFO startup parameter the file pool that is currently being started-up), *users* should include BFS users for each of those other file pool servers, where those BFS users actively employ FIFO objects. That is, you should allow for the total of all potential concurrent BFS FIFO users serviced by the current FIFO server.

If the current file pool server provides both its own repository services and FIFO services for other file pools, *users* should include the combination of the value calculated for current file pool repository users with the value calculated above for other file pool FIFO users.

An error occurs if you specify a numeric value of less than 1 or greater than 32767. The default value is 100.

This startup parameter is applicable to repository file pool servers, CRR recovery servers, and FIFO servers.

### **Note:**

1. Choose *users* carefully. The server optimizes its processing based on this value.
2. In dedicated maintenance mode, this startup parameter is used only to calculate the CATBUFFERS and CTLBUFFERS values. You do not need to change the USERS value when running the server in dedicated maintenance mode.

## Chapter 21. File Pool Administration and Server Commands

This chapter describes the commands that an individual may use to administer SFS and CRR and to operate their servers.

**Note:**

For a description of the possible combinations and types of file pool servers, see [Chapter 1, “File Pool Administration Overview,”](#) on page 3.

Some of the commands can be used with a server regardless of whether it is a repository file pool server, a CRR recovery server or a FIFO server. Other commands are intended only for one type of server.

### Administration and Server Command Types

The file pool commands are divided into three groups:

- administration commands
- server machine commands
- operator commands

### Administration Commands

Administration commands require file pool administration authority and are issued from a user machine that has been granted administration authority. The server must be available for multiple user access. These commands perform a variety of tasks such as enrolling users into a file pool, backing up or restoring users' data, and inquiring about the status of the file pool.

Table 30 on page 357 provides an alphabetic list of the administration commands, their intended user, and where they are located.

Command Name	Intended User
<a href="#">“DATASPACE” on page 396</a>	Repository File Pool Administrator
<a href="#">“DELETE ADMINISTRATOR” on page 400</a>	File Pool Administrator
<a href="#">“DELETE LOCK” on page 403</a>	Repository File Pool Administrator
<a href="#">“DELETE PUBLIC” on page 406</a>	Repository File Pool Administrator
<a href="#">“DELETE USER” on page 408</a>	Repository File Pool Administrator
<a href="#">“ENROLL ADMINISTRATOR” on page 417</a>	File Pool Administrator
<a href="#">“ENROLL PUBLIC” on page 420</a>	Repository File Pool Administrator
<a href="#">“ENROLL USER” on page 422</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL BACKUP” on page 432</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL CLEANUP” on page 437</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL CONTROL BACKUP” on page 439</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL DISABLE” on page 443</a>	Repository File Pool Administrator

<i>Table 30. File Pool Administration Commands (continued)</i>	
<b>Command Name</b>	<b>Intended User</b>
<a href="#">“FILEPOOL ENABLE” on page 447</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL FILELOAD” on page 451</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL FORMAT AUDIT” on page 455</a>	File Pool Administrator
<a href="#">“FILEPOOL LIST BACKUP” on page 459</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL LIST MINIDISK” on page 467</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL MINIDISK” on page 470</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL RENAME” on page 484</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL RELOAD” on page 476</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL RESTORE” on page 488</a>	Repository File Pool Administrator
<a href="#">“FILEPOOL UNLOAD” on page 495</a>	Repository File Pool Administrator
<a href="#">“MODIFY USER” on page 544</a>	Repository File Pool Administrator
<a href="#">“QUERY ACCESSORS (SFS only)” on page 547</a>	Repository File Pool Administrator
<a href="#">“QUERY DATASPACE (SFS only)” on page 550</a>	Repository File Pool Administrator
<a href="#">“QUERY FILEPOOL AGENT” on page 557</a>	File Pool Administrator
<a href="#">“QUERY FILEPOOL CATALOG” on page 560</a>	File Pool Administrator
<a href="#">“QUERY FILEPOOL CONFLICT” on page 563</a>	Repository File Pool Administrator
<a href="#">“QUERY FILEPOOL COUNTER” on page 569</a>	File Pool Administrator
<a href="#">“QUERY FILEPOOL CRR (CRR only)” on page 574</a>	CRR Administrator
<a href="#">“QUERY FILEPOOL DISABLE” on page 577</a>	File Pool Administrator
<a href="#">“QUERY FILEPOOL LOG” on page 583</a>	File Pool Administrator
<a href="#">“QUERY FILEPOOL MINIDISK” on page 586</a>	File Pool Administrator
<a href="#">“QUERY FILEPOOL OVERVIEW” on page 589</a>	File Pool Administrator
<a href="#">“QUERY FILEPOOL REPORT” on page 592</a>	File Pool Administrator
<a href="#">“QUERY FILEPOOL STATUS” on page 623</a>	File Pool Administrator
<a href="#">“QUERY FILEPOOL STORGRP” on page 652</a>	File Pool Administrator
<a href="#">“QUERY LIMITS” on page 655</a>	Repository File Pool Administrator
<a href="#">“SET THRESHOLD” on page 664</a>	Repository File Pool Administrator
<p><b>Note:</b> The following administration commands can also be entered by any connected user, but the output may not be useful to them: QUERY FILEPOOL AGENT, QUERY FILEPOOL CONFLICT, QUERY FILEPOOL COUNTER, QUERY FILEPOOL DISABLE, QUERY FILEPOOL LOG, QUERY FILEPOOL MINIDISK, QUERY FILEPOOL OVERVIEW, QUERY FILEPOOL REPORT, QUERY FILEPOOL STATUS, QUERY FILEPOOL STORGRP, QUERY LIMITS.</p>	

## Server Machine Commands

Server machine commands are all FILESERV *operand* commands. (For example, FILESERV START). The FILESERV command is issued from the server machine operator console and performs maintenance

operations on a file pool. When FILESERV is running against a file pool, the file pool is not available to other users.

**Note:** FILESERV START makes the file pool available to other users.

See Table 31 on page 359 for an alphabetic list of the server machine commands, their intended user, and where they are located.

Command Name	Intended User
<a href="#">“FILESERV BACKUP” on page 500</a>	Repository File Pool Server Machine
<a href="#">“FILESERV CRRLOG” on page 502</a>	CRR Server Machine
<a href="#">“FILESERV DEFAUDIT” on page 504</a>	File Pool Server Machine
<a href="#">“FILESERV DEFBACKUP” on page 506</a>	Repository File Pool Server Machine
<a href="#">“FILESERV DEFCRRLOG” on page 509</a>	CRR Server Machine
<a href="#">“FILESERV FIXCENT” on page 511</a>	File Pool Server Machine
<a href="#">“FILESERV GENERATE” on page 513</a>	File Pool Server Machine
<a href="#">“FILESERV LIST” on page 521</a>	Repository File Pool Server Machine
<a href="#">“FILESERV LOG” on page 523</a>	File Pool Server Machine
<a href="#">“FILESERV MINIDISK” on page 525</a>	Repository File Pool Server Machine
<a href="#">“FILESERV MOVEUSER” on page 528</a>	Repository File Pool Server Machine
<a href="#">“FILESERV REGENERATE” on page 531</a>	Repository File Pool Server Machine
<a href="#">“FILESERV REORG” on page 535</a>	Repository File Pool Server Machine
<a href="#">“FILESERV START” on page 537</a>	File Pool Server Machine

## Operator Commands

These commands are issued from the server machine operator console for the following tasks:

- Control multiple user mode execution
- Log management
- Problem management
- CRR resynchronization management

See Table 32 on page 359 for an alphabetic list of the operator commands, their intended user, and where they are located.

Command Name	Intended User
<a href="#">“AUDIT” on page 362</a>	File Pool Operator
<a href="#">“BACKUP” on page 366</a>	Repository File Pool Operator
<a href="#">“CRR ERASE LU” on page 368</a>	CRR Operator
<a href="#">“CRR ERASE LUWID” on page 370</a>	CRR Operator
<a href="#">“CRR QUERY LOG” on page 371</a>	CRR Operator
<a href="#">“CRR QUERY LOGTABLE” on page 373</a>	CRR Operator

<i>Table 32. File Pool Operator Commands (continued)</i>	
<b>Command Name</b>	<b>Intended User</b>
<a href="#">“CRR QUERY LU” on page 376</a>	CRR Operator
<a href="#">“CRR QUERY LUWID” on page 382</a>	CRR Operator
<a href="#">“CRR RESUME” on page 388</a>	CRR Operator
<a href="#">“CRR RESYNC” on page 390</a>	CRR Operator
<a href="#">“CRR SUSPEND” on page 394</a>	CRR Operator
<a href="#">“DEFBACKUP” on page 398</a>	Repository File Pool Operator
<a href="#">“DISABLE” on page 412</a>	Repository File Pool Operator
<a href="#">“ENABLE” on page 415</a>	Repository File Pool Operator
<a href="#">“ERASE LUNAME” on page 427</a>	Repository File Pool Operator
<a href="#">“ETRACE” on page 429</a>	File Pool Operator
<a href="#">“FORCE” on page 539</a>	Repository File Pool Operator
<a href="#">“GRANT ADMIN” on page 541</a>	File Pool Operator
<a href="#">“ITRACE” on page 542</a>	File Pool Operator
<a href="#">“QUERY DEFBACKUP” on page 553</a>	Repository File Pool Operator
<a href="#">“QUERY DISABLE” on page 555</a>	Repository File Pool Operator
<a href="#">“QUERY LOGTABLE” on page 658</a>	Repository File Pool Operator
<a href="#">“QUERY PREPARED” on page 660</a>	Repository File Pool Operator
<a href="#">“REVOKE ADMIN” on page 663</a>	File Pool Operator
<a href="#">“STOP” on page 666</a>	File Pool Operator

## Using the Online HELP Facility

You can receive online information about the commands described in this document using the z/VM HELP Facility. For example, to display a menu of SFS/CRR administration and operator commands, enter:

```
help sfsadmin menu
```

To display information about a specific SFS/CRR administration and operator command (MODIFY in this example), enter:

```
help sfsadmin modify
```

You can also display information about a message by entering one of the following commands:

```
help msgid or help msg msgid
```

For example, to display information about message DMS001E, you can enter one of the following commands:

```
help dms001e or help msg dms001e
```

For more information about using the HELP Facility, see [z/VM: CMS User's Guide](#). To display the main HELP Task Menu, enter:

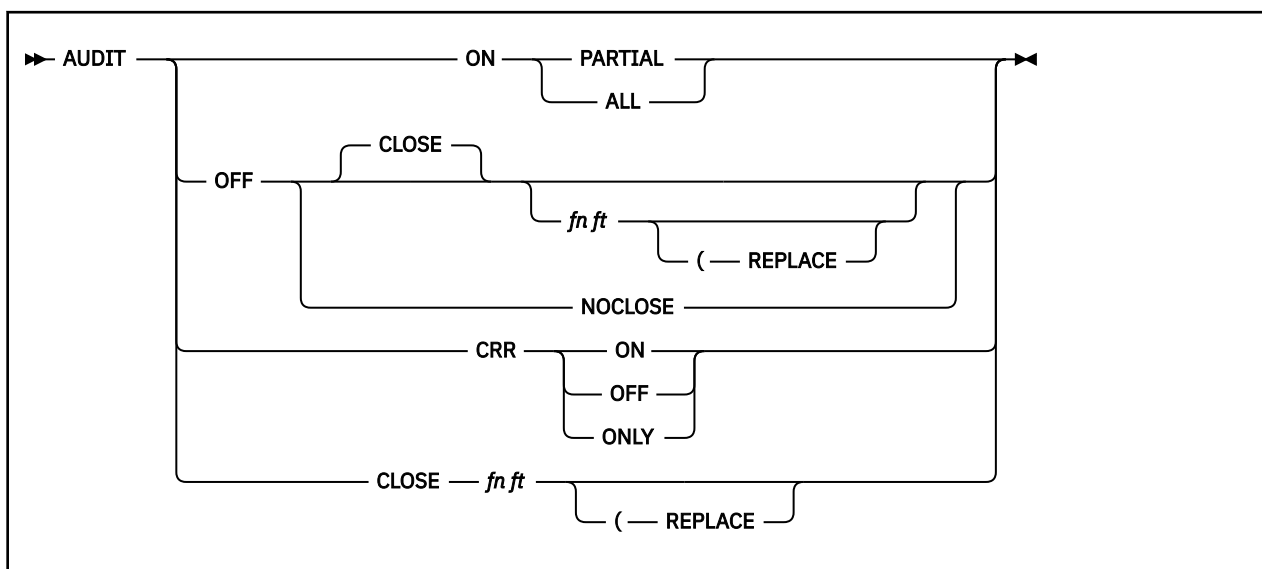
```
help
```



For more information about the HELP command, see [z/VM: CMS Commands and Utilities Reference](#) or enter:

```
help cms help
```

## AUDIT



### Authorization

File Pool Operator

### Purpose

Use the AUDIT operator command to start or stop server security audit trace processing of the file pool server.

### Operands

#### ON

indicates security audit trace processing should be performed.

#### PARTIAL

indicates the server should trace only:

- SFS and Byte File System (BFS) access attempts that are unsuccessful
- All SFS access attempts by users having file pool administration authority
- All BFS access attempts by users having appropriate privileges
- Use of the CRR recovery server operator command:
  - CRR ERASE LU
  - CRR ERASE LUWID
  - CRR RESYNC
- Use of these SFS file pool server operator commands:
  - ERASE LUNAME
  - FORCE PREPARED

#### ALL

indicates the server should trace both successful and unsuccessful accesses to any SFS and BFS objects, and the use of the CRR recovery server operator commands:

- CRR ERASE LU
- CRR ERASE LUWID

- CRR RESYNC

and these file pool server operator commands:

- ERASE LUNAME
- FORCE PREPARED

You should specify ALL only when essential because the server must write an audit record for every SFS and BFS access attempt, which could cause poor performance for user requests.

#### **OFF**

stops security audit trace processing for the server.

#### **CLOSE**

indicates the file being used for security audit output should be closed. This file, known as the *security audit trace file*, is identified by the DDNAME=AUDIT control statement in the POOLDEF file. You can change the output file specified in the POOLDEF file by using the FILESERV DEFAUDIT command.

When OFF is specified, CLOSE is the default.

#### ***fn ft***

is the file name and file type of a file to be created to capture a "snapshot" of the security audit trace file. This prevents overwriting the security audit trace file and allows you to get audit information while auditing remains on. For more information, see usage note ["11" on page 365](#).

#### **(REPLACE**

indicates the audit "snapshot" file being created should replace any existing file with that file ID.

#### **NOCLOSE**

indicates the security audit trace file should be left open. Any subsequent AUDIT ON PARTIAL or AUDIT ON ALL command would then continue to append security audit trace data to the same data file. If left open, the data file would be closed during shut down processing.

#### **CRR**

modifies current audit settings regarding the audit trace of the following CRR recovery server operator commands:

- CRR ERASE LU
- CRR ERASE LUWID
- CRR RESYNC

and these SFS file pool server operator commands:

- ERASE LUNAME
- FORCE PREPARED

(AUDIT ON PARTIAL or AUDIT ON ALL must already be in effect for AUDIT CRR to work.)

#### **CRR ON**

indicates audit trace processing will now start collecting CRR audit records, which traces the use of the following CRR recovery server operator commands:

- CRR ERASE LU
- CRR ERASE LUWID
- CRR RESYNC

and these SFS file pool server operator commands:

- ERASE LUNAME
- FORCE PREPARED

Because both partial and complete audits collect CRR audit records, this operand is only meaningful after you have stopped collecting CRR audit records (AUDIT CRR OFF) and now want to start collecting them again.

### CRR OFF

indicates audit trace processing will now stop collecting CRR audit records.

### CRR ONLY

indicates audit trace processing will now collect only CRR audit records.

## Usage Notes

1. All messages are displayed on the server operator console.
2. AUDIT ON PARTIAL or AUDIT ON ALL starts server security audit trace processing, which traces accesses to the file pool, files, directories, use of file pool operator commands that intervene into CRR activity by changing SFS and CRR logs.
3. AUDIT ON PARTIAL or AUDIT ON ALL processing writes an audit trace record for any attempted connection to a file pool that is unsuccessful because of a lack of authorization. An attempt to connect to a file pool will be unsuccessful if the user is not explicitly enrolled and PUBLIC is not enrolled in the file pool. (When PUBLIC is enrolled, all users are authorized to connect.)
4. If AUDIT ON PARTIAL processing is already in effect and AUDIT ON PARTIAL is entered again, a warning message will be written to the operator console (message DMS3470W).  
  
If AUDIT ON ALL processing is in effect and AUDIT ON PARTIAL is entered, an informational message will be written and AUDIT ON PARTIAL processing will be initiated (message DMS3471I).
5. If AUDIT ON ALL processing is already in effect and AUDIT ON ALL is entered again, a warning message will be written to the operator console (message DMS3470W).  
  
If AUDIT ON PARTIAL processing is in effect when AUDIT ON ALL is entered, an informational message will be written and AUDIT ON ALL processing will be initiated (message DMS3471I).
6. Trace output will be written to the security audit trace data file identified by the ddname=AUDIT. File pool server security audit trace processing will not be started if a FILEDEF command for a ddname of AUDIT is not in effect when the command is issued.  
  
You do not, however, need to enter a FILEDEF command for ddname=AUDIT. Instead, FILESERV command processing automatically executes one for you before the start of multiple user mode operation. FILESERV command processing builds the FILEDEF command from data contained in the POOLDEF file. To define, change, or delete the file to be used for security audit trace output, use the command, "FILESERV DEFAUDIT" on page 504. Variable-length records are written.
7. Security audit trace data file information can be formatted for display by using the command, "FILEPOOL FORMAT AUDIT" on page 455.
8. When the CLOSE operand is specified or allowed to default, the server attempts to close the audit trace file. If that file resides in another file pool and there is insufficient DASD storage in the target storage group or an insufficient number of unused blocks in the target file space, the server will stop. Either of these conditions would also cause the target server to roll back the audit file updates. You lose all audit information since the audit file was opened.  
  
You can minimize the chance of having the server stop by ensuring there is adequate physical storage in the storage group and an adequate number of unused blocks in the file space. You should also consider restricting the write activity to the file space, so other users do not consume the blocks intended to hold the audit trace file.
9. AUDIT CRR ON processing writes a CRR audit record when an operator issues a command that alters the CRR logs or file pool repository logs. Partial audits (AUDIT ON PARTIAL) and complete audits (AUDIT ON ALL) already collect CRR audit records. So AUDIT CRR ON is only meaningful after you have stopped collecting CRR audit records (AUDIT CRR OFF) and now want to start collecting them again.
10. AUDIT CLOSE *fn ft* closes the security audit trace file but does not change the auditing state. If auditing is ON, it remains ON. If auditing is OFF NOCLOSE, it becomes OFF CLOSE. Note that you must specify the file name and file type of the audit "snapshot" file. In a continuous operation, you should use AUDIT CLOSE *fn ft* periodically to capture audit records in a file you can then process offline. For

more information about specifying the file name and file type of a "snapshot" file, see usage note "11" on page 365.

11. The purpose of using the *fn ft* operands is to protect the security audit trace file. When you specify AUDIT OFF CLOSE or AUDIT CLOSE, the server closes the security audit trace file with the name specified in the DDNAME=AUDIT control statement in the POOLDEF file. If auditing continues or is started again, the security audit trace file is overwritten by subsequent records.

To avoid this, you can specify a file name and file type along with the close. This causes the server to rename the security audit trace file to the file name and file type you specify, creating a "snapshot" of the audit information. If auditing continues or is started again, the new records are written to a new security audit trace file using the original name.

If a file with the specified file name and file type already exists, you must specify the REPLACE option. In that case, the existing file is erased.

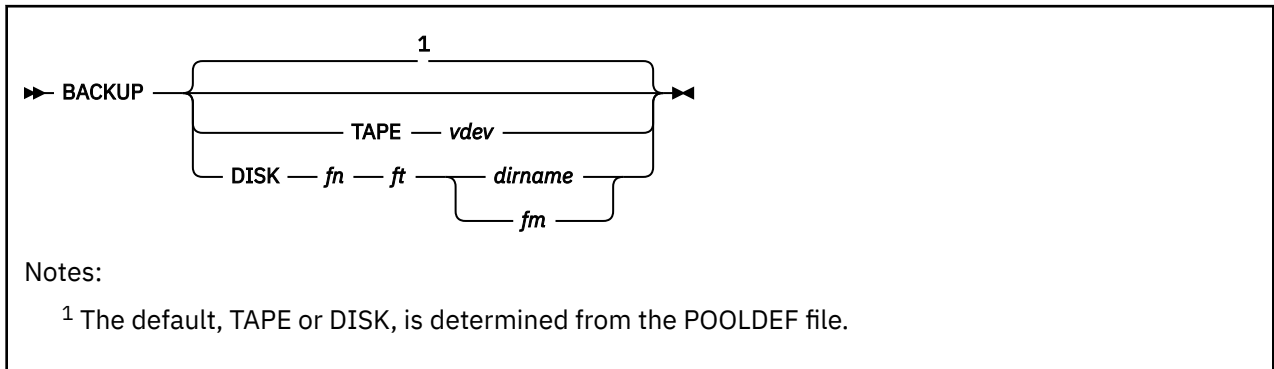
You can create an audit "snapshot" file only when all of the following are true:

- The security audit trace file is open. That means audit is either ON or in the OFF NOCLOSE state.
- The security audit trace file is an SFS or minidisk file.
- The security audit trace file is not empty. That is, at least one auditable event has occurred since auditing was started.

If any of these conditions are not met, the command fails with error message DMS3254E.

The *fn ft* operands are optional on AUDIT OFF CLOSE. However, if you do not specify them, you risk overwriting the existing security audit trace file when you restart auditing.

## BACKUP



### Authorization

Repository File Pool Operator

### Purpose

Use the BACKUP operator command to start a backup of the control data while multiple user mode processing continues.

### Operands

#### TAPE *vdev*

identifies the virtual device number *vdev* for a standard label (SL) tape file. The control data backup will be directed to the identified tape device. If an assignment for the control data backup file already exists, it is temporarily replaced.

#### DISK *fn ft dirname*

#### DISK *fn ft fm*

identifies a CMS file to which the control data backup will be directed. The file can reside on a minidisk or in another file pool. The *dirname* is the fully qualified directory name *filepoolid:userid.n1.n2..n8* of the SFS directory where the backup file is to be created. The *fm* is the file mode of an accessed minidisk or SFS directory. If an assignment for the control data backup file already exists, it is temporarily replaced.

### Usage Notes

1. For a list of supported virtual device numbers for tapes, see the description of the FILEDEF command in *z/VM: CMS Commands and Utilities Reference*.
2. All messages are written to the server operator console.
3. For CRR recovery servers, IBM does not recommend doing backups of control data (that is, use the NOBACKUP startup parameter).
4. Server processing will create the control data backup file either at the default destination currently in effect, or at the destination explicitly specified on the BACKUP operator command itself.

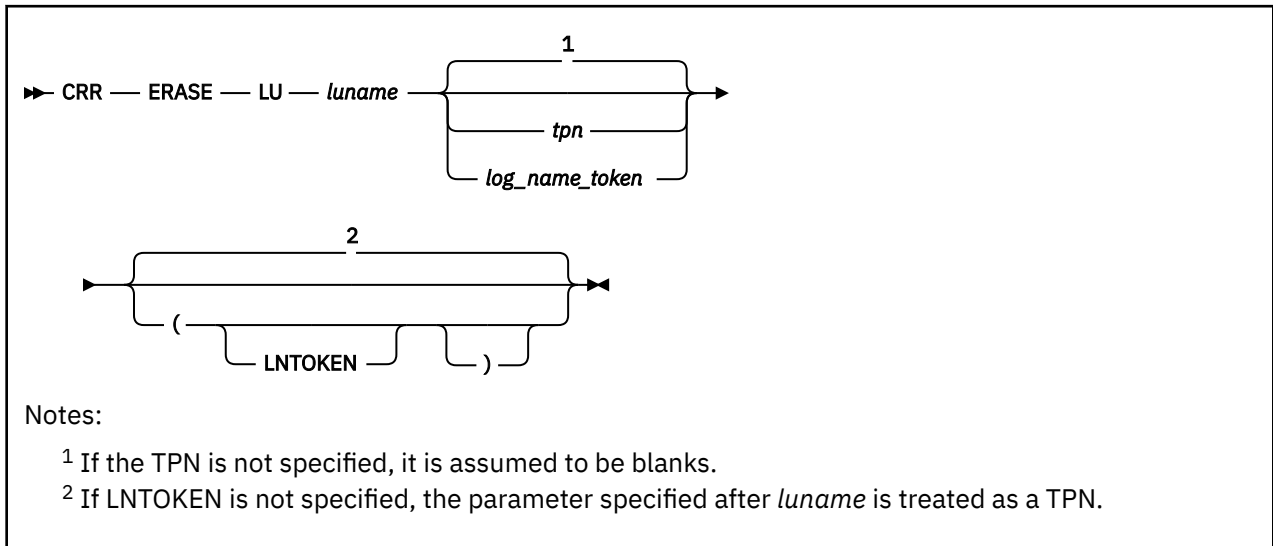
The default destination for the backup file is originally determined from the contents of the POOLDEF file. If you want to define or permanently change the file to which backups are directed, use the command, “FILESERV DEFBACKUP” on page 506. The FILESERV DEFBACKUP command updates the POOLDEF file backup related control statements. If you want to change the default destination for the backup file while the server is running in multiple user mode, you can use the operator command, “DEFBACKUP” on page 398.

5. You do not need to enter a FILEDEF command for The server creates the proper environment for the backup processing. An error message is issued if something goes wrong and the backup fails.

6. When the control data backup file resides on a minidisk, BACKUP processing uses a temporary file during the backup. This file is named \$\$TEMP \$BACKUP. If backup processing completes successfully, the file is erased. If backup processing fails and the backup file resides on a minidisk, follow the recovery actions in [“Special Considerations for Control Data Backups to CMS File”](#) on page 106.
7. The BACKUP command will not succeed unless the BACKUP startup parameter is in effect. An error message is displayed if the BACKUP command is entered and the NOBACKUP startup parameter is in effect.
8. When the server has written the control data to the backup file, it tries to close the backup file. If that file resides in another file pool and there is insufficient DASD storage in the target storage group or an insufficient number of unused blocks in the target file space, the server may stop.  

You can minimize the chance of having the server stop by ensuring there is adequate physical storage in the storage group and an adequate number of unused blocks in the file space. You should also consider restricting the write activity to the file space, so other users do not consume the blocks intended to hold the control data backup.
9. When the backup file is created in another file pool, the performance degradation to the users of the file pool being backed up, is less than it would have been if the file were created on tape or minidisk. However, the backup process itself takes longer to complete. For more information, see [“Special Considerations for Control Data Backups to File Pool”](#) on page 105.
10. After backup processing successfully completes, the server verifies the current default destination for backup files is valid. If a problem is detected, a warning message is issued, telling you to either correct the problem or provide a valid destination for the next backup.
11. Explicitly redirecting the backup file by specifying a new destination on the BACKUP command does not change the current default destination.
12. If backup was started automatically (because the log was over 80% full) and it failed, the default control data backup file destination is invalidated and you will have to specify the backup file destination explicitly on the BACKUP or STOP BACKUP operator command. Depending on what the problem is, you could either correct it and specify the same destination, or you could try to redirect the control data file (for example, from minidisk to tape). In any event, if the problem is not corrected by the time the logs become 95% full, the server will stop. If this happens, process FILESERV BACKUP and start the server again.
13. If the default control data backup file destination was invalidated, you can establish a new default destination with the help of the DEFBACKUP operator command.
14. You can find out the current default assignment for the control data backup file, or where the last successful control data backup file was created, by issuing the QUERY DEFBACKUP operator command.
15. If the control backup is directed to tape, it is assumed the tape will be ready when the backup starts. If it is not, message DMS113S will be issued and control backup will stop.
16. It is not allowed to direct the control backup to a file that already has established data definitions using the FILEDEF command. If this is attempted, backup processing will stop. Note that the file pool server issues a FILEDEF command for security audit (AUDIT) when AUDIT is specified in the DMSPARMS file.

## CRR ERASE LU



### Authorization

CRR Operator

### Purpose

Use the CRR ERASE LU operator command to erase the specified LU name and transaction program name (TPN) entries from the CRR log name table.

**Note:** The TPN is sometimes called the resource ID.

### Operands

#### *luname*

is the fully qualified LU name of a protected resource or protected conversation that consists of:

- SNA network ID
- LU name

If both the protected resource and the CRR recovery server reside on the same processor or TSAF or CS collection, you must specify *luname* as \*LOCAL. If you specify \*LOCAL, you must also specify *tpn*.

#### *tpn*

is the TPN of a protected resource, such as a file pool server. Protected conversations do not have TPNs in the CRR log name table.

#### *log\_name\_token*

is the token associated with a log name entry. If the TPN contains unprintable characters, the log name token associated with the protected resource must be specified. (Unprintable characters are any character except the following: A-Z, a-z, 0-9, @, #, \$, . (period).)

#### LNTOKEN

indicates the TPN contained unprintable characters and a log name token is being used to uniquely identify the LU to be erased.

### Usage Notes

1. The input log name token may be from 1 to 6 characters, including leading zeros.



2. The CRR ERASE LU command can only be used if there is no current sync point or resynchronization processing for the specified LU name.

If there is current sync point or resynchronization processing for the specified LU name, an error is issued and processing for this command terminates without deleting the CRR log name table entry.

3. The optional TPN or log name token, which is obtained from the output of the CRR QUERY LOGTABLE command, must be used if the log name entry in the CRR log name table is for a protected resource.
4. If the TPN is not specified, it is assumed to be blanks.
5. If an LU name or a protected resource are redefined or otherwise 'deleted' so they no longer exist, this command can be used to release space taken up in the CRR log name table by the log name for the deleted protected resource.
6. Use the CRR QUERY LOGTABLE command to determine the LU names and TPNs in the CRR log name table. The CRR log name table resides on the CRR logs.
7. If CRR security audit tracing is enabled, the CRR ERASE LU command generates an audit trace record.
8. The log name tokens may have been resequenced since the last CRR QUERY LOGTABLE command was entered. Therefore, you may want to enter the CRR QUERY LOGTABLE just prior to issuing the CRR ERASE LU command.

## CRR ERASE LUWID

---

►► CRR — ERASE — LUWID — *token* ◄◄

### Authorization

CRR Operator

### Purpose

Use the CRR ERASE LUWID operator command to erase the CRR log records associated with the specified instance of the logical unit of work ID (LUWID). This will prevent resynchronization activity from occurring for the LUWID instance.

### Operands

#### *token*

identifies the LUWID instance that will no longer have resynchronization activity. The *token* could be obtained from one of the following:

- Resynchronization messages displayed at the operator's console of this CRR recovery server
- Output of the CRR QUERY LU command
- Output of the CRR QUERY LUWID command

**Note:** The value in *token* identifies an LUWID instance. An LUWID instance is a subset of a coordinated transaction (also called a CRR logical unit of work). The coordinated transaction is identified by the LUWID. The LUWID instance can consist of one or more resource logical units of work.

### Usage Notes

1. The CRR ERASE LUWID command can only be used if there is no current sync point or resynchronization processing for the specified LUWID instance. The LUWID instance must be in resynchronization pending process.

If there is current sync point or resynchronization processing for the specified LUWID instance, an error is issued and processing for this command terminates without deleting the CRR log records.

2. Before issuing this command, it may be necessary to first use the CRR RESYNC command and participating resource manager commands (such as SFS's QUERY PREPARED and FORCE PREPARED commands) to manually complete resynchronization for the LUWID instance.
3. The operator must be very careful when using this command because it erases logged CRR information and could potentially add to the inconsistency of the coordinated transaction.
4. If CRR security audit tracing is enabled, the CRR ERASE LUWID command generates an audit trace record.

## CRR QUERY LOG

► CRR — QUERY — LOG ◄

### Authorization

CRR Operator

### Purpose

Use the CRR QUERY LOG operator command to display the status of the CRR log minidisks.

### Usage Notes

1. If the CRR log name table starts to become full (for example, 95% of the CRR log name table space is used up), you may want to free up space in the CRR log name table by issuing:
  - CRR QUERY LOGTABLE command to determine the LU names and transaction program names (TPNs) that are in the CRR log name table.
 

**Note:** The TPN is sometimes called resource ID.
  - CRR ERASE LU command to erase unwanted LU name and TPN entries from the CRR log name table.
 

For more information on freeing space in the CRR log name table, see [“Free Space in the CRR Log Name Table”](#) on page 323.
2. If one of the two CRR log minidisks is disabled, you may want to stop the CRR recovery server and replace the disabled CRR log minidisk. For more information on replacing one CRR log minidisk, see [“Replace One CRR Log Minidisk”](#) on page 328.
 

If the problem can be fixed online, the disabled CRR log minidisk is automatically enabled after the next CRR log checkpoint.
3. If both CRR log minidisks (CRR1 and CRR2) are disabled because of an I/O error, at the next CRR log checkpoint, the CRR recovery server terminates. For more information about replacing both CRR log minidisks, see [“Replace Both CRR Log Minidisks”](#) on page 328.

### Responses

The following is an example of the information that could be displayed for the CRR log minidisks:

```
Time hh:mm:ss          CRR QUERY LOG - filepoolID
Date mm/dd/yy         LUNAME - nnnnnnnn.llllllll

Log Status

Total Number of Log Minidisk 4K Blocks:  nnnnnnnnnn
Number of 4K Blocks for Log Ring:        nnnnnnnnnn
Number of 4K Blocks for Logname Table:   nnnnnnnnnn

Percent(%) of Log Ring Space Used:       nnn %
Percent(%) of Logname Table Space Used:   nnn %

Primary Log is <Disabled|Enabled>
Secondary Log is <Disabled|Enabled>
```

### Time and Date

are the local time and date the CRR QUERY LOG command was issued.

### CRR QUERY LOG

identifies the command issued and the file pool ID associated with this CRR recovery server.

### LUNAME

is the fully qualified LU name associated with this CRR recovery server where:

- *nnnnnnnn* is the optional SNA network ID.
- *llllllll* is the LU name.

### Log Status

contains the following fields:

#### **Total Number of Log Minidisk 4K Blocks**

is the number of 4KB blocks contained in each of the two CRR log minidisks.

#### **Number of 4K Blocks for Log Ring**

is the number of 4KB blocks to be used for recording CRR log records in each of the two CRR log minidisks. The log ring contains about 67% of the 4KB blocks in the CRR log minidisks.

#### **Number of 4K Blocks for Logname Table**

is the number of 4KB blocks to be used for storing the CRR log name table in each of the two CRR log minidisks. The CRR log name table contains about 33% of the 4KB blocks in the CRR log minidisks.

#### **Percent(%) of Log Ring Space Used**

is the percentage of the log ring space used for recording CRR log records.

#### **Percent(%) of Logname Table Space Used**

is the percentage of the CRR log name table space used for storing the CRR log name table. If *nnn* becomes high (around 95%), you may want to increase the size of the CRR log minidisks or free space in the CRR log name table. For more information, see [“CRR Log Name Table Management” on page 321](#).

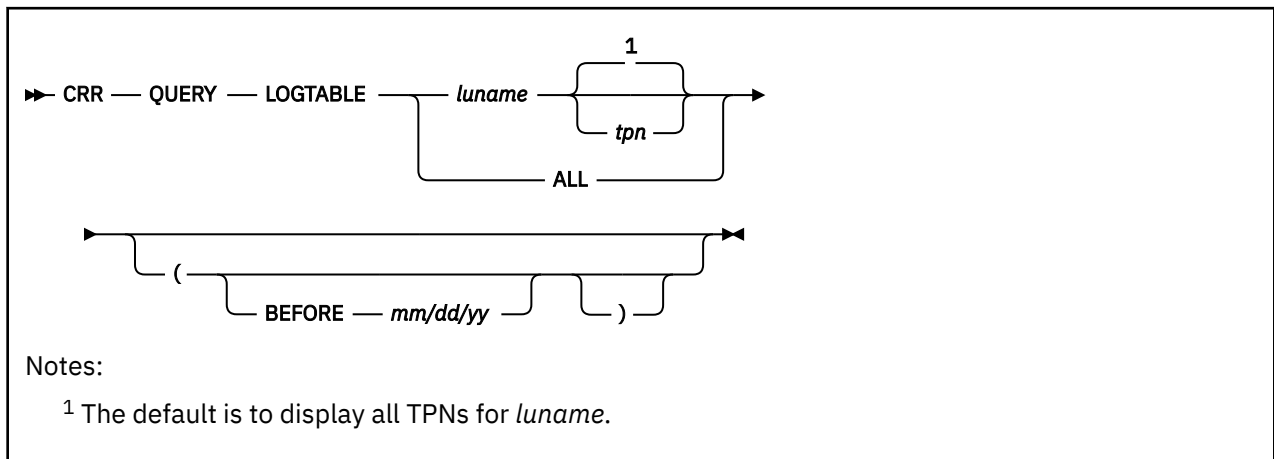
#### **Primary Log is <Disabled|Enabled>**

The primary CRR log minidisk (CRR1) is enabled unless an I/O error occurred since the last CRR log checkpoint.

#### **Secondary Log is <Disabled|Enabled>**

The secondary CRR log minidisk (CRR2) is enabled unless an I/O error occurred since the last CRR log checkpoint.

## CRR QUERY LOGTABLE



### Authorization

CRR Operator

### Purpose

Use the CRR QUERY LOGTABLE operator command to display the specified LU name and transaction program name (TPN) for entries in the CRR log name table.

**Note:** The TPN is sometimes called resource ID.

### Operands

*luname*

*luname tpn*

displays information about all CRR log name table entries with an LU name specified by *luname* and optionally displays TPNs specified by *tpn*. If *tpn* is not specified, the CRR QUERY LOGTABLE command displays all TPNs for the specified *luname*. You can specify the *luname* as \*LOCAL to display TPNs for all LU names on this system.

**ALL**

displays information about all CRR log name table entries.

### Options

**BEFORE mm/dd/yy**

limits the display to CRR log name table entries that were referenced for the last time before the specified date. The month (*mm*), day (*dd*), and year (*yy*) must be two digits each.

### Usage Notes

1. The CRR QUERY LOGTABLE command provides information required by the CRR ERASE LU command for freeing space in the CRR log name table. For more information about freeing space in the CRR log name table, see [“Free Space in the CRR Log Name Table”](#) on page 323.
2. If *tpn* is not specified, the command will display information about all log name table entries with the specified LU name. For example, if the CRR recovery server operator issues:

```
crr query logtable gdlvm8.recover
```

## CRR QUERY LOGTABLE

the display returns all CRR log name table entries that have the fully qualified LU name of GDLVM8.RECOVER.

3. If the date is not specified correctly, for example without leading zeros or values that are not valid, it will be ignored and information about all requested entries will be displayed regardless of their date of last reference.
4. The date of last reference, which is displayed with the CRR log name table entry, is calculated based on Coordinated Universal Time (UTC).
5. The only characters contained in the TPN considered printable characters are: A-Z, a-z, 0-9, @, #, \$, period (.). All other characters contained in the TPN are considered unprintable characters.
6. When the TPN in the log name table entry contains unprintable characters, each unprintable character will be replaced by a period and the hexadecimal translation of the TPN will follow on the next line in the display. A log name token will also appear on the line following the translated TPN. This log name token is used in the CRR ERASE LU command instead of the TPN whenever the TPN is required.
7. The log name tokens are resequenced periodically.

## Responses

The following is an example of the information that could be displayed for multiple log name table entries found:

```
Time: hh:mm:ss          CRR QUERY LOGTABLE - filepoolid
Date: mm/dd/yy         LUNAME - nnnnnnnn.llllllll

Remote   Transaction   Local   Last
LU Name  Program Name  LU Name Referenced

luname  tpn          luname  mm/dd/yy
.        .            .        .
.        .            .        .
luname  tpn          luname  mm/dd/yy
```

### Time and Date

are the local time (*hh:mm:ss*) and date (*mm/dd/yy*) the CRR QUERY LOGTABLE command was issued.

### CRR QUERY LOGTABLE

identifies the command issued and the file pool ID associated with this CRR recovery server.

### LUNAME

is the fully qualified LU name associated with this CRR recovery server where:

- *nnnnnnnn* is the optional SNA network ID
- *llllllll* is the LU name

### Remote LU name

is a heading followed by a list of LU name entries (*luname*) from the CRR log name table. For TPNs on this system or TSAF or CS collection, the LU name is displayed as \*LOCAL.

### Transaction Program Name

is a heading followed by a list of transaction program names (*tpn*) associated with the remote *luname*. If the *tpn* contains unprintable characters, it will be displayed after replacing each of the unprintable characters with a period. The hexadecimal translation of the *tpn* will be shown on the following line in the display. A log name token will also appear on the line following the translated TPN. This log name token would be used in the CRR ERASE LU command instead of the TPN whenever the TPN is required.

### Local LU name

is a heading followed by a list of LU name entries (*luname*) which are sent to the associated participating resource. For TPNs on this system or TSAF or CS collection, the LU name is displayed as \*LOCAL.

**Last Referenced**

is a heading followed by a list of dates that indicate when this CRR log name table entry was last referenced for:

- Initial exchange of log names
- Resynchronization activity for this LU name

**Examples**

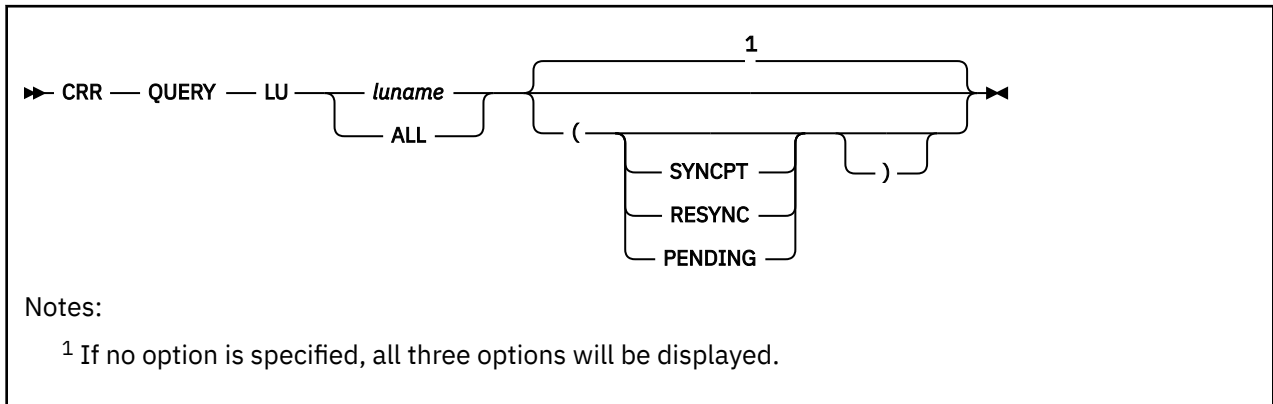
The following log name table output represents the type of data that will be displayed when an entry contains a TPN which consists of 1 or more unprintable characters.

Remote LU Name	Transaction Program Name	Local LU Name	Last Referenced
GDLVME.RECOVER *LOCAL	RECOVERYTPN 2..A 'F20601C1'X 000002	GDLVM1.SERV1 *LOCAL	dd/mm/yy dd/mm/yy
*LOCAL	MYTPN	*LOCAL	dd/mm/yy

The second entry in the above display consists of a TPN that contains 2 unprintable characters. The unprintable characters have been replaced by a period and the hexadecimal representation of the TPN follows on the next line. The unique log name token assigned to this entry is 2.

**Note:** The translated TPN and log name token only appear in the entry that has a TPN which consists of 1 or more unprintable characters.

## CRR QUERY LU



### Authorization

CRR Operator

### Purpose

Use the CRR QUERY LU operator command to display the status of the logical units of work ID (LUWIDs) known to this CRR recovery server.

The status does not include information about the protected resources and protected conversations involved in each LUWID instance. If that information is needed, the token returned from the CRR QUERY LU display should be used as input to the CRR QUERY LUWID command.

### Operands

#### *luname*

displays all active participants at the specified LU name known to this CRR recovery server.

You must specify the LU name as \*LOCAL for all LU names on this system or TSAF or CS collection.

#### **ALL**

displays all active participants at all LU names known to this CRR recovery server.

### Options

#### **SYNCPT**

displays only those LUWID instances for the specified LU name in synchronization (sync) point processing.

#### **RESYNC**

displays only those LUWID instances for the specified LU name in resynchronization processing.

#### **PENDING**

displays only those LUWID instances for the specified LU name that have:

- Not yet started resynchronization processing
- Completed downstream resynchronization and waiting for its initiator or cascaded initiator to send the resynchronization transaction (reserved TPN X'06F2').

### Usage Notes

1. If you enter:



```
crr query lu gdlvme.recover
```

the display returns information about active sync point participants at the processor with the SNA network ID of GDLVME and the LU name (or gateway name) of RECOVER.

2. If you enter:

```
crr query lu gdlvme.recover (resync
```

the display returns information about active sync point participants in resynchronization processing at the processor with the SNA network ID of GDLVME and the LU name (or gateway name) of RECOVER.

3. If you enter:

```
crr query lu all
```

the display returns information about active sync point participants at all LU names know by this CRR recovery server.

4. By specifying one of the three options (SYNCPT, RESYNC, or PENDING) the display can be limited to LUWIDs in sync point, resynchronization, or resynchronization pending process. Only one option can be specified at a time. If none of these options is specified, all three will be displayed.

## Responses

The following is an example of the information that could be displayed for each LUWID instance found:

```
Time: hh:mm:ss          CRR QUERY LU - filepoolid
Date: mm/dd/yy        LUNAME - nnnnnnnn.llllllll

LUWID                  Token
nnnnnnnn.llllllll.iiiiiiiiiii.ssss  tttttttt
Name                   Process
userid                 current_process
Syncpoint Role        Syncpoint State  Status
current_syncpt_role  current_syncpt_state  current_status
Transaction Tag
transaction_tag_field

Initiator Name
luname tpn
Recovery TPN
recovery_tpn
Recovery Token        Index
rrrrrrrr             iii
Resync Role           Resync State  Access Userid
current_resync_role  current_resync_state  access_userid

LUWID                  Token
nnnnnnnn.llllllll.iiiiiiiiiii.ssss  tttttttt
Name                   Process
userid                 current_process
Syncpoint Role        Syncpoint State  Status
current_syncpt_role  current_syncpt_state  current_status
Transaction Tag
transaction_tag_field
```

### Time and Date

are the local time (*hh:mm:ss*) and date (*mm/dd/yy*) the CRR QUERY LU command was entered.

### CRR QUERY LU

identifies the command entered and the file pool ID associated with this CRR recovery server.

### LUNAME

is the fully qualified LU name associated with this CRR recovery server where:

- *nnnnnnnn* is the optional SNA network ID
- *llllllllll* is the LU name

### LUWID

is the logical unit of work ID associated with the entire coordinated transaction where:

- *nnnnnnnn.llllllll* is the fully qualified LU name where the LUWID was created that consists of:
  - *nnnnnnnn* is the optional SNA network ID
  - *llllllll* is the LU name
- *iiiiiiiiiiii* is the instance number
- *ssss* is the sequence number

### Token

is the value (*ttttttt*) returned in the display the operator can use as input to enter subsequent commands, relating to the LUWID instance, such as CRR QUERY LUWID, CRR SUSPEND, CRR RESUME, CRR RESYNC, and CRR ERASE LUWID.

**Note:** The value in *token* identifies an LUWID instance. An LUWID instance is a subset of a coordinated transaction (also called a CRR logical unit of work). The coordinated transaction is identified by the LUWID. The LUWID instance can consist of one or more resource logical units of work.

The token returned in this display is valid as long as this instance of the LUWID exists. As soon as the work represented by this instance is finished, it is erased from the CRR log and the token is then no longer valid.

### Name

is the user ID (*userid*) for the application's virtual machine on this processor (the one the output is from).

### Process

is the process this LUWID instance is in, where *current\_process* could be one of the following:

- RESYNCHRONIZATION–Recovery process (in resynchronization, including timed wait)
- RESYNCHRONIZATION PENDING–Waiting for recovery (initiator sending resynchronization TPN X'06F2') or resynchronization has not started yet
- SYNCPOINT–Sync point processing (logging for two-phase commit)

### Syncpoint Role

is the role associated with this node in the sync point tree, where *current\_syncpt\_role* could be one of the following:

- AGENT–Local resource node in sync point tree. The role of agent and initiator cascade can be found at any node of the sync point tree, except at the node of the true initiator of the sync point. The same node can have the role of agent or initiator cascade depending on how far this LUWID instance has progressed into sync point processing when you entered the CRR QUERY LU command.

A node has the role of agent, if the sync point activity is flowing from this node, **up** the sync point tree to an initiator or initiator cascade.

- Blanks–Resynchronization processing issued a forget state for this node in the sync point tree.
- INITIATOR–Initiator node in the sync point.
- INITIATOR CASCADE–Intermediate node in the sync point tree. The role of initiator cascade and agent can be found at any node of the sync point tree, except at the node of the true initiator of the sync point. The same node can have the role of initiator cascade or agent depending on how far this LUWID instance has progressed into sync point processing when you entered the CRR QUERY LU command.

A node has the role of initiator cascade, if the sync point activity is flowing from this node, **down** the sync point tree to an agent.

### Syncpoint State

is the state associated with this node in the sync point tree, where *current\_syncpt\_state* could be one of the following:

- COMMITTED–If this state is found at an initiator (see INITIATOR under Syncpoint Role) or cascaded initiator (see INITIATOR CASCADE under Syncpoint Role), it indicates the **intent** of the second phase of the two-phase commit, that is a commit is being propagated down the sync point tree. If this state is found at an agent, it indicates the **result** of the second phase of the two-phase commit

for this branch of the sync point tree. The intent to commit is being propagated to other branches of the sync point tree.

- COMMITTED (RIP)—This node was designated as last agent by the initiator. Acting as the last agent, all changes for this branch of the sync point tree were committed. However, resynchronization is in progress to complete the commit.
- DELAYED FORGET—CRR log record may be deleted in the future, when the downstream resynchronization work has completed and the delayed forget request is received from AVS. No response to the initiator or sync point manager (SPM) is required.
- FORGET (RIP)—Sync point has completed, however resynchronization is in progress for a portion of the sync point tree.
- HEURISTIC MIXED—This is a heuristic damage situation where either changes were committed when they were supposed to be rolled back, or changes were rolled back when they were supposed to be committed.
- IN-DOUBT—All the downstream agents have prepared their changes and are ready to either commit or roll back the changes, depending on the directions received from the initiator. The first phase of the two-phase commit has completed and is waiting to start the second phase of the two-phase commit. CRR only logs the in-doubt state for agents (see AGENT under Syncpoint Role).

When an initiator cascade node receives a rollback from an initiator, the in-doubt state remains recorded in the initiator cascade node's log until all agents have responded to the initiator cascade node, and the initiator cascade node has sent a response to the initiator; the log record is erased.

- SYNCPOINT PENDING—This is the initially logged sync point state. It indicates a sync point is starting the first phase of a two-phase commit. For a cascaded initiator, the sync point pending state has a different meaning. The CRR operator may need to examine the state of the initiator.

### Status

is the status of the CRR recovery server task associated with this LUWID instance, where *current\_status* could be one of the following:

- ACTIVE – indicates a CRR recovery server task is associated with the LUWID instance and is currently active.
- AVS WAIT – indicates a CRR recovery server task is associated with the LUWID instance and it is currently waiting for a response from AVS, for example, to deactivate a session.
- Blanks – indicate the coordinated transaction is not yet associated with a CRR recovery server task or the transaction is still in sync point processing.
- COMMUNICATION WAIT – indicates a 'link' exists between a CRR recovery server task and either another CRR recovery server or a participating resource manager and is waiting for communication from it.
- GENERAL WAIT – indicates a CRR recovery server task is associated with the LUWID instance and it is currently in some other type of wait, not covered by lock wait, timed wait, I/O wait, and communication wait.
- I/O WAIT – indicates the CRR recovery server task is waiting for DASD I/O to complete.
- LOCK WAIT – indicates a CRR recovery server task is associated with the LUWID instance and it is currently waiting for a lock or a latch to be freed.
- SUSPENDED – indicates a CRR recovery server task is associated with the LUWID instance, but has been suspended by the CRR SUSPEND command.
- TIMED WAIT – indicates a CRR recovery server task is associated with the LUWID instance and it is currently waiting for the timed wait interval to elapse before trying its next attempt at resynchronization. Resynchronization attempts to connect to a protected resource or another CRR recovery server.

### Transaction Tag

is a field supplied by the application program using the Set Transaction Tag CSL routine (DMSSETAG) or the Get Work Unit ID CSL routine (DMSGETWU). It is 1-80 bytes long and if nothing was supplied by

the application it is blanks. For additional information on the DMSSETAG or DMSGETWU CSL routines, see *z/VM: CMS Callable Services Reference*.

### Initiator Name

is the fully qualified LU name (*luname*) and Transaction Program Name (*tpn*) associated with the LUWID instance that initiated the sync point. If this heading is missing, this node is the initiator of the sync point. The initiator may not be the true initiator of the sync point; it could be a cascaded initiator. If the *tpn* contains unprintable characters, it will be displayed after replacing each of the unprintable characters with a period. The hexadecimal translation of the *tpn* will be shown on the following line in the display.

**Note: The following fields describe the attributes of the initiator of the sync point:**

### Recovery TPN

*recovery\_tpn* is the TPN associated with resynchronization processing. If the *recovery\_tpn* contains unprintable characters, it will be displayed after replacing each of the unprintable characters with a period. The hexadecimal translation of the *recovery\_tpn* will be shown on the following line in the display.

### Recovery Token

*rrrrrrr* is a conversation correlator. It associates this entry with the corresponding entry for the initiator.

### Index

*iii* is the index representing the initiator in this LUWID instance and is used as input to the CRR RESYNC command. The value in index identifies a resource that is doing a subset of work represented by an LUWID instance.

### Resync Role

is the role, where *current\_resync\_role* could be one of the following:

- FORGET – if the initiator has completed resynchronization and need never be contacted again.
- NOT-IN-RESYNC – If not in resynchronization processing.
- RESYNCHRONIZATION – if the CRR RESYNC command has been entered on behalf of our initiator.
- RESYNC NEEDED – If resynchronization processing has started.

### Resync State

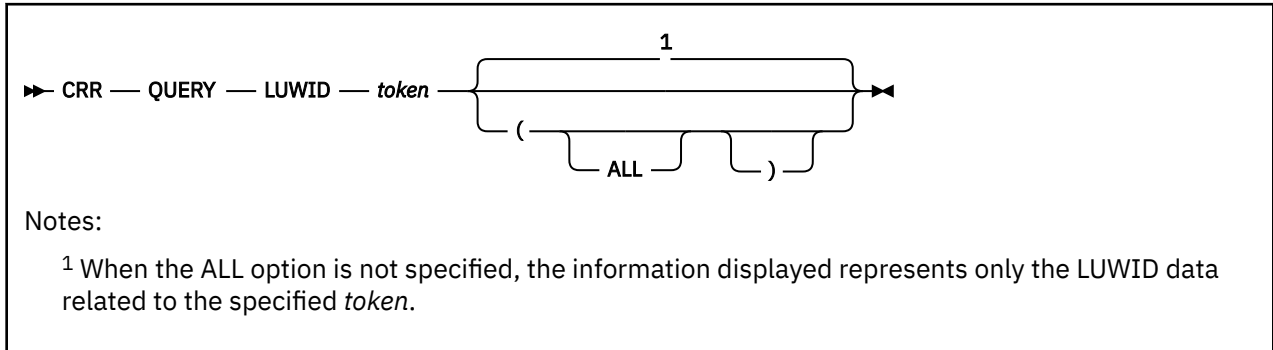
is the state associated with our initiator as reported to the CRR recovery server during resynchronization processing, where *current\_resync\_state* could be one of the following:

- BACKOUT – All changes were backed out. The backout state is equivalent to the LU 6.2 architected state of reset.
- COMMITTED – All the changes were committed. If this state is found at an initiator or cascaded initiator, it indicates the **intent** of the second phase of the two-phase commit, that is, a commit is being propagated down the sync point tree. If this state is found at an agent, it indicates the **result** of the second phase of the two-phase commit for this branch of the sync point tree. The intent to commit is being propagated to other branches of the sync point tree.
- HEURISTIC BACKOUT – The operator independently rolled back the changes. At this point, it is not known if the operator's decision to rollback will cause heuristic damage (the heuristic mixed state). The heuristic backout state is equivalent to the LU 6.2 architected state of heuristic reset.
- HEURISTIC COMMITTED – The operator independently committed the changes. At this point, it is not known if the operator's decision to commit will cause heuristic damage (the heuristic mixed state).
- HEURISTIC MIXED – This is a heuristic damage situation where either changes were committed when they were supposed to be rolled back, or changes were rolled back when they were supposed to be committed.
- NOT-IN-RESYNC – If in sync point process rather than resynchronization or if resynchronization has not gotten a response yet.
- RESYNC NEEDED – If resynchronization processing has started.

**Access Userid**

is the user ID (*access\_userid*) of one of the partners in the protected conversation.

## CRR QUERY LUWID



### Authorization

CRR Operator

### Purpose

Use the CRR QUERY LUWID operator command to display the status, at a given instance of a synchronization (sync) point, of the protected resources and protected conversations involved in a logical unit of work ID (LUWID) known to this CRR recovery server.

The CRR QUERY LUWID command displays some of the same information about an active sync point participant you get from the CRR QUERY LU command. But, the CRR QUERY LUWID command also gives information about the protected resources and protected conversations involved in the active sync point at this node.

An instance corresponds to a unique conversation correlator or recovery token associated with the LUWID.

### Operands

#### *token*

identifies the LUWID instance whose protected resources and protected conversations you want to display. The *token* could be obtained from one of the following:

- Resynchronization messages displayed at the operator's console for this CRR recovery server
- Output of the CRR QUERY LU command

**Note:** The value in *token* identifies an LUWID instance. An LUWID instance is a subset of a coordinated transaction (also called a CRR logical unit of work). The coordinated transaction is identified by the LUWID. The LUWID instance can consist of one or more resource logical units of work.

### Options

#### ALL

causes all instances of LUWID known to this CRR recovery server identified by the specified token to be included in the display.

### Usage Notes

1. This command displays information about the specific LUWID instance related to the *token* that was given in a previously issued CRR QUERY LU command or in a message from resynchronization processing.

2. The *token* is valid as long as this instance of the LUWID exists. As soon as the work represented by this instance is finished, it is erased from the CRR log and the *token* then is no longer valid.

## Responses

The following is an example of the information that could be displayed for the instance of the LUWID found which matches the token specified:

```

Time: hh:mm:ss          CRR QUERY LUWID - filepoolid
Date: mm/dd/yy         LUNAME - nnnnnnnn.llllllll

LUWID                  Token
nnnnnnnn.llllllll.iiiiiiiiiii.ssss  tttttttt
Name                   Process
userid                 current_process
Syncpoint Role        Syncpoint State  Status
current_syncpt_role  current_syncpt_state  current_status
Transaction Tag
transaction_tag_field

Initiator Name
luname tpn
Recovery TPN
recovery_tpn
Recovery Token
rrrrrrrr
Resync Role           Resync State  Index
current_resync_role current_resync_state iii
Access Userid
access_userid

Resources
luname tpn Recovery TPN
recovery_tpn
Recovery Token
rrrrrrrr
Resync Role           Resync State  Index
current_resync_role current_resync_state iii
Access Userid
access_userid

luname tpn
Recovery TPN
recovery_tpn
Recovery Token
rrrrrrrr
Resync Role           Resync State  Index
current_resync_role current_resync_state iii
Access Userid
access_userid

```

### Time and Date

are the local time (*hh:mm:ss*) and date (*mm/dd/yy*) the CRR QUERY LUWID command was issued. If the QUERY LUWID command was specified with the ALL option, the time and date will only be displayed one time at the top of the display.

### CRR QUERY LUWID

identifies the command issued and the file pool ID associated with this CRR recovery server.

### LUNAME

is the fully qualified LU name associated with this CRR recovery server where:

- *nnnnnnnn* is the optional SNA network ID
- *llllllllll* is the LU name

### LUWID

is the logical unit of work ID associated with the entire coordinated transaction where:

- *nnnnnnnn.llllllllll* is the fully qualified LU name where the LUWID was created that consists of:
  - *nnnnnnnn* is the optional SNA network ID
  - *llllllllll* is the LU name
- *iiiiiiiiiii* is the instance number
- *ssss* is the sequence number

### Token

is the value (*ttttttt*) the operator supplied as input to the CRR QUERY LUWID command. The token could be obtained from one of the following:

- Resynchronization messages displayed at the operator's console for this CRR recovery server
- Output of the CRR QUERY LU command

### Name

is the user ID (*userid*) of the application's virtual machine on this processor (the one the output is from).

### Process

is the process this LUWID instance is in, where *current\_process* could be one of the following:

- RESYNCHRONIZATION – Recovery process (in resynchronization, including timed wait).
- RESYNCHRONIZATION PENDING – Waiting for recovery (initiator sending resynchronization TPN X'06F2') or resynchronization processing has not started yet.
- SYNCPOINT – Usual two-phase commit processing.

### Syncpoint Role

is the role associated with this node in the sync point tree, where *current\_syncpt\_role* could be one of the following:

- AGENT – Local resource node in the sync point tree. The role of agent and initiator cascade can be found at any node of the sync point tree, except at the node of the true initiator of the sync point. The same node can have the role of agent or initiator cascade depending on how far this LUWID instance has progressed into sync point processing when you issued the CRR QUERY LUWID command.

A node has the role of agent, if the sync point activity is flowing from this node, **up** the sync point tree to an initiator or initiator cascade.

- Blanks – Resynchronization processing issued a forget state for this node in the sync point tree.
- INITIATOR – Initiator node in the sync point.
- INITIATOR CASCADE – Intermediate node in the sync point tree. The role of initiator cascade and agent can be found at any node of the sync point tree, except at the node of the true initiator of the sync point. The same node can have the role of initiator cascade or agent depending on how far this LUWID instance has progressed into sync point processing when you issued the CRR QUERY LUWID command.

A node has the role of initiator cascade, if the sync point activity is flowing from this node, **down** the sync point tree to an agent.

### Syncpoint State

is the state associated with this node in the sync point tree, where *current\_syncpt\_state* could be one of the following:

- COMMITTED – If this state is found at an initiator (see INITIATOR under Syncpoint Role) or cascaded initiator (see INITIATOR CASCADE under Syncpoint Role), it indicates the **intent** of the second phase of the two-phase commit, that is, a commit is being propagated down the sync point tree. If this state is found at an agent, it indicates the **result** of the second phase of the two-phase commit for this branch of the sync point tree. The intent to commit is being propagated to other branches of the sync point tree.
- COMMITTED (RIP) – This node was designated as last agent by the initiator. Acting as the last agent, all changes were committed. However, resynchronization is in progress to complete the commit.
- DELAYED FORGET – CRR log record may be deleted in the future, when the downstream resynchronization work has completed and the delayed forget request is received from AVS. No response to the initiator or sync point manager (SPM) is required.
- FORGET (RIP) – Sync point has completed, however resynchronization is in progress for a portion of the sync point tree.



- HEURISTIC MIXED – This is a heuristic damage situation where either changes were committed when they were supposed to be rolled back, or changes were rolled back when they were supposed to be committed.
- IN-DOUBT – All the downstream agents have prepared their changes and are ready to either commit or roll back the changes, depending on the directions received from the initiator. The first phase of the two-phase commit has completed and is waiting to start the second phase of the two-phase commit. CRR only logs the in-doubt state for agents (see AGENT under Syncpoint Role).

When an initiator cascade node receives a rollback from an initiator, the in-doubt state remains recorded in the initiator cascade node's log until all agents have responded to the initiator cascade node, and the initiator cascade node has sent a response to the initiator; then the log record is erased.

- SYNCPOINT PENDING – This is the initially logged sync point state. It indicates a sync point is starting the first phase of a two-phase commit. For a cascaded initiator, the sync point pending state has a different meaning. The CRR operator may need to examine the state of the initiator.

### Status

is the status of the CRR recovery server task associated with this LUWID instance, where *current\_status* could be one of the following:

- ACTIVE – indicates a CRR recovery server task is associated with the LUWID instance and is currently active.
- AVS WAIT – indicates a CRR recovery server task is associated with the LUWID instance and it is currently waiting for a response from AVS, for example, to deactivate a session.
- Blanks – indicate the LUWID instance is not yet associated with a CRR recovery server task or the LUWID instance still in sync point processing.
- COMMUNICATION WAIT – indicates a 'link' exists between a CRR recovery server task and either another CRR recovery server or a participating resource manager and is waiting for communication from it.
- GENERAL WAIT – indicates a CRR recovery server task is associated with the LUWID instance and it is currently in some other type of wait, not covered by lock wait, timed wait, I/O wait, and communication wait.
- I/O WAIT – indicates the CRR recovery server task is waiting for DASD I/O to complete.
- LOCK WAIT – indicates a CRR recovery server task is associated with the LUWID instance and it is currently waiting for a lock or a latch to be freed.
- SUSPENDED – indicates a CRR recovery server task is associated with the LUWID instance, but has been suspended by the CRR SUSPEND command.
- TIMED WAIT – indicates a CRR recovery server task is associated with the LUWID instance and it is currently waiting for the timed wait interval to elapse before trying its next attempt at resynchronization. Resynchronization attempts to connect to a protected resource or another CRR recovery server.

### Transaction Tag

is a field supplied by the application program using the Set Transaction Tag CSL routine (DMSSETAG) or the Get Work Unit ID CSL routine (DMSGETWU). It is 1-80 bytes long and if nothing was supplied by the application it is blanks. For additional information on the DMSSETAG or DMSGETWU CSL routines, see [z/VM: CMS Callable Services Reference](#).

### Initiator Name

is the fully qualified LU name (*luname*) and Transaction Program Name (*tpn*) associated with the LUWID instance that initiated the sync point. If this heading is missing, this node is the initiator of the sync point. The initiator may not be the true initiator of the sync point, it could be a cascaded initiator. If the *tpn* contains unprintable characters, it will be displayed after replacing each of the unprintable characters with a period. The hexadecimal translation of the *tpn* will be shown on the following line in the display.

The following fields describe the attributes of the initiator of the sync point:

**Recovery TPN**

*recovery\_tpn* is the TPN associated with resynchronization processing. If the *recovery\_tpn* contains unprintable characters, it will be displayed after replacing each of the unprintable characters with a period. The hexadecimal translation of the *recovery\_tpn* will be shown on the following line in the display.

**Recovery Token**

*rrrrrrr* is a conversation correlator if the resource is a protected conversation or a resource defined token if it was a protected resource. It is used to associate this entry with the corresponding resource task.

**Index**

*iiii* is the index representing the initiator in this LUWID instance and is used as input for the CRR RESYNC command. The value in index identifies a resource that is doing a subset of work represented by an LUWID instance.

**Resync Role**

is the role, where *current\_resync\_role* could be one of the following:

- FORGET – if the initiator has completed resynchronization and need never be contacted again.
- NOT-IN-RESYNC – if not in resynchronization processing.
- RESYNCHRONIZATION – if the CRR RESYNC command has been issued on behalf of our initiator.
- RESYNC NEEDED – if resynchronization processing has started.

**Resync State**

is the state associated with our initiator as reported to the CRR recovery server during resynchronization processing, where *current\_resync\_state* could be one of the following:

- BACKOUT – All changes were backed out. The backout state is equivalent to the LU 6.2 architected state of reset.
- COMMITTED – All the changes were committed. If this state is found at an initiator or cascaded initiator, it indicates the **intent** of the second phase of the two-phase commit, that is, a commit is being propagated down the sync point tree. If this state is found at an agent, it indicates the **result** of the second phase of the two-phase commit for this branch of the sync point tree. The intent to commit is being propagated to other branches of the sync point tree.
- HEURISTIC BACKOUT – The operator independently rolled back the changes. At this point, it is not known if the operator's decision to rollback will cause heuristic damage (the heuristic mixed state). The heuristic backout state is equivalent to the LU 6.2 architected state of heuristic reset.
- HEURISTIC COMMITTED – The operator independently committed the changes. At this point, it is not known if the operator's decision to commit will cause heuristic damage (the heuristic mixed state).
- HEURISTIC MIXED – This is a heuristic damage situation where either changes were committed when they were supposed to be rolled back, or changes were rolled back when they were supposed to be committed.
- NOT-IN-RESYNC – If in sync point process rather than resynchronization or if resynchronization has not gotten a response yet.
- RESYNC NEEDED – If resynchronization processing has started.

**Access Userid**

is the user ID (*access\_userid*) that was used to access the associated resource. If the resource adapter associated with the resource did not register this user ID, the display will contain blanks, otherwise this field will contain the user ID that was registered. It should match any user IDs in corresponding displays at the resource.

This is also the alternate user ID associated with the LUWID instance.

Because the APPC/VM connection to the resources may have been done with an alternate user ID active and any CRR QUERYS done at that resource would give the alternate user ID, the alternate user ID will be the one logged in the CRR log records to make them compatible.

*access\_userid* could also be the user ID one of the partners in a protected conversation.

### Resources

contains a list of protected resources and protected conversations associated with this LUWID instance at this node. Each protected resource and protected conversation contains the following information:

- Fully qualified LU name (*luname*) and transaction program name (*tpr*) for a protected resource and a fully qualified LU name for a protected conversation. A protected resource or protected conversation local to the processor, TSAF or CS collection, has an LU name of \*LOCAL.
- Recovery TPN—Same meaning as previously defined for the initiator of the sync point, but here it applies to the protected resources and protected conversations.
- Recovery token—Same meaning as previously defined for the initiator of the sync point, but here it applies to the protected resources and protected conversations.
- Index—Same meaning as previously defined for the initiator of the sync point, but here it applies to the protected resources and protected conversations.
- Resync role—Same meaning as previously defined for the initiator of the sync point, but here it applies to the protected resources and protected conversations. In addition, resources (not initiators) could also have a resynchronization role of:
  - AGENT – if the agent has responded with a state, but its log record has not been erased (forgotten) yet.
- Resync state—Same meaning as previously defined for the initiator of the sync point, but here it applies to the protected resources and protected conversations.
- Access user ID—Same meaning as previously defined for the initiator of the sync point, but here it applies to the protected resources and protected conversations.

## CRR RESUME

►► CRR — RESUME — *token* ◄◄

### Authorization

CRR Operator

### Purpose

Use the CRR RESUME operator command to restart the automatic periodic retry of resynchronization for the logical unit of work ID (LUWID) instance specified by *token*. This cancels the suspended status of the LUWID instance, which was initiated by the CRR SUSPEND command. If the attempt at resynchronization does not work, the LUWID instance goes into timed wait.

Also use the CRR RESUME operator command to bypass the timed wait interval.

### Operands

#### *token*

identifies the LUWID instance to be put into the automatic periodic retry of resynchronization. The *token* could be obtained from one of the following:

- Resynchronization messages displayed at the operator's console of this CRR recovery server
- Output of the CRR QUERY LU command
- Output of the CRR QUERY LUWID command

**Note:** The value in *token* identifies an LUWID instance. An LUWID instance is a subset of a coordinated transaction (also called a CRR logical unit of work). The coordinated transaction is identified by the LUWID. The LUWID instance can consist of one or more resource logical units of work.

### Usage Notes

1. The automatic periodic retry of resynchronization consists of:
  - a. The CRR recovery server attempts resynchronization of the LUWID instance.
 

If the protected resources or protected conversations are available and resynchronization completes, this LUWID instance is no longer in automatic periodic retry of resynchronization.

If the protected resources or protected conversations are not available and therefore resynchronization does not complete, this LUWID instance goes into a timed wait.
  - b. The LUWID instance remains in timed wait for a time period (called timed wait interval) that is established by the RESYNCINTERVAL startup parameter. At the end of the timed wait interval, there is another attempt at resynchronization. Go to step “1.a” on page 388.
2. The CRR RESUME command can only be issued while the LUWID instance is in timed wait or suspended status. You can determine if the LUWID instance is in a timed wait or suspended status by examining the *current\_status* value in the display from either the CRR QUERY LU or CRR QUERY LUWID.
3. The operator uses the CRR QUERY LU command to determine if the specified LUWID instance now has the protected resources or protected conversations available to continue resynchronization. If the protected resources or protected conversations are available, using the CRR RESUME command makes the LUWID instance available for the automatic periodic retry of resynchronization in this CRR recovery server. If possible, resynchronization immediately continues. If resynchronization does not find all protected resources or protected conversations available to process the LUWID instance, the timed wait continues for the LUWID instance and you may want to use the CRR SUSPEND command.

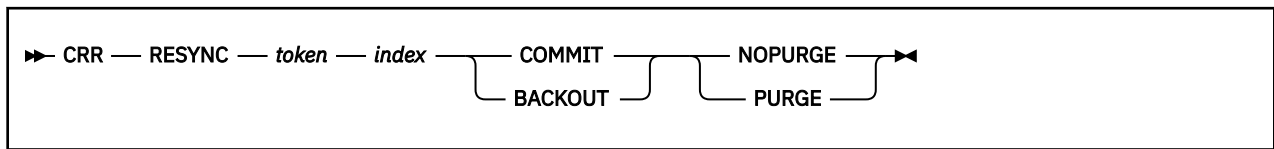
Using the CRR SUSPEND command temporarily withdraws an LUWID instance from timed wait and puts the LUWID instance into a suspended status, which makes the LUWID instance unavailable for the automatic periodic retry of resynchronization.

The timed wait interval, which is determined by the RESYNCINTERVAL startup parameter, is the amount of time the LUWID instance remains in timed wait before leaving timed wait to make the next attempt at resynchronization. The default timed wait interval is 10 minutes.

If you want to change the timed wait interval between attempts to resynchronize, you must use the RESYNCINTERVAL startup parameter. For more information on the RESYNCINTERVAL startup parameter, see Chapter 20, [“File Pool Server Startup Parameters,”](#) on page 339. For more instructions on changing the timed wait interval, see [“Bypass and Change Timed Wait Interval”](#) on page 325.

4. The CRR RESUME command restores an LUWID instance to the resynchronization process after it has been stopped by the CRR SUSPEND command.
5. The CRR RESUME command can also be used to bypass the timed wait interval before the next attempt at resynchronization. If the CRR RESUME command is issued for an LUWID instance that has not been previously put into a suspended status by the CRR SUSPEND command, the only function of the CRR RESUME command is to cause an immediate resynchronization attempt, regardless of the amount of time left in the timed wait interval before another attempt at resynchronization. If the CRR RESUME command is unsuccessful, the LUWID instance returns to the original timed wait interval. If the CRR RESUME command is successful, but the attempt at resynchronization is not, the LUWID instance issues a new timed wait interval with the clock reset to the value of the RESYNCINTERVAL startup parameter.

## CRR RESYNC



### Authorization

CRR Operator

### Purpose

Use the CRR RESYNC operator command to provide a *heuristic response* for a protected resource or protected conversation, when the automatic periodic retry of resynchronization has failed to complete resynchronization, so resynchronization can continue for an instance of a logical unit of work ID (LUWID).

A heuristic decision requires the operator to enter the CRR QUERY LU and CRR QUERY LUWID commands and then to examine the output of these commands to determine the status of the LUWID instance at all nodes involved with this LUWID. After the operator examines the output and discusses the situation with a participating resource operator and/or a CRR recovery server operator on another system, they can use the CRR RESYNC command (or participating resource commands, such as SFS's FORCE PREPARED) to either commit or back out the unavailable protected resource. Then the operators may want to erase log records of this LUWID from both the CRR logs and the protected resource logs. For more information, see “Problem Management” on page 330.

### Note:

1. The operator must be very careful when determining to either commit or back out the unavailable protected resource. If the operator's heuristic decision is the opposite of the intent of synchronization (sync) point processing, the heuristic decision results in *heuristic damage*.
2. Do not confuse the CRR resynchronization function with the CRR RESYNC command. The CRR resynchronization function is started by the CRR recovery server when an error has occurred during sync point processing. If the error is so severe resynchronization is not able to complete the LUWID instance in a reasonable amount of time, an operator can either enter the CRR RESYNC command to complete resynchronization processing or enter the CRR SUSPEND command to stop the automatic periodic retry of resynchronization processing.

### Operands

#### *token*

identifies the node in the sync point tree that is the target of the heuristic decision. The *token* could be obtained from one of the following:

- Resynchronization messages displayed at the operator's console of this CRR recover server
- Output of the CRR QUERY LU command
- Output of the CRR QUERY LUWID command

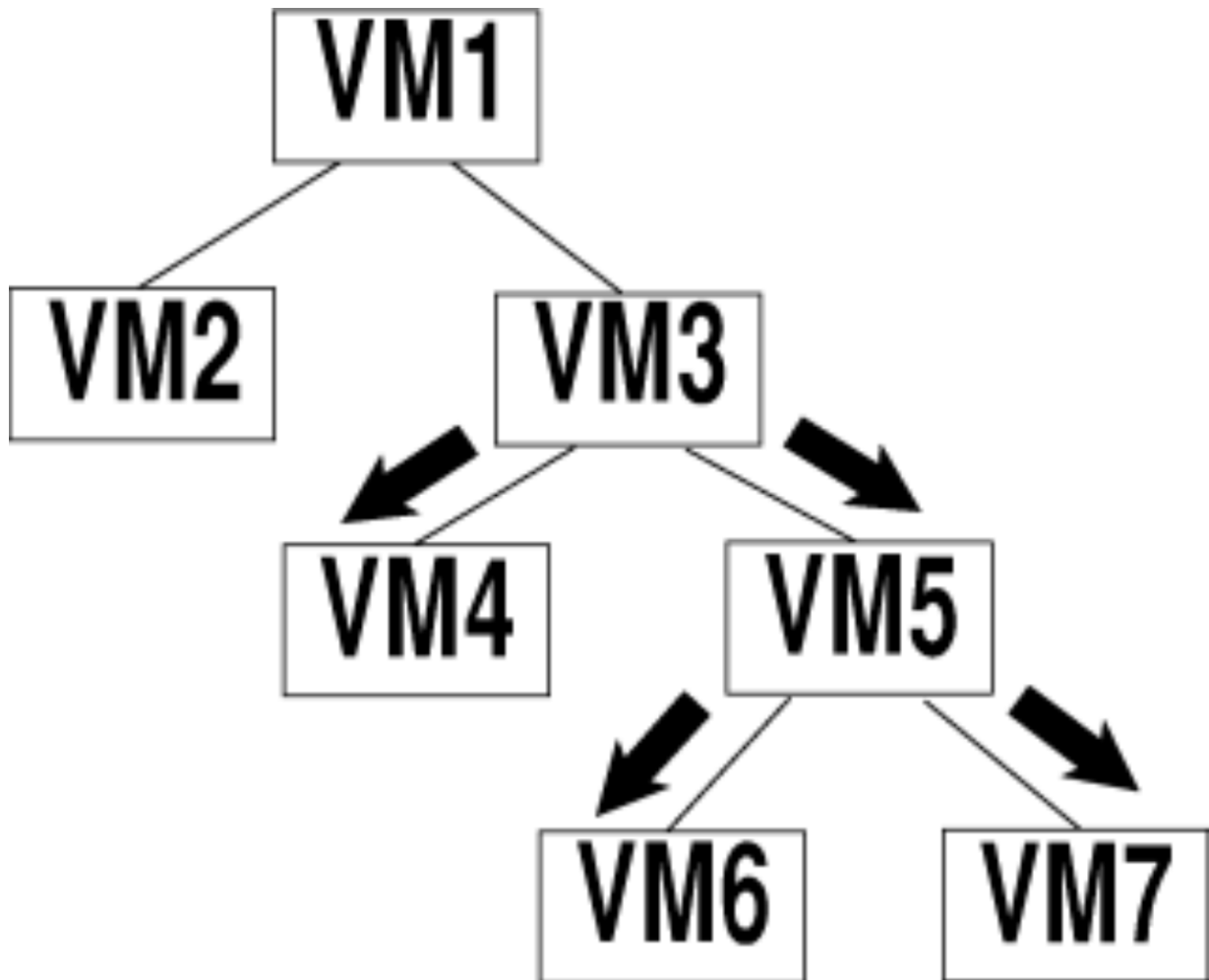
**Note:** The value in *token* identifies an LUWID instance. An LUWID instance is a subset of a coordinated transaction (also called a CRR logical unit of work). The coordinated transaction is identified by the LUWID. The LUWID instance can consist of one or more resource logical units of work.

#### *index*

is the index value shown in the CRR QUERY LUWID display corresponding to agent, initiator, cascaded initiator and protected resource(s) of the target node specified by *token*. The operator must specify the index value, which denotes the communication path the heuristic decision will flow.

If the index denotes an initiator (parent) in the sync point tree, the flow of the heuristic decision is considered to be parental direction from the initiator (target node) down the sync point tree.

See [Figure 51 on page 391](#) for an example of parental direction flow.



*Figure 51. Parental Direction Flow*

VM3 is the initiator that is the target of the CRR RESYNC command. The heuristic decision flows down the sync point tree. In this example, the protected conversation between the initiator at VM1 and the cascaded initiator at VM3 has gone down. The operator uses the CRR RESYNC command to respond for the cascaded initiator at VM3 to drive resynchronization down to VM4 and VM5. VM5 continues the resynchronization down to VM6 and VM7.

If the index denotes a protected resource (child) in the sync point tree, the flow of the heuristic decision is considered to be a child's response from the protected resource up the sync point tree to the initiator (target node).

See [Figure 52 on page 392](#) for an example of a child's response flow.

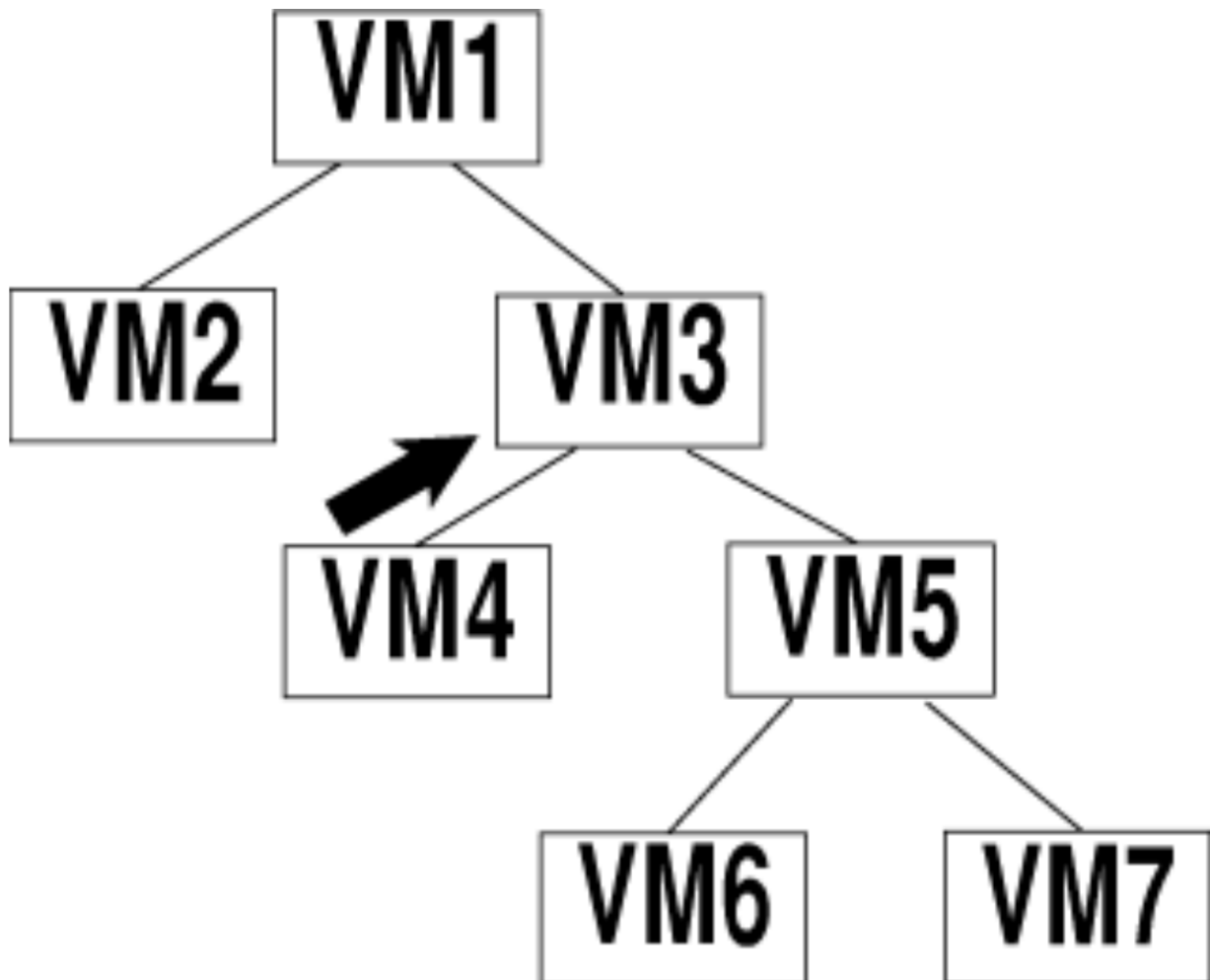


Figure 52. Child's Response Flow

VM3 is the initiator that is the target of the CRR RESYNC command and VM4 contains a protected resource. The heuristic decision flows up the sync point tree to the initiator at VM3. VM3 must be in resynchronization processing and failed at least one attempt to resynchronize the protected resource at VM4 before the operator can enter the CRR RESYNC command at VM3. In this example, the operator at VM3 uses the CRR RESYNC command to respond for the protected resource at VM4 so VM3 can complete resynchronization processing.

#### COMMIT

is the heuristic decision, which is provided to the target node in the sync point tree, as specified by *token* and index to commit the unavailable resource. A response of commit from a child is regarded as a heuristic committed response.

#### BACKOUT

is the heuristic decision, which is provided to the target node in the sync point tree, as specified by *token* and index to back out the unavailable resource. A response of backout from a child is regarded as a heuristic backout response.

#### NOPURGE

indicates when the operator provides the heuristic response, the CRR log records are left in place so subsequent resynchronization processing can proceed normally and clean up the CRR logs.

#### PURGE

indicates when the operator provides the heuristic response, it is known as the last possible response (possibly because permanently broken hardware), and the CRR log records (for the LUWID instance at this node as specified by *token*) are cleaned up as much as possible during command processing.



## Usage Notes

1. It may be necessary to use the CRR ERASE LUWID command and participating resource manager commands (such as SFS's QUERY PREPARED and FORCE PREPARED commands) to manually complete resynchronization for the LUWID instance.
2. As part of the processing of the CRR RESYNC command, resynchronization issue a prompt message to allow the operator to continue or to stop the command. This is required because the CRR RESYNC command can be used to introduce heuristic damage to a sync point tree. The continue or abort prompt reminds the operator of the potentially serious consequences of the CRR RESYNC command and gives the opportunity to suppress command execution.
3. The operator must be careful to understand the implications of using the CRR RESYNC command. The CRR RESYNC command with the PURGE operand breaks off a portion of the sync point tree so it is no longer available to resynchronization. The operator may have to use additional commands to clean up the CRR log entries for the broken-off portion of the sync point tree (for example, CRR ERASE LU, CRR ERASE LUWID). This may also involve cooperation by operators at other processors and SNA network nodes. The purging of obsolete CRR log records may also be spread over time because unavailability of some processors or SNA network nodes. It is the operator's responsibility to coordinate this manual process. For more information, see [“Step 7: Take Heuristic Action”](#) on page 336.
4. If CRR security audit tracing is enabled, the CRR RESYNC command generates an audit trace record.

## CRR SUSPEND

► CRR — SUSPEND — *token* ◄

### Authorization

CRR Operator

### Purpose

Use the CRR SUSPEND operator command to stop the automatic periodic retry of resynchronization for the logical unit of work ID (LUWID) instance specified by *token*. This changes the status of the specified LUWID instance from timed wait to suspended.

### Operands

#### *token*

identifies the LUWID instance to be suspended from the automatic periodic retry of resynchronization. The *token* could be obtained from one of the following:

- Resynchronization messages displayed at the operator's console of this CRR recovery server
- Output of the CRR QUERY LU command
- Output of the CRR QUERY LUWID command

**Note:** The value in *token* identifies an LUWID instance. An LUWID instance is a subset of a coordinated transaction (also called a CRR logical unit of work). The coordinated transaction is identified by the LUWID. The LUWID instance can consist of one or more resource logical units of work.

### Usage Notes

1. The automatic periodic retry of resynchronization consists of the following steps:
  - a. The CRR recovery server attempts resynchronization of the LUWID instance.
 

If the protected resources or protected conversations are available and resynchronization completes, this LUWID instance is no longer in automatic periodic retry of resynchronization.

If the protected resources or protected conversations are not available and therefore resynchronization does not complete, this LUWID instance goes into a timed wait.
  - b. The LUWID instance remains in timed wait for a time period (called timed wait interval) established by the RESYNCINTERVAL startup parameter. At the end of the timed wait interval, there is another attempt at resynchronization. Go to step [“1.a” on page 394](#).
2. The CRR SUSPEND command should only be issued while the LUWID instance is in timed wait. You can determine if the LUWID instance is in a timed wait by examining the *current\_status* value in the display from either the CRR QUERY LU or CRR QUERY LUWID command. If you use the CRR SUSPEND command to suspend an LUWID instance already suspended, the CRR SUSPEND command will complete successfully with no change to the state of the LUWID instance.
3. The operator uses the CRR QUERY LU command to determine if the specified LUWID instance now has the protected resources or protected conversations available to continue resynchronization. If the protected resources or protected conversations are available, using the CRR RESUME command makes the LUWID instance available for the automatic periodic retry of resynchronization in this CRR recovery server. If possible, resynchronization immediately continues. If resynchronization does not find all protected resources or protected conversations available to process the LUWID instance, the timed wait continues for the LUWID instance and you may want to use the CRR SUSPEND command. Using the CRR SUSPEND command temporarily withdraws an LUWID instance from timed wait and

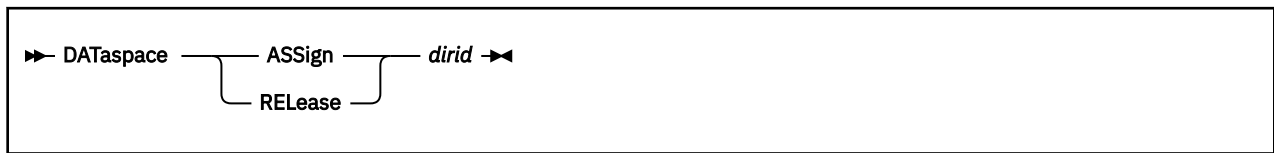
puts the LUWID instance into a suspended status, which makes the LUWID instance unavailable for the automatic periodic retry of resynchronization.

The timed wait interval, which is determined by the RESYNCINTERVAL startup parameter, is the amount of time the LUWID instance remains in timed wait before leaving timed wait to make the next attempt at resynchronization. The default timed wait interval is 10 minutes.

If you want to change the timed wait interval between attempts to resynchronize, you must use the RESYNCINTERVAL startup parameter. For more information on the RESYNCINTERVAL startup parameter, see Chapter 20, [“File Pool Server Startup Parameters,”](#) on page 339. For more information on the instructions on changing the timed wait interval, see [“Bypass and Change Timed Wait Interval”](#) on page 325.

4. The suspended status of the LUWID instance ends and the automatic periodic retry of resynchronization starts if:
  - The CRR RESUME command is issued, or
  - Another resynchronization transaction is initiated for this LUWID instance. For example, when there is a TPN X'06F2' to exchange log names in an upstream shoulder tap. An upstream shoulder tap is when the unavailable protected resource becomes available and connects to its initiator or cascaded initiator letting it know the protected resource is now available. The initiator or cascaded initiator ends the suspended status and goes into the automatic periodic retry of resynchronization.
5. Suspension is not persistent across CRR recovery server shutdown or error. When a CRR recovery server is restored to service after a shutdown or an error, all LUWID instances reflected in its CRR log that were in suspended status are now in timed wait.

## DATASPACE



### Authorization

Repository File Pool Administrator

### Purpose

Use the DATASPACE command to make a directory control directory eligible for use in a data space. *Directory*, in this case, means the contents of the directory and all the file data within the directory. The DATASPACE command is also used to remove eligibility. You must have file pool administration authority to issue this command.

### Operands

#### ASSign

makes the specified directory eligible for use in a data space. The directory you specify must be a directory control directory. File control directories cannot be mapped to data spaces.

#### RELease

removes data space eligibility for the specified directory.

#### *dirid*

identifies the SFS directory for which you are assigning or releasing eligibility.

### Usage Notes

- When a directory is made eligible, that eligibility persists (even across server shutdowns) until one of these events takes place:
  - Eligibility is explicitly removed by a DATASPACE RELEASE command.
  - The directory is erased.
  - The directory attribute is changed from DIRCONTROL to FILECONTROL.
- If the DATASPACE command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will not be successful.
- Even if the server is not running in an XC virtual machine, you can still use the DATASPACE command. Directory control directories can exist and can be assigned data space eligibility when the server is running in an XA virtual machine. However, the server cannot actually use data spaces unless it is running in an XC virtual machine.
- When you change data space eligibility, the change applies to new accesses. Existing accesses are not affected.
- Whether a directory is assigned to a data space has no effect on the way its function appears to the user. Data space eligibility is purely a performance option.

Even if the directory is eligible for data space use, however, there is no guarantee a performance improvement will be attained. Nor is there a guarantee a data space will be used at all. Several factors determine whether an eligible directory will actually reside in a data space and derive performance benefits:

- The architecture (XA or XC) of the virtual machine in which the server is running

- Access mode

Only users who access (using the CMS ACCESS or VMLINK commands) the directory in read-only mode benefit from data spaces.

- Availability of data spaces

If not enough data spaces are available, the directory may not be loaded into a data space. So, performance is not improved for the directory.

Too much writing to a directory in a data space can exhaust the number of available data spaces. Each time a file in a directory is changed, SFS creates a new version of the directory while maintaining older versions as long as they remain accessed. Each time the directory is changed and then accessed read-only, the latest version of the directory is mapped into a new data space, until all suitable data spaces are in use. Thereafter, new read-only accessors do not use the directory in a data space until a sufficiently large data space becomes available.

- Location of the user relative to the server machine

Only users on the same system as the file pool server machine can acquire directories in data spaces.

6. The server is always used when reading from or writing to files that have the INPLACE attribute. Data spaces are not used for INPLACE files.

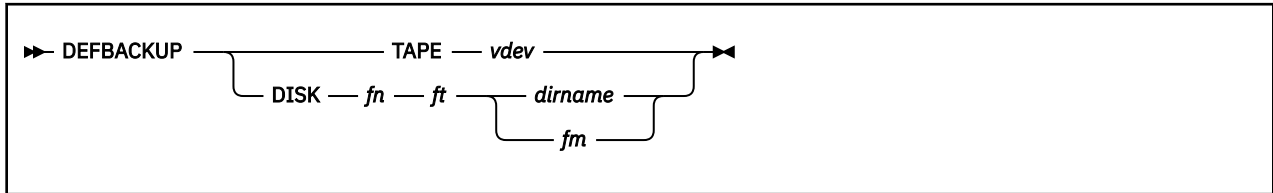
## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS and REXX/VM Messages and Codes](#).

Messages:

- DMS1139E You are not authorized to enter this command [RC=76]
- DMS1210E Directory *dirname* not found [RC=28]
- DMS2023E File pool *filepoolid* does not support the requested *command* command [RC=88]
- DMS2037E Directory *dirname* is not a directory control directory [RC=40]

## DEFBACKUP



### Authorization

Repository File Pool Operator

### Purpose

Use the DEFBACKUP operator command to change the assignment of the file pool control data backup file. The assignment will remain in effect until another DEFBACKUP is entered, or until multiple user mode file pool server processing ends.

### Operands

#### TAPE *vdev*

identifies the virtual device number *vdev* for a standard label (SL) tape file. All subsequent control data backups will be directed to the identified tape device. If an assignment for the control data backup file already exists, it is replaced.

#### DISK *fn ft dirname*

#### DISK *fn ft fm*

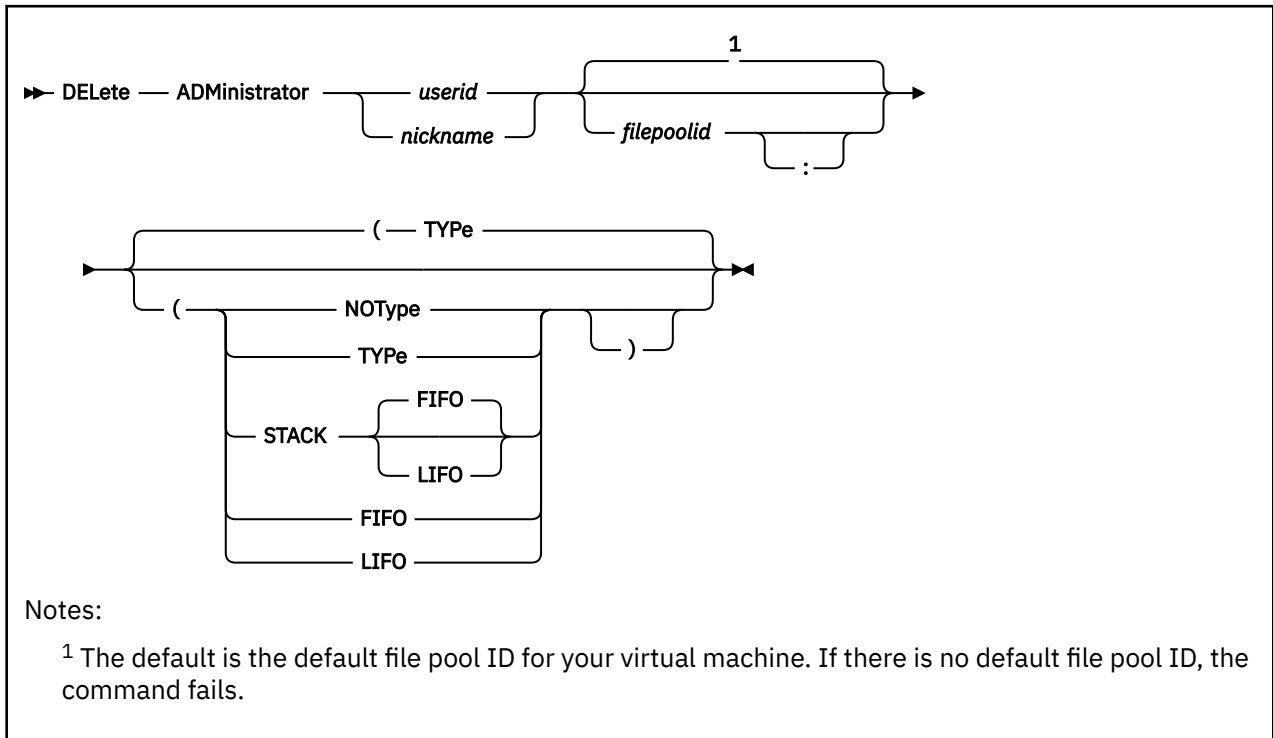
identifies a CMS file to which the output of subsequent control data backups will be directed. The file can reside on minidisk or in another file pool. The *dirname* is the fully qualified directory name *filepoolid: userid n1.n2. ...n8* of the SFS directory where the backup file is to be created. The *fm* denotes the file mode of an accessed CMS minidisk or SFS directory. If an assignment for the control data backup file already exists, it is replaced.

### Usage Notes

1. For a list of supported virtual device numbers for tapes, see the description of the FILEDEF command in *z/VM: CMS Commands and Utilities Reference*.
2. All messages are written to the server machine operator console.
3. For CRR recovery servers, IBM does not recommend doing backups of control data (that is, use the NOBACKUP startup parameter).
4. This command does not change the contents of the POOLDEF file. The next time the server is started, the default destination will be determined from the contents of the POOLDEF file.
5. The server verifies the destination for the backup file is valid. If a problem is detected, a warning message is issued telling you to either correct the problem or provide a valid destination for the next backup. The previous backup file destination remains in effect.
6. The default control data backup file destination can be invalidated if an automatic backup fails to complete.
7. You do not need to enter a FILEDEF command for ddname BACKUP. The server creates the proper environment for the backup processing.
8. You can find out the current default assignment for the control data backup file, or where the last successful control data backup file was created, by issuing the QUERY DEFBACKUP operator command.
9. If the control backup is directed to tape, it is assumed the tape will be ready when the backup starts. If it is not, message DMS113S will be issued and control backup will stop.

10. It is not allowed to direct the control backup to a file that already has established data definitions from the FILEDEF command. If this is attempted, backup processing will stop. Note that the file pool server enters a FILEDEF command for security audit (AUDIT) when AUDIT is specified in the DMSPARMS file.

## DELETE ADMINISTRATOR



### Authorization

File Pool Administrator

### Purpose

Use the DELETE ADMINISTRATOR command to revoke administration authority from a user for a particular file pool. You must have administration authority to use this command.

### Operands

#### *userid*

identifies the user ID to be deleted.

#### *nickname*

identifies a nickname that represents a user ID or a list of user IDs. (Use the NAMES command to define nicknames.)

#### *filepoolid*

#### *filepoolid:*

identifies the file pool for which the user is to lose administrative authority. If not specified, the default file pool ID for your virtual machine will be used.

### Options

#### TYPE

displays at the terminal the user IDs from which administration authority is being revoked. TYPE is the default.

#### NOType

suppresses the display of the user IDs from which administration authority is being revoked.



**STACK FIFO****STACK LIFO**

places the user IDs from which administration authority is being revoked in the console stack rather than displaying it at the terminal. FIFO is the default.

**LIFO**

stacks the user IDs from which administration authority is being revoked in a last in, first out order. This option is equivalent to STACK LIFO.

**FIFO**

specifies that the user IDs from which administration authority is being revoked should be stacked in a first in, first out order. This option is equivalent to STACK FIFO.

**Usage Notes**

1. You cannot delete yourself.
2. If the DELETE ADMINISTRATOR command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
3. Administrators can be deleted regardless of how they gained their authority. (A user can be granted administration authority at startup, through the ENROLL ADMINISTRATOR command, or through the GRANT ADMIN file pool server operator command.)
4. If the deleted user ID is specified in an ADMIN startup parameter, that user ID will be given administration authority again the next time file pool server processing is started.
5. If you specify a nickname that represents a list of users, and the delete errors for any user in the list, the entire command will be unsuccessful. No administrators are deleted. Command processing stops after the first error.
6. If you specify a nickname and the NODE tag in the NAMES file indicates the user is on another processor, the LOCALID tag must also be specified.
7. DELETE ADMINISTRATOR does not cause a DELETE USER. When a user's administration authority is deleted, it does not affect any other authorizations the user has. Only the administration authority is deleted.
8. You can delete administrators only for a file pool in which you have administration authority. Administration authority does not span file pools.
9. If the specified administrator is accessing other users' directories (for which he or she has no explicit authorizations), those directories remain accessed until the end of the CMS session.

**Note:** All authority checking for a DIRCONTROL directory is on access. Therefore, if a user accesses a DIRCONTROL directory by his administrator authority and his administrator authority is revoked, his authority on that directory remains until access of the DIRCONTROL directory is released.

10. For FILECONTROL directories, if a user has opened a file he has authority for because he is an administrator, and his administrator authority is revoked before his changes are committed, he can still commit the data but will be unable to reopen the file.
11. When DELETE ADMINISTRATOR is entered, explicit locks acquired by the specified *userid* are not affected. That is, the DELETE ADMINISTRATOR command does not automatically delete locks. The user can remove them if he or she is explicitly enrolled or if PUBLIC is enrolled. Or, another file pool administrator can delete the locks.

When administrator authority is deleted, the user may find he or she cannot use the QUERY LOCK command to view locks he or she created. This happens if the user has no explicit authority on the object and was able to create the lock only by virtue of administration authority. After administration authority is revoked, the QUERY LOCK command does not display the locks for the user because the user has no authorization on the file or directory. Although the user cannot display the lock, the user can still enter a DELETE LOCK command to delete it.

12. You can determine which user IDs are enrolled as administrators by entering the following command:

```
query enroll administrator for all vmsys
```

Substitute your own file pool ID for VMSYS.

A description of the QUERY ENROLL command is in [z/VM: CMS Commands and Utilities Reference](#).

### Messages and Return Codes

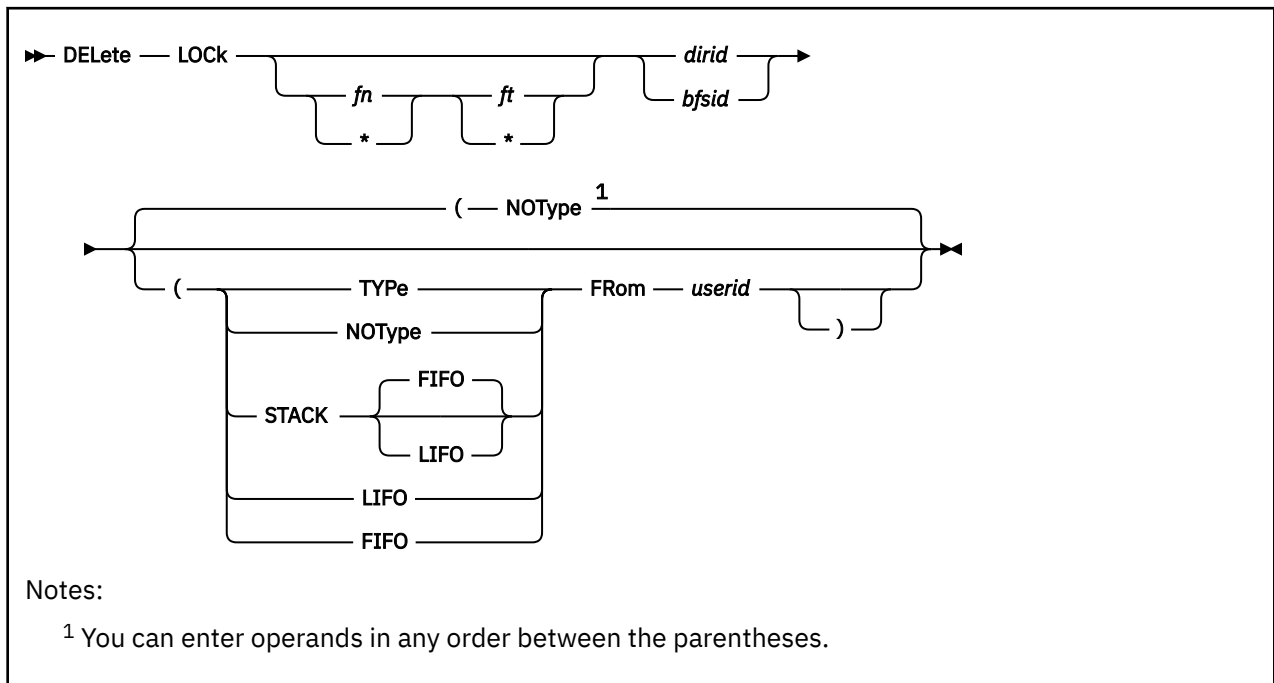
In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS and REXX/VM Messages and Codes](#).

This command internally issues IDENTIFY, FINIS, NUCXDROP, and NUCXLOAD commands. Therefore, messages from those commands can also be displayed. (For more information about those commands, see [z/VM: CMS Commands and Utilities Reference](#).)

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS066E *option1* and *option2* are conflicting options [RC=24]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS149E Userid *userid* not valid [RC=32]
- DMS637E Missing nodeid for the AT operand [RC=24]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=32]
- DMS653E Error executing *command rc=nn* [RC=40]
- DMS1172E You are not allowed to delete your own userid [RC=76]
- DMS1201E STACK option cannot follow FIFO or LIFO [RC=24]
- DMS1224W One or more userids were not enrolled as ADMINISTRATORs [RC=04]

## DELETE LOCK



### Authorization

Repository File Pool Administrator

### Purpose

Use the DELETE LOCK command to release an explicit lock on a Shared File System (SFS) directory, an SFS file, or a Byte File System (BFS) regular file. (Explicit locks are created by the CREATE LOCK command, which is described in [z/VM: CMS Commands and Utilities Reference](#).) This command provides function equivalent to the CSL Routine DMSDELOC - Delete Lock, which is described in [z/VM: CMS Callable Services Reference](#).

While administration authority is not required when this command is entered for one's own locks on SFS objects, it is needed if you want to delete a lock someone else has created. A file pool administrator can delete locks for BFS files. It may be necessary for you (as an administrator) to delete a lock on a needed object when the lock creator is not available.

### Operands

#### *fn ft*

identifies the file name and file type of the file to be unlocked. Pattern-matching characters (\* and %) can be specified to unlock a set of files. For more information about pattern-matching characters, see [z/VM: CMS Commands and Utilities Reference](#).

#### *dirid*

identifies a directory. If you specify *fn ft*, the *dirid* identifies the directory containing the file(s) to be unlocked. If *fn ft* are omitted, the directory itself is unlocked.

#### *bfsid*

identifies the Byte File System.

## DELETE LOCK

### Options

#### **NOType**

suppresses the display of file names, file types, and directory names for which locks are being deleted. NOType is the default.

#### **TYPe**

displays at the terminal the file names, file types, and directory names for which locks are being deleted.

#### **STACK FIFO**

#### **STACK LIFO**

places the file names, file types, and directory names for which locks are being deleted in the console stack rather than displaying them at the terminal. FIFO is the default.

#### **FIFO**

specifies the file names, file types, and directory names for which locks are being deleted should be stacked in a first in, first out order. This option is equivalent to STACK FIFO.

#### **LIFO**

specifies the file names, file types, and directory names for which locks are being deleted should be stacked in a last in, first out order. This option is equivalent to STACK LIFO.

#### **FRom *userid***

indicates the userid from whom the lock is to be released. Only a file pool administrator can use this option.

### Usage Notes

1. If pattern-matching characters (\* or %) are used in *fn* or *ft*, locks are not deleted in the directory for:
  - Sub-directories
  - Erased or revoked aliases
  - Files you are not authorized to read or writeA message is displayed for erased or revoked aliases. Processing continues for the remaining file names that match the specified pattern.
2. Only the creator of a lock can delete it, unless a user with administrator authority uses the FROM option.
3. If the DELETE LOCK command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will be unsuccessful.
4. You can issue the DELETE LOCK command from the command line, from an exec, or as a function from a program. No error messages are issued if DELETE LOCK is issued:
  - As a function from a program
  - From a CMS exec file that has the &CONTROL NOMSG option in effect
  - From an EXEC 2 exec where CMDCALL is not in effect
  - From a REXX exec with ADDRESS COMMAND in effect
5. Use the QUERY LOCK command to display information about the locks that have been created on files and directories. The QUERY LOCK command is described in [z/VM: CMS Commands and Utilities Reference](#).
6. If you specify a nickname and the NODE tag in the NAMES file indicates the user is on another processor, you must also specify the LOCALID tag.
7. DELETE LOCK may be used to delete an explicit lock on a Byte File System file; however, the CMS short file name format of the file must be used. (The CMS short name identifies the CMS file name and file type of an object in the backup file that is unique within a BFS to each file.) You may not enter the DELETE LOCK command using the fully qualified BFS path name. Pattern matching is also not allowed.

## Messages and Return Codes

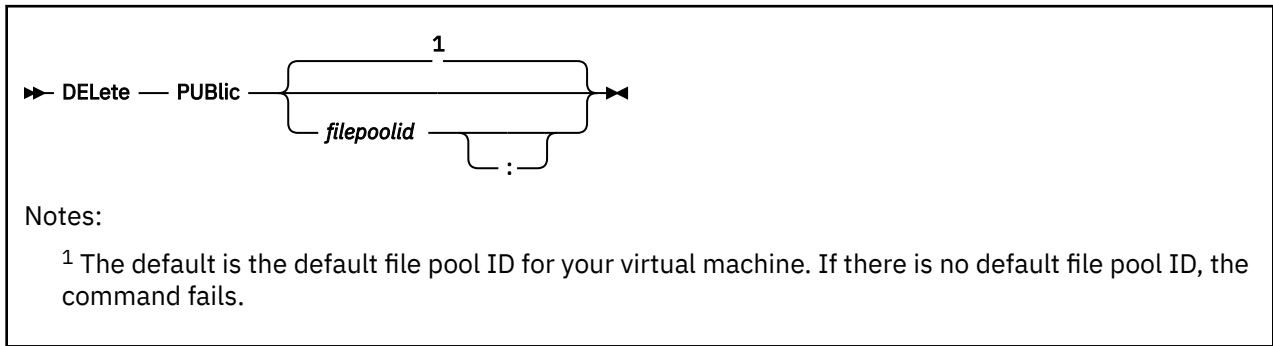
In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS and REXX/VM Messages and Codes](#) and [z/VM: CMS Commands and Utilities Reference](#).

This command internally issues IDENTIFY, FINIS, NUCXDROP, and NUCXLOAD commands. Therefore, messages from those commands can also be displayed. (For more information about those commands, see [z/VM: CMS Commands and Utilities Reference](#).)

Messages:

- DMS002E File *fn ft {fm/dirname}* not found [RC=28]
- DMS069E Filemode *mode* not accessed [RC=36]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS149E Userid *userid* not valid [RC=32]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=32]
- DMS653E Error executing *command rc=nn* [RC=40]
- DMS1132E Incorrect number of operands [RC=24]
- DMS1139E You are not authorized to enter this command [RC=76]
- DMS1160E Directory *dirname* already open. [RC=70]
- DMS1163E The DELETE LOCK command failed for *{fn ft fm/dirname}* [RC=28, 76, or 70]
- DMS1184E Directory *dirname* not found or you are not authorized for it [RC=28]
- DMS1184E File *fn ft* or directory *dirname* not found or you are not authorized for it [RC=28]
- DMS1187E Too many subdirectory levels in *dirid* [RC=24]
- DMS1188E Filemode *fm* is not associated with a directory [RC=74]
- DMS1189E Filemode *fm* is associated with a top directory [RC=24]
- DMS1206W No locks are held by *userid* for *fn ft {fm/dirname}* [RC=4 ]
- DMS1209E Nickname *nickname* resolved to more than one userid; lock(s) can be deleted from only one userid at a time [RC=88]
- DMS1210E Directory *dirname* not found [RC=28]
- DMS1291E There are no unused work units available. [RC=88]
- DMS1223E There is no default file pool currently defined [RC=40]

## DELETE PUBLIC



### Authorization

Repository File Pool Administrator

### Purpose

Use the DELETE PUBLIC command to remove the connect authority given to public on the ENROLL PUBLIC command. Administration authority is required to use the command.

### Operands

*filepoolid*

***filepoolid:***

is the ID of the file pool from which the public enrollment is to be removed. If not specified, the default file pool ID for your virtual machine will be used.

### Usage Notes

1. This command does not affect the users individually enrolled by the ENROLL USER command.
2. When DELETE PUBLIC is entered, explicit locks created by users who were not explicitly enrolled in the file pool are not automatically deleted. A user having file pool administration authority can delete the locks.
3. When the DELETE PUBLIC command is entered, connections are not broken for users who are currently connected to the file pool. Once their connections are severed for any reason (for example, the user re-IPLs CMS), they cannot reconnect to the file pool unless they are explicitly enrolled.
4. To determine whether ENROLL PUBLIC has been entered for a file pool, enter a QUERY ENROLL USER FOR ALL command. If ENROLL PUBLIC is in effect, the QUERY command will show <PUBLIC> in its output.

Note that if a user ID of PUBLIC is enrolled, *PUBLIC* (without the < and > characters) will be displayed. If both ENROLL USER PUBLIC and ENROLL PUBLIC have been entered for the file pool, you will see both *PUBLIC* and <*PUBLIC*> in the QUERY ENROLL output. The QUERY ENROLL command is described in [z/VM: CMS Commands and Utilities Reference](#).

5. If the DELETE PUBLIC command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will fail.

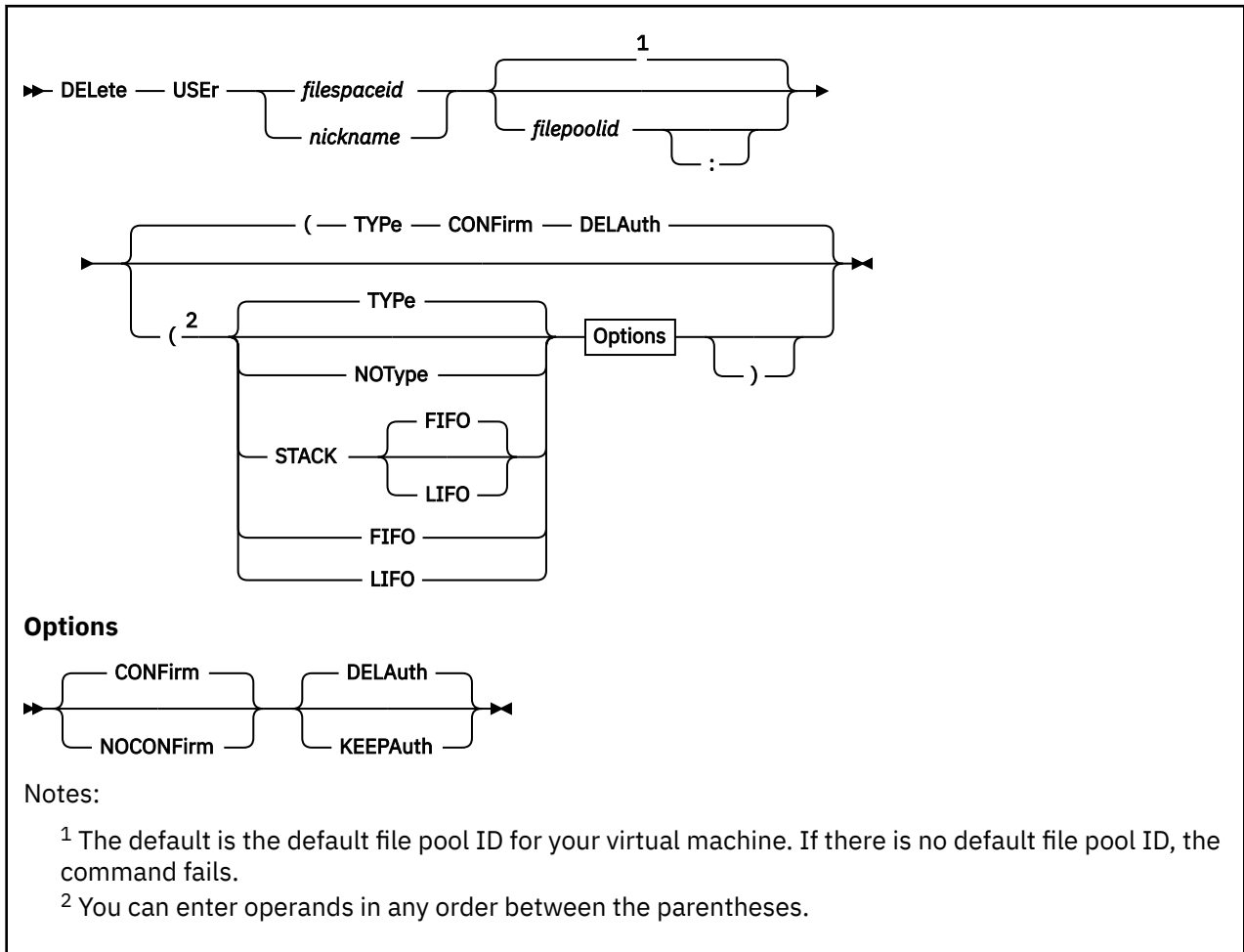
### Messages and Return Codes

In addition to the message listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS and REXX/VM Messages and Codes](#).

Messages:

- DMS1169W Public connect authority has not been established [RC=04]

## DELETE USER



### Authorization

Repository File Pool Administrator

### Purpose

Use the DELETE USER command to remove a user or Byte File System (BFS) from a particular file pool. This command provides function equivalent to the CSL Routine DMSDEUSR - Delete File Space, which is described in [z/VM: CMS Callable Services Reference](#).

### Operands

#### *filespaceid*

identifies the file space (user ID or Byte File System) to be deleted.

#### *nickname*

is a nickname for a user ID, a Byte File System, or a list of user IDs or a list of Byte File System names. (Use the NAMES command to define nicknames.)

#### *filepoolid*

#### *filepoolid:*

identifies the file pool from which the file space is to be removed. If not specified, the default file pool ID for your virtual machine is used.



## Options

### TYPE

displays at the terminal the file spaces to be deleted (if CONFIRM is specified), or that have already been deleted (if NOCONFIRM is specified and the TYPE option is specified or allowed to default).

### NOTYPE

suppresses the display of file spaces.

### STACK FIFO

### STACK LIFO

places the list of file spaces in the console stack rather than displaying it at the terminal. FIFO is the default.

### LIFO

specifies the list of file spaces is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

### FIFO

specifies the list of file spaces is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

### CONFIRM

causes a confirmation prompt to be displayed. It asks you to confirm the delete is desired. You can then confirm the delete or cancel it. Unless NOTYPE is specified, a list of file spaces to be deleted will be displayed as part of the prompt.

### NOCONFIRM

indicates no confirmation prompt should be issued. Unless NOTYPE is specified, a list of file spaces that were deleted will be displayed at the end of successful command processing.

### DELAUTH

deletes all SFS authorizations granted to the file space being removed. This is the default.

### KEEPAUTH

keeps all SFS authorizations granted to the file space being removed.

## Usage Notes

### Common Notes for SFS and BFS

1. If a *nickname* is specified that represents a list of file spaces, and the deletion of one of the file spaces in the list is unsuccessful any file spaces in the list before this file space are deleted. If TYPE, STACK, LIFO, or FIFO is specified, a list of file spaces IDs that were deleted is typed or stacked.
2. When CONFIRM is specified or allowed to default, a prompt is displayed. The prompt asks whether you really want to delete the file spaces or whether you want to cancel the deletion. The DELETE USER command reads the response to this message directly from the terminal — it bypasses the program stack. For this reason, it is recommended that programs or execs that issue this command should specify NOCONFIRM to avoid the prompt.
3. If the DELETE USER command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
4. If you specify a nickname and the NODE tag in the NAMES file indicates the user is on another processor, you must also specify the LOCALID tag.
5. Trying to delete a file space not enrolled in the file pool produces a warning message.
6. A user with file pool administration authority can delete himself from the file pool. That user does not lose administration authority. He or she can still connect to the file pool and exercise any functions permitted by administration authority.

### Notes for SFS only

## DELETE USER

1. All resources belonging to the user are freed. Base files, aliases, and directories the user owned are removed from the file pool. Any authorities the user granted on his or her files and directories are also removed. Any aliases other users had on the deleted user's files are erased.
2. When DELAUTH is specified or allowed to default, all authorizations granted to the deleted user ID are also deleted. If KEEPAUTH is specified, the authorizations granted to the user ID are kept. Use the KEEPAUTH operand if you are removing the file space data and objects but wish to have the user ID remain authorized to other objects in the file pool.
3. DELETE USER is unsuccessful if:
  - The user to be deleted has a DIRCONTROL directory that is accessed read/write by someone other than the issuer of the DELETE USER request.
  - Any of the user's files or directories are open or *implicitly* locked (by **anyone**, including the issuer).
  - There are any uncommitted changes for the user's files or directories (there is an active work unit).

If the user is in an XEDIT session, the DELETE USER usually succeeds because files are not implicitly locked for the duration of the XEDIT session. (Files are often explicitly locked during an XEDIT session, but that does not prevent DELETE USER from succeeding.)
4. DELETE USER breaks the connection to the file pool for the users being deleted, unless PUBLIC has been enrolled in the file pool or the user has file pool administration authority.

### Notes for BFS only

1. All resources belonging to the file system are freed. Files, directories, special files and links are removed from the file pool.
2. When DELAUTH is specified or allowed to default, all SFS authorizations granted to the deleted file space are also deleted. If KEEPAUTH is specified, the SFS authorizations granted to the file space ID are kept. Use the KEEPAUTH operand if you are removing the file space data and objects but wish to have the file space ID remain authorized to other objects in the file pool.
3. DELETE USER is unsuccessful if:
  - Any of the user's files or directories are open or *implicitly* locked (by **anyone**, including the issuer).
  - There are any uncommitted changes for the user's files or directories.
4. Users who have the Byte File System mounted will receive a message indicating the Byte File System is not available if they attempt to use it after it has been deleted.

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS and REXX/VM Messages and Codes](#) and [z/VM: CMS Commands and Utilities Reference](#).

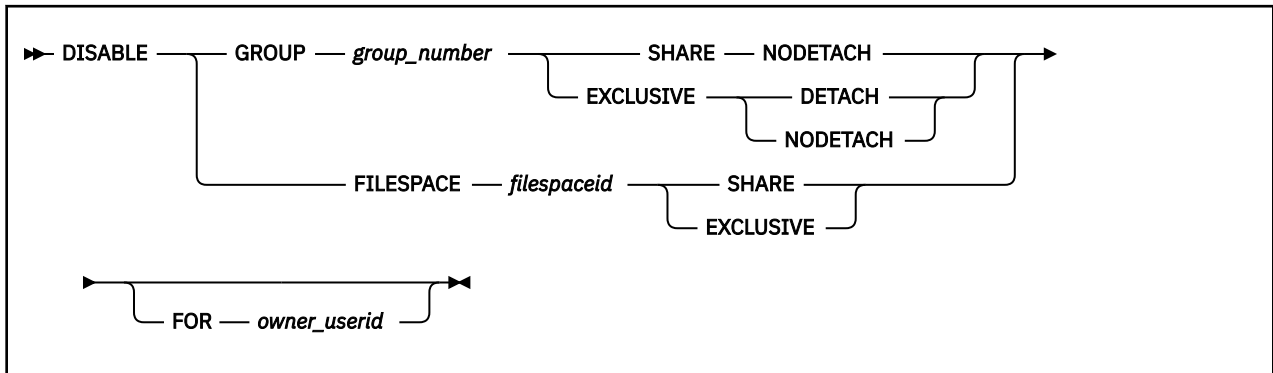
This command internally issues IDENTIFY, FINIS, NUCXDROP, and NUCXLOAD commands. Therefore, messages from those commands can also be displayed. (For more information about those commands, see [z/VM: CMS Commands and Utilities Reference](#).)

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS066E *option1* and *option2* are conflicting options [RC=24]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS149E Userid *filepaceid* not valid [RC=32]
- DMS637E Missing nodeid for the AT operand [RC=24]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=32]
- DMS653E Error executing *command* rc=*nn* [RC=40]
- DMS789E Incorrect response [RC=24]

- DMS925E I/O error on screen [RC=100]
- DMS1173E Userid *filespaceid* cannot be deleted because the file space is currently in use [RC=70]
- DMS1201E STACK option cannot follow FIFO or LIFO [RC=24] [RC=24]
- DMS1215E {File *fn ft dirname*|Directory *dirname*} is locked by another user [RC=70]
- DMS1224W One or more user IDs were not enrolled [RC=4]
- DMS2023E File pool *filepoolid* does not support the requested option [RC=88]
- DMS2031E Mixing operations on SFS objects and BFS objects is incorrect within a single work unit [RC=70]

## DISABLE



### Authorization

Repository File Pool Operator

### Purpose

Use the DISABLE operator command to disable a storage group or file space for write access (SHARE) or all access (EXCLUSIVE). When all access is prohibited (EXCLUSIVE), the storage group minidisks may also be detached. The disable may optionally be done on behalf of another user by specifying the user ID. This command provides function equivalent to the CSL Routines DMSDISFS - Disable File Space and DMSDISSG - Disable Storage Group which are described in [z/VM: CMS Callable Services Reference](#).

### Operands

#### GROUP *group\_number*

indicates the number of a storage group to be disabled. *group\_number* can be any number from 2 to the MAXDISKS value specified for the file pool. The storage group specified must, however, exist (that is, it must have a minidisk assigned to it).

For more information, see [MAXDISKS](#).

#### FILESPACE *filepaceid*

identifies the file space which is to be disabled. Do not specify a nickname. Nicknames are not recognized on server operator commands.

A file space can be either:

- an SFS file space, or
- a Byte File System (BFS)

#### SHARE

indicates a *share disable lock* should be acquired on the file space or storage group. A share disable lock allows users to read, but not modify, items in the locked object. Files also cannot be put into migrated status, recalled or erased by any user.

#### EXCLUSIVE

indicates an *exclusive disable lock* should be acquired on the file space or storage group. An exclusive disable lock prevents anyone other than the owner of the lock from reading or modifying the items in the locked object. Files also cannot be put into migrated status, recalled, or erased by other users.

#### DETACH

this operand causes the server to detach the minidisks associated with a storage group being disabled. This parameter is valid only when a storage group is being disabled and the EXCLUSIVE option is specified. The minidisks are re-attached when the storage group is enabled.

**Note:** After the storage group is disabled with DETACH, its files cannot be put into migrated status, recalled or erased.

### **NODETACH**

causes the server to keep the links to the storage group minidisks. This parameter is valid only when a storage group is being disabled.

### **FOR owner\_userid**

identifies a user ID on whose behalf the DISABLE is being done. If FOR *owner\_userid* is omitted, the DISABLE will be done on behalf of the user ID of the server machine. Do not specify a nickname. Nicknames are not recognized on server operator commands.

If FOR *owner\_userid* is specified when disabling a storage group, the *owner\_userid* must have administration authority.

If FOR *owner\_userid* is specified when disabling a file space and *owner\_userid* is not the same as the *userid* specified on the FILESPACE operand, the *owner\_userid* should have (or be scheduled to have) administration authority. Otherwise, the *owner\_userid* will not be able to enable the file space. If the *owner\_userid* and the *filespaceid* in the FILESPACE operand are the same, the user ID does not need administration authority. The user can enable his or her own file space by coding a program or REXX exec that calls the DMSENAFS Callable Services Library (CSL) routine.

## **Usage Notes**

1. All messages are written to the file pool server operator console.
2. The lock acquired by the DISABLE command persists until an ENABLE command or routine is called for the storage group or file space.
3. The FOR *owner\_userid* operand identifies the user ID that is to own the lock. If it is omitted, the server machine is the owner. In this case, there are two ways to allow users to access the locked object:
  - a. By entering the ENABLE operator command.
  - b. By entering the FILEPOOL ENABLE administrator command (or the Enable CSL routine) for the object, specifying the server machine user ID as the owner of the lock.
4. It is possible for this operator command to be unsuccessful:
  - Because a lock needed to complete the command is held by some user
  - Or because the file space or storage group contains an SFS directory control directory that is accessed read/write by another user or read-only by you. (Other users accessing the directory in read-only mode lose access, without any message.) If you are accessing the directory, it must be in read/write mode.

In this case, you should enter the command again at a later time. (The server doesn't wait for the lock to become free because its operator console would be unavailable during the wait.)

5. Callable Services Library (CSL) routines are available in VMLIB that disable and enable storage groups and file spaces. They are:

**DMSDISFS** disables a file space  
**DMSDISSG** disables a storage group  
**DMSENAFS** enables a file space  
**DMSENASG** enables a storage group

These routines can be called from application programs that call in an administration machine. See [z/VM: CMS Callable Services Reference](#) for more about these routines.

6. If a file space or storage group is disabled, DFSMS/VM file migration, recall, and expiration cannot be performed for files in that file space or storage group. However, if the file space or storage group is disabled exclusive, the owner of the disable lock can start DFSMS/VM file migration or recall.
7. File pool administrator commands can be issued to disable and enable storage groups and file spaces. They are:

**FILEPOOL DISABLE FILESPACE** disables a file space

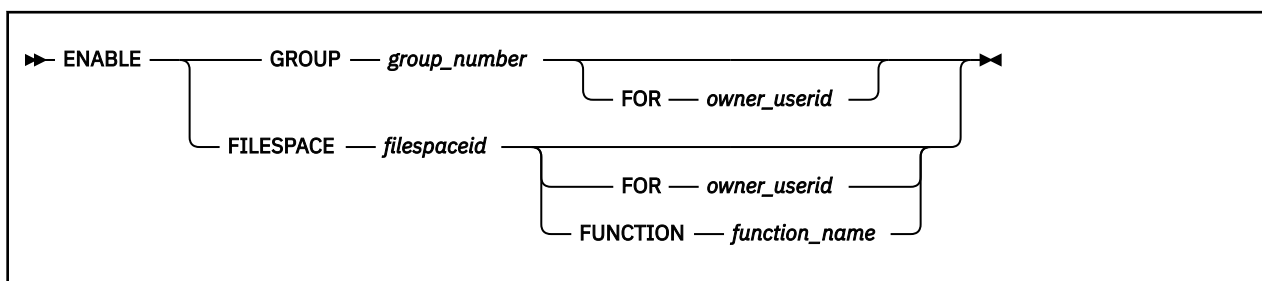
## DISABLE

**FILEPOOL DISABLE GROUP** disables a storage group

**FILEPOOL ENABLE FILESPACE** enables a file space

**FILEPOOL ENABLE GROUP** enables a storage group

## ENABLE



### Authorization

Repository File Pool Operator

### Purpose

Use the ENABLE operator command to reinstate use of a disabled storage group or file space. This command provides function equivalent to the CSL Routines DMSENAFS - Enable File Space and DMSENASG - Enable Storage Group which are described in [z/VM: CMS Callable Services Reference](#).

### Operands

#### **GROUP** *group\_number*

indicates the number of a storage group to be enabled. *group\_number* can be any number from 2 to the MAXDISKS value specified for the file pool. For more information, see [MAXDISKS](#).

#### **FILESPACE** *filepaceid*

identifies a file space which is to be enabled. Do not specify a nickname. Nicknames are not recognized on server operator commands.

A file space can be either:

- an SFS file space, or
- a Byte File System (BFS)

#### **FOR** *owner\_userid*

is the user ID of the *owner* specified in the DISABLE command (see usage notes).

#### **FUNCTION** *function\_name*

identifies the function that disabled the file space and its storage group. The FOR *owner\_userid* operand must not be used when this operand is used. The list of valid function names currently consists of one name, RENAME. This function is valid only with the ENABLE FILESPACE format of the command. When this function name is used, the specified file space and its associated storage group, which were disabled because an SFS rename user ID attempt using the FILEPOOL RENAME command, are unlocked.

This should be done only in extreme cases where FILEPOOL RENAME cannot finish and the lock needs to be removed in order to recover.

### Usage Notes

1. All messages are written to the server operator console.
2. The ENABLE command will unlock the storage group or file space allowing general access to the objects within them.
3. If FOR *owner\_userid* was specified in the DISABLE command (or CSL routine), it must be specified in the ENABLE command with the same user ID.

## ENABLE

4. If FOR *owner\_userid* was not specified in the DISABLE command (or CSL routine), it must not be specified in the ENABLE command.
5. Callable Services Library (CSL) routines are available in VMLIB that disable and enable storage groups and file spaces. They are:

**DMSDISFS** disables a file space

**DMSDISSG** disables a storage group

**DMSENAFS** enables a file space

**DMSENASG** enables a storage group

These routines can be called from application programs that process in an administration machine. See [z/VM: CMS Callable Services Reference](#) for more about these routines.

6. File pool administrator commands can be issued to disable and enable storage groups and file spaces. They are:

**FILEPOOL DISABLE FILESPACE** disables a file space

**FILEPOOL DISABLE GROUP** disables a storage group

**FILEPOOL ENABLE FILESPACE** enables a file space

**FILEPOOL ENABLE GROUP** enables a storage group

7. When the FILEPOOL RENAME command is used, the specified file space (and its associated storage group) is locked for the duration of the rename function. If the rename function is unsuccessful, the locks remain held by the rename function.

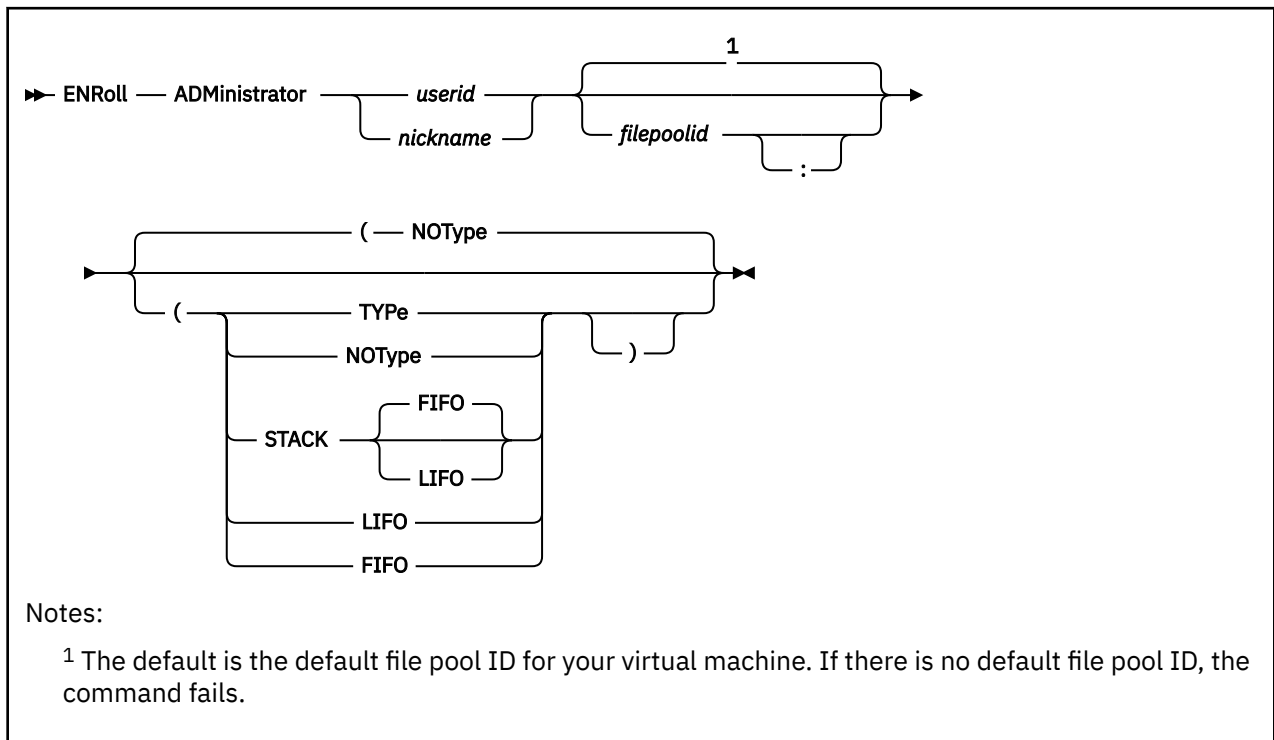
The locks can be released only by successful completion of the rename function or by specifying FUNCTION RENAME on the ENABLE operator command or FILEPOOL ENABLE administrator command. When this is the case, the existing (old) file space ID should be specified for *filepaceid*.

8. When dataspace are successfully created and a storage group is disabled and then enabled, the enable processing will cause an error if there is a FBA disk that is not allocated in 8-block increments and not aligned on a 4KB boundary. This might occur if the minidisk was replaced with a FBA non-aligned minidisk when the storage group was disabled. Message DMS3074E is issued on the server console for non-aligned minidisks.

Otherwise, if there were no dataspace created, when a storage group is enabled and a minidisk is detected that is not aligned properly, message DMS3074I is issued for each minidisk, and processing continues.



## ENROLL ADMINISTRATOR



### Authorization

File Pool Administrator

### Purpose

Use the ENROLL ADMINISTRATOR command to grant file pool administration authority for the specified file pool. You must have file pool administration authority to use this command.

File pool administration authority lets you use the commands described in this chapter which are identified as administrator commands. In addition, administration authority lets you to do anything to users' objects that the users themselves can do. For example, you can read from or write to any file in the file pool.

File pool administration authority using the ENROLL ADMINISTRATOR command is temporary, and lasts only until one of the following occurs:

- File pool server processing is stopped or caused an error
- A DELETE ADMINISTRATOR command is entered for the user ID
- A REVOKE ADMIN operator command is entered at the server machine console (or its secondary user console)

For permanent file pool administration authority, use the ADMIN statement in the DMSPARMS file.

### Operands

#### *userid*

indicates the user is to be enrolled.

#### *nickname*

identifies a nickname that represents a user ID or a list of user IDs. (Use the NAMES command to define nicknames.)

### *filepoolid*

#### **filepoolid:**

identifies the file pool in which the administrator is to be enrolled. If not specified, the default file pool ID for your virtual machine will be used.

## Options

### **NOType**

suppresses the display of user IDs to be enrolled. NOType is the default.

### **TYPe**

displays at the terminal the user IDs to be enrolled.

### **STACK FIFO**

### **STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. FIFO is the default.

### **LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

### **FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

## Usage Notes

1. A user does not have to be enrolled in the file pool (either explicitly by ENROLL USER or implicitly by ENROLL PUBLIC) to receive file pool administration authority. A user with file pool administration authority can connect to the file pool and exercise all privileges permitted by that authority.
2. If a user group is specified and the enrollment of any user causes an error, all enrollments are unsuccessful. Processing stops after the first error.
3. You can determine which user IDs are enrolled as administrators by entering the following command:

```
query enroll administrator for all vmsys
```

Substitute your own file pool ID for VMSYS.

A description of the QUERY ENROLL command is in *z/VM: CMS Commands and Utilities Reference*.

4. If the ENROLL ADMINISTRATOR command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
5. If you specify a nickname and the NODE tag in the NAMES file indicates the user is on another processor, you must also specify the LOCALID tag.

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in *z/VM: CMS and REXX/VM Messages and Codes* and *z/VM: CMS Commands and Utilities Reference*.

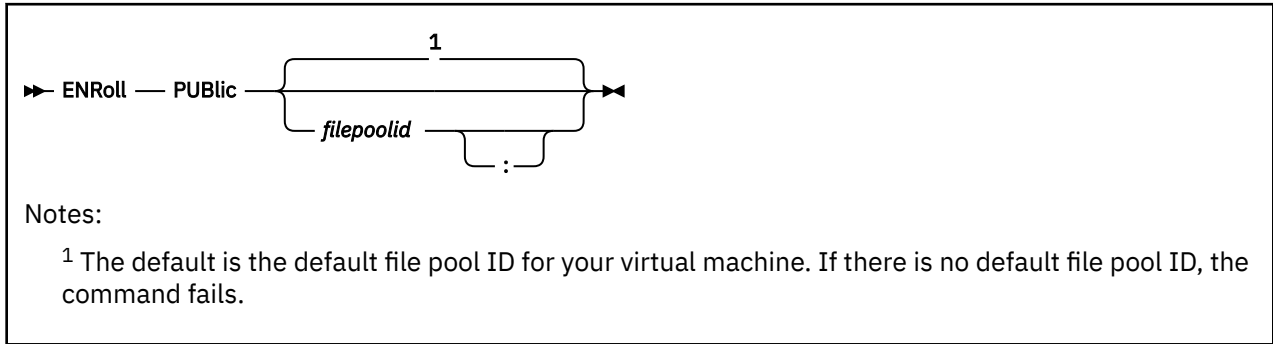
This command internally issues IDENTIFY, FINIS, NUCXDROP, and NUCXLOAD commands. Therefore, messages from those commands can also be displayed. (For more information about those commands, see *z/VM: CMS Commands and Utilities Reference*.)

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS066E *option1* and *option2* are conflicting options [RC=24]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS149E Userid *userid* not valid [RC=32]

- DMS637E Missing nodeid for the AT operand [RC=24]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=32]
- DMS653E Error executing *command rc=nn* [RC=40]
- DMS1165W One or more userids were already enrolled as ADMINISTRATORS [RC=4]
- DMS1201E STACK option cannot follow FIFO or LIFO [RC=24]

## ENROLL PUBLIC



### Authorization

Repository File Pool Administrator

### Purpose

Use the ENROLL PUBLIC command to give connect authority for a file pool to all users. File pool administration authority is required to use this command.

### Operands

*filepoolid*

***filepoolid:***

identifies the file pool in which PUBLIC is being enrolled. If not specified, the default file pool ID for your virtual machine will be used.

### Usage Notes

1. A user not explicitly enrolled in the file pool can connect to the file pool if PUBLIC is enrolled. Such users have no file space or top directory in the file pool. Users cannot create their own directory structure of files. They can read, write, and create locks on other users' files or directories if the other users have granted the proper authority either to PUBLIC or to that specific user ID. (Users can grant authorities to user IDs that are not enrolled in the file pool.)
2. ENROLL PUBLIC does not add any space for users. If a user is to be given space, he or she must be enrolled explicitly by the ENROLL USER command.
3. Entering an ENROLL PUBLIC command is not the same as entering an ENROLL USER command for *userid* PUBLIC. ENROLL PUBLIC lets all users connect to the file pool. ENROLL USER PUBLIC simply lets the *userid* PUBLIC connect to the file pool.

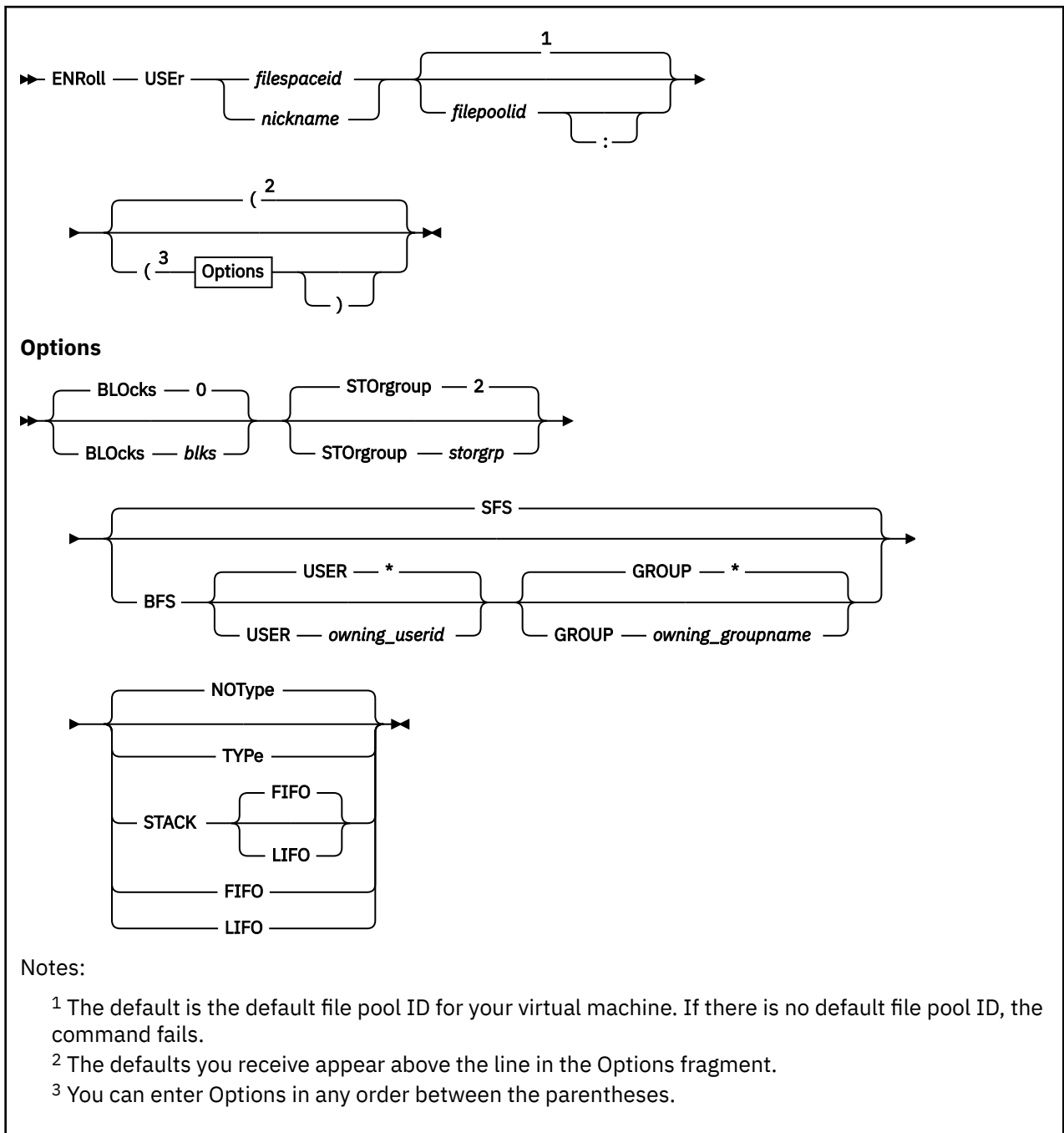
When ENROLL PUBLIC has been entered for a file pool, a subsequent QUERY ENROLL USER FOR ALL command shows <PUBLIC> in its output. If a user ID of PUBLIC is enrolled, *PUBLIC* (without the < and > characters) is displayed. If both ENROLL USER PUBLIC and ENROLL PUBLIC have been entered for the file pool, you see both *PUBLIC* and <PUBLIC> in the QUERY ENROLL output. The QUERY ENROLL command is described in [z/VM: CMS Commands and Utilities Reference](#).

4. If the ENROLL PUBLIC command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
5. If you restore a file pool by entering a FILESERV GENERATE command and restoring all user storage groups, the enrollment of PUBLIC is lost. If you want to retain the enrollment of PUBLIC, you must enter the ENROLL PUBLIC command again.
6. If you are using BFS file spaces you can have the equivalent of PUBLIC connect authority by having a non-default User ID (UID). See [z/VM: CP Planning and Administration](#) for more information.

## Messages and Return Codes

The only messages displayed by this command are the system messages listed in [z/VM: CMS and REXX/VM Messages and Codes](#).

## ENROLL USER



### Authorization

Repository File Pool Administrator

### Purpose

Use the ENROLL USER command to enroll a user or create a Byte File System (BFS) in a specific file pool. This command provides function equivalent to the CSL Routine DMSENUUSR - Enroll File Space, which is described in *z/VM: CMS Callable Services Reference*.

## Operands

### ***filepaceid***

identifies the file space to be enrolled. This is either a user ID to be enrolled or the name of the Byte File System (BFS) to be created. Do not enroll a user ID that begins with a plus (+) or a minus (-) or that contains a colon (:) or a period (.). These characters are used as separator characters in SFS directory IDs and the file space ID is a part of the SFS directory ID. Also, BFS file space IDs may not contain the slash (/) or X'00' characters.

### ***nickname***

identifies a nickname that represents a user ID, a BFS, or a list of user IDs. (Use the NAMES command to define nicknames.) You may not use a nickname representing a list if any of the file spaces in the list are Byte File System.

### ***filepoolid***

#### ***filepoolid:***

identifies the file pool in which the file space is to be created. If not specified, the default file pool ID for your virtual machine is used.

## Options

### **BLOCKS *blks***

is an optional parameter that indicates the number of 4096-byte file blocks that can be consumed by the file space.

When ENROLL USER is entered with the BLOCKS option, physical DASD space is not immediately reserved. Physical DASD space is used only as it is needed. The BLOCKS value is a *limit* to the number of blocks a user is allowed to consume. The BLOCKS value represents *committed* blocks, (that is, blocks permanently written to the file pool).

For SFS file spaces, there is no limit, other than the physical space available in the storage group, on the number of uncommitted blocks a user or application can consume.

For BFS file spaces, consumption greater than the BLOCKS value is not allowed.

The *blks* can range from 1 through 2,147,483,647. If the BLOCKS option is omitted, the file space is created with 0 blocks.

### **STOrgroup *storgrp***

is an optional parameter where *storgrp* specifies the storage group to which the user is assigned. All explicitly enrolled file pool users are assigned to some storage group even if they do not have space in the file pool. The *storgrp* can range from 2 through 32767, but must not be greater than the MAXDISKS value for the file pool. (MAXDISKS is set during file pool generation and can be changed by the FILESERV REGENERATE command.) The storage group specified must exist (that is, must have minidisks assigned to it), even if you are not allocating space in the file pool. If not specified, this parameter defaults to 2.

### **SFS**

specifies the file space to be created is an SFS file space. SFS is the default.

### **BFS**

specifies the file space is to be created is a Byte File System (BFS).

### **USER *owning\_userid***

identifies the VM user ID whose UID should become the owning UID of the top directory of the file space.

USERID \* indicates the owning UID or GID (or both) of the top directory of the file space should be the same as those associated with the VM user ID on which the command was entered.

### **GROUP *owning\_groupname***

identifies the group name whose GID should become the owning GID of the top directory of the file space.

## ENROLL USER

GROUP \* indicates the owning GID should be the same as that associated with the VM User ID on which the command was entered.

### NOType

suppresses the display of the file space IDs to be enrolled. NOType is the default.

### TYPe

displays at the terminal the file space IDs to be enrolled.

### STACK FIFO

### STACK LIFO

places the information in the console stack rather than displaying it at the terminal. FIFO is the default.

### LIFO

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

### FIFO

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

## Usage Notes

### Common Notes for SFS and BFS

1. Note that deadlocks can occur when file spaces are enrolled concurrently by different administrators, especially if the file space IDs are standardized (USER1, USER2, USER3, for example). In this case, one of the jobs is automatically rolled back. If this occurs, retry the command.
2. You can change the amount of space a file space is allowed to consume by entering the MODIFY USER command.
3. You can move the file space to a different storage group by using the FILEPOOL UNLOAD and FILEPOOL RELOAD commands. For instructions on how to do this, see [“FILEPOOL UNLOAD” on page 495](#) and [“FILEPOOL RELOAD” on page 476](#).
4. Use the QUERY ENROLL command to determine who is enrolled in a file pool. The QUERY ENROLL command is described in [z/VM: CMS Commands and Utilities Reference](#).
5. Entering an ENROLL USER command for a user ID of PUBLIC is not the same as entering an ENROLL PUBLIC command. ENROLL PUBLIC lets all user IDs connect to the file pool. ENROLL USER PUBLIC simply lets the user ID PUBLIC connect to the file pool.

When ENROLL PUBLIC has been entered for a file pool, a subsequent QUERY ENROLL USER FOR ALL command will show <PUBLIC> in its output. If a user ID of PUBLIC is enrolled, PUBLIC (without the < and > characters) will be displayed. If both ENROLL USER PUBLIC and ENROLL PUBLIC have been entered for the file pool, you will see both PUBLIC and <PUBLIC> in the QUERY ENROLL output.

Specifying either <PUBLIC> or \* as the user ID to be enrolled is not allowed. An error message will be issued.

6. If the ENROLL USER command is entered from an EXEC or ASSEMBLER program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
7. You can change the name of a file space using the FILEPOOL RENAME command.
8. Because DFSMS/VM uses file space IDs beginning with DFSMS you should avoid enrolling a file space ID that begins with DFSMS. Also if you do enroll such a file space, DFSMS/VM will not manage anything in that file space. See [z/VM: DFSMS/VM Storage Administration](#) for more information.
9. Enrolling a BFS file space means no SFS file space can be created for a user with that same name.

### Notes for SFS only

1. If a *nickname* that represents a list of user IDs was specified, and the enrollment is unsuccessful for one file space in the list, enrollment of all file spaces in the list will be unsuccessful.
2. When an ENROLL USER command is entered, the following applies for the specified user IDs regardless of whether the BLOCKS option is specified. The user:



- Can connect to the file pool.
  - Is given a top directory. The name of the top directory is the user ID.
  - Can exercise all privileges explicitly granted to PUBLIC.
  - Can exercise all privileges explicitly granted to the user ID.
3. If the BLOCKS option is omitted, the specified user ID is not given a file space in the file pool. The user is given a top directory, as always, and can create a directory structure and put aliases in it. But, the user cannot create base files in his or her own directories. If the user has WRITE or directory control write (DIRWRITE) authority to other users' directories, the user can create files in those directories (assuming those other users have space allocated to them).

Even if the user has no file space, he or she can still exercise all privileges explicitly granted to PUBLIC. For example, suppose user TOM is enrolled without space. User GARY grants WRITE authorization to his PC FORUM file to PUBLIC. TOM can connect to the file pool and create an alias to the PC FORUM file. Even though TOM has no space of his own, he can still write to the PC FORUM file.

Users without file space can exercise all privileges explicitly granted to the user ID. For example, user JACK might grant READ authority on one of his files to user IRA. Even if user IRA has no space of his own, he still has a top directory and can create an alias to JACK's file.

4. If the BLOCKS option is specified, the user is given a file space in which he or she can create base files. The user can authorize others to write to his or her directories and files, in which case others will be able to use blocks in that file space.
5. If you specify a nickname and the NODE tag in the NAMES file indicates the user is on another processor, you must also specify the LOCALID tag.
6. A warning threshold of 90 percent is established for a user when you enroll the user with space (the BLOCKS option is specified). When the number of file blocks a user consumes exceeds 90 percent of his or her total allocated space, CMS issues a warning message to the user. Others who are authorized to write to the file space may also receive the warning message.

Users may change their default threshold warning percent by using the SET THRESHOLD command or the administrator can change the threshold warning percent for any file space with the SET THRESHOLD command with the FOR *userid* or FOR *nickname* option.

7. Note you can enroll a user ID in a file pool even if no such user ID exists in the z/VM system directory. This lets you enroll a set of user IDs in a file pool before they are added to your z/VM system.

### Notes for BFS only

1. When a BFS file space is created, the owning UID is given read, write, and search permission for the top directory. Group and public permissions are not given. The owner of the file space or a BFS superuser can use the OPENVM PERMIT command to change permissions associated with the directory.
2. If the BLOCKS option is omitted, the Byte File System is created, but is not allocated any space in the file pool. Users can create BFS objects that do not require space. This excludes only non-empty regular files.
3. If the BLOCKS option is specified, a Byte File System is created in which users can create BFS regular files as well as other BFS objects.
4. Threshold values are not established or maintained for Byte File System file spaces.
5. You cannot enter the ENROLL USER command with the BFS option while in subset mode, DOS mode, or running on a level of CP which is earlier than VM/ESA Version 2 Release 1.
6. The user ID provided on the USER option and group name provided on the GROUP option are translated into a UID and a GID based on the values in the POSIX database on the system on which the command is issued. This database can be in the z/VM CP directory, or managed by an External Security Manager (ESM). When using these values, the administrator who is issuing the command should be on the same system as the file pool containing the new BFS or results are unpredictable.

7. To specify a user ID on the USER option, the requesting VM user ID must be authorized to obtain a user entry in the POSIX database. To specify a group name on the GROUP option, the requestor must be authorized to obtain a group entry in the database.

Typically, the requesting VM user ID has the attribute POSIXOPT QUERYDB ALLOW set, either through a statement in its CP directory entry or through a specified or defaulted setting in the system configuration file not overridden in the directory entry.

If that is not the case, one of the following must be true:

- The external security manager (ESM) grants the requestor the authority to read the entry.
- An ESM is either not installed or defers authorization to CP, and one of the following is true:
  - The requestor is a superuser (has an effective UID of 0).
  - The current real or effective UID or GID matches the UID or GID for the specified user or group.
  - For a GID, the requestor is a member of the specified group.

## Messages and Return Codes

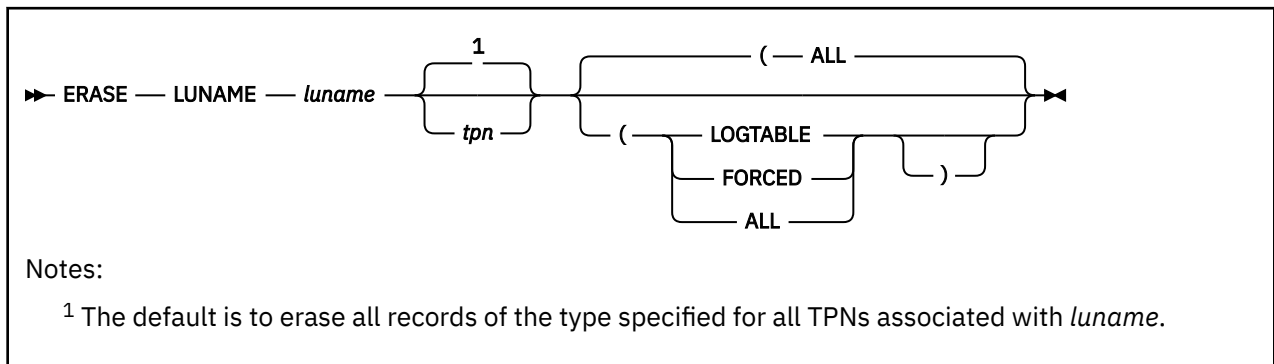
In addition to the messages listed below, this command displays other system messages. These system messages are listed in *z/VM: CMS and REXX/VM Messages and Codes*.

This command internally issues IDENTIFY, FINIS, NUCXDROP, and NUCXLOAD commands. Therefore, messages from those commands can also be displayed. (For more information about those commands, see *z/VM: CMS Commands and Utilities Reference*.)

Messages:

- DMS029E Incorrect parameter *parm* in the option *option* field [RC=24]
- DMS065E *option* option specified twice [RC=24]
- DMS066E *option1* and *option2* are conflicting options [RC=24]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect operand: *operand* [RC=24]
- DMS637E Missing node ID for the AT operand [RC=24]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=32]
- DMS653E Error executing *command rc=nn* [RC=40]
- DMS1166E Userid *filespaceid* is already enrolled [RC=40] {User ID *userid* | Group name *groupid*} cannot be resolved.
- DMS1175E Storage group does not exist [RC=40]
- DMS1175E Storage group does not exist. The file pool must be regenerated [RC=40]
- DMS1201E STACK option cannot follow FIFO or LIFO [RC=24]
- DMS1209E Nickname *nickname* resolved to more than one name; only one Byte File System can be created at a time [RC=40]
- DMS2023E File pool *filepoolid* does not support the requested option [RC=88]
- DMS2132E Error obtaining UID or GID [RC=104]
- DMS2132E Error obtaining UID or GID. User not authorized [RC=104]
- DMS2132E Error obtaining UID or GID. User not found [RC=104]
- DMS2132E Error obtaining UID or GID. Group not found [RC=104]
- DMS2132E Error obtaining UID or GID. Database not available [RC=104]
- DMS2132E Error obtaining UID or GID. Command not allowed in CMS/DOS environment, in CMS subset mode, or on this level of CP [RC=104]

## ERASE LUNAME



### Authorization

Repository File Pool Operator

### Purpose

Use the ERASE LUNAME SFS operator command to remove log name table entries, history of previously forced work, or both, for an LU name or LU name-transaction program name (TPN) pair.

**Note:** The TPN is sometimes called a resource ID.

### Operands

#### *luname*

indicates the fully qualified LU name of the CRR recovery server that consists of:

- Optional SNA network ID
- Period if the SNA network ID is supplied
- LU name.

If both the SFS file pool server and the CRR recovery server reside on the same processor, TSAF collection, or CF collection, you must specify \*LOCAL for the *luname*. If you specify \*LOCAL, you must also specify *tpn*.

#### *tpn*

is the TPN of the CRR recovery server obtained from the QUERY LOGTABLE SFS operator command.

### Options

#### **ALL**

erases both the log name table entries and history of forced work. ALL is the default.

#### **LOGTABLE**

erases only log name table entries.

#### **FORCED**

erases only history of forced work.

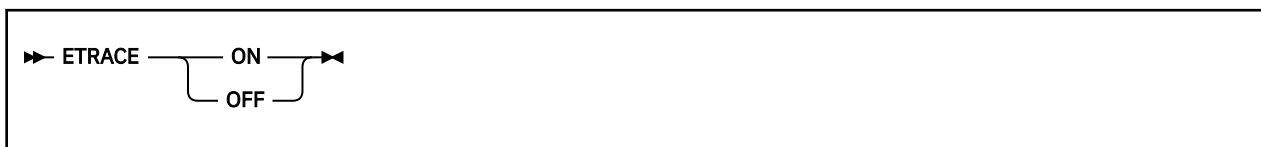
### Usage Notes

1. Use the QUERY LOGTABLE command to list log name table entries.
2. Use the QUERY PREPARED command to list history of forced work.
3. If just the LU name is specified, all records of the type specified for all TPNs associated with that LU name will be erased.

## ERASE LUNAME

4. If the optional TPN is supplied, only the records of the type specified for the LU name and TPN pair will be erased.
5. The SFS log name table entries cannot be erased if prepared work for that LU name or LU name and TPN pair exist. You must determine if the prepared work will be able to complete. If it cannot complete, you have to force the prepared work before erasing it from the file pool logs. Refer to the QUERY PREPARED and FORCE PREPARED commands for further information.
6. This command should be used with *extreme caution* because it erases the record of any heuristic force performed.
7. If CRR security audit tracing is enabled, the ERASE LUNAME command generates an audit trace record.

## ETTRACE



### Authorization

Repository File Pool Operator

### Purpose

Use the ETRACE operator command to start or stop server external trace processing.

### Operands

#### ON

starts server external trace processing. A series of prompts will be issued to allow the type and level of trace data to be specified. These prompts are described below under Usage Notes. The server virtual machine must be enabled for tracing, using the TRSOURCE facility before the ETRACE ON command is issued.

#### OFF

stops server trace processing.

**Note:** If you are using TRSOURCE with BLOCK mode specified, ETRACE OFF must be issued before disabling the TRSOURCE trace to ensure all the data was retrieved.

### Usage Notes

1. You should use the ETRACE facility only when requested by the designated support group for your installation.
2. All messages are written to the server operator console.
3. ETRACE ON starts server external trace processing. A series of prompts will be issued to allow the type and level of trace data to be specified. A '0' may be entered as a response to any prompt to cancel the ETRACE command. The first prompt you see is:

```
DMS3084R Enter one of: USERID, * (for all).
          Or reply 0 (Cancel) for cancel
```

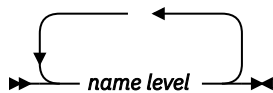
In response to this prompt, enter a user ID for which processing should be traced. A single user ID or \* (for all user IDs) can be specified. Do not specify a nickname. Nicknames are not recognized by server operator commands. The second prompt you see is:

```
DMS3087R Enter one or both of: DAC SAC.
          Or reply 0 (Cancel) for cancel
```

In response to this prompt, specify SAC or DAC to indicate the type of server processing that should be traced. (SAC and DAC are different sections of the server. See below for types of information in each section.)

The next prompt will ask for either DAC or SAC function names, depending on what you entered in response to the preceding message. The prompt will ask you to enter *trace level pairs*. A *trace level pair* is a function name followed by the level of tracing desired for that function. Your response should be in this format:

## ETTRACE



where:

**name**

identifies the function that should be traced. (Names are listed below.)

**level**

indicates the level of detail of the trace data. Tracing is done in limited detail if level 1 is specified. Detailed tracing is done if level 2 is specified. If 0 is specified, no tracing is done for the specified name.

The message you will see if *DAC* was entered in response to message DMS3087R is:

```
DMS3088R Enter DAC function name and trace level pairs
Valid function names are: * CA CT SS RQ SP ST RP PM WK
Valid function names are: RESYN CRLOG BRLM
Valid trace level values are: 0, 1, 2
Or reply 0 (Cancel) for cancel
```

The DAC function names that can be specified are:

**\***

All DAC functions

**CA**

Cache Management

**CT**

Catalog Management

**SS**

Session Management

**RQ**

Request Management

**SP**

Space Management

**ST**

Startup Management

**RP**

Response Management

**PM**

Pool Management

**WK**

Work Management

**RESYN**

CRR Resynchronization

**CRLOG**

CRR Log Management

**BRLM**

BFS Byte Range Lock Management

The message you will see if *SAC* was entered in response to message DMS3087R is:

```
DMS3090R Enter SAC function name and trace level pairs
Valid function names are: * ENTRY EXIT LOG LOCK LUW
Valid function names are: DC DM STOR INDEX FA WS
Valid trace level values are: 0, 1, 2
Or reply 0 (Cancel) for cancel
```

The SAC function names that can be specified are:

**\***

All SAC functions

**ENTRY**

SAC entry calls

**EXIT**

Returns from SAC entry calls

**LOG**

Log and Recovery Management

**LOCK**

Lock Management

**LUW**

Logical Unit of Work Management

**DC**

Data Control

**DM**

Data Manipulation

**STOR**

Storage Management

**INDEX**

SAC Index

**FA**

File Access

**WS**

Working Storage Management (both SAC and DAC usage)

Finally, you see these messages when ETRACE is successfully activated:

```
DMS1338I ETRACE set ON for DMSTRACE
DMS3095I ETRACE is now active
```

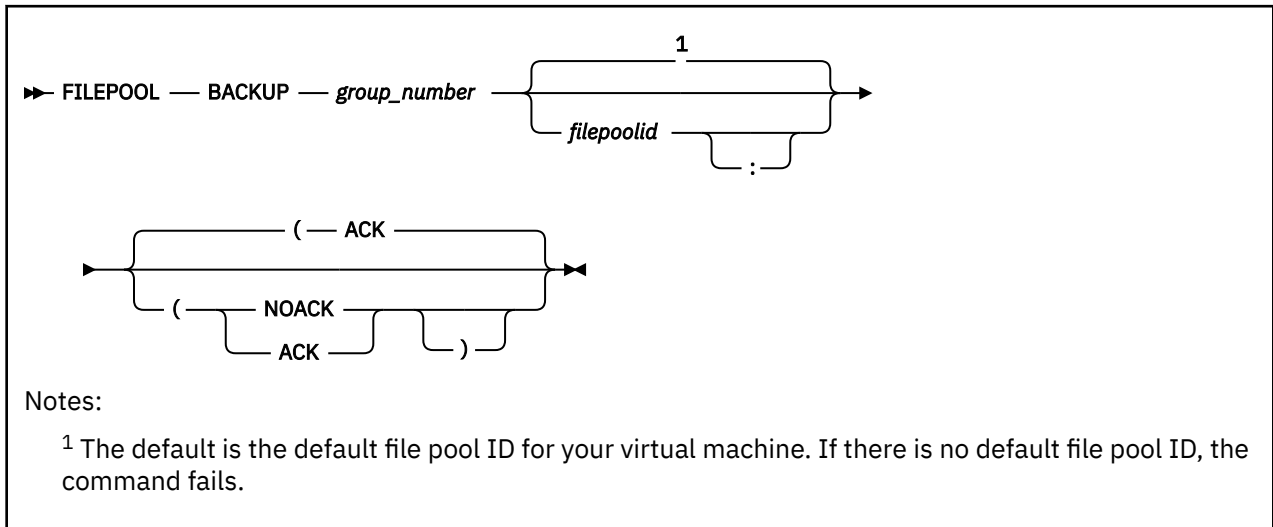
- Trace output is formatted and viewed in the following manner.

Trace output data will be written to a TRSOURCE system trace file and can be viewed using the CP TRACERED utility. The list of these files may be displayed by issuing *QUERY TRFILES*. For more information, see *z/VM: Dump Viewing Facility*. The virtual machine that issues the TRSOURCE command does not have to be the server machine. However, the TRSOURCE facility must be started before the ETRACE ON command is issued. The TRSOURCE command can be issued from any virtual machine having CP privilege class A, C, or E. (Note that most file poolserver machines do not have privilege class C.)

**Note:** TRSOURCE must be enabled in BLOCK mode (with GT BLOCK operand). For more information about TRSOURCE, see *z/VM: CP Commands and Utilities Reference*.

- ETTRACE related errors do not stop server processing. Messages identifying the error condition are written and server processing continues with ETRACE processing suppressed.
- You can also start external trace processing by using the startup parameter *ETTRACE*.
- This command is not to be confused with the CMS ETRACE command, which enables or disables the recording of external events. CMS ETRACE is described in *z/VM: CMS Commands and Utilities Reference*.

## FILEPOOL BACKUP



### Authorization

Repository File Pool Administrator

### Purpose

Use the FILEPOOL BACKUP command to back up all data in a user storage group and all associated file pool catalog data. The resultant backup file can be used as input to the FILEPOOL RESTORE and FILEPOOL FILELOAD commands should a data loss in the storage group later occur. This command can only be issued from a virtual machine that has file pool administration authority.

Only storage groups that contain user file data can be backed up by this command. FILEPOOL BACKUP cannot be used to back up storage group 1 because it contains catalog data.

### Operands

#### *group\_number*

is the number of the storage group to be backed up. The number must be from 2 through 32767 and the storage group must exist (that is, must have at least one minidisk allocated to it).

#### *filepoolid*

#### *filepoolid:*

is the ID of the file pool in which the storage group resides. If not specified, the default file pool ID for your virtual machine will be used.

### Options

#### ACK

causes FILEPOOL BACKUP to display an acknowledgement message at the start of each stage of the backup process.

#### NOACK

suppresses acknowledgement messages.

### Usage Notes

1. The FILEPOOL BACKUP command obtains a *share disable lock* for the storage group being backed up. A share disable lock lets users read from the storage group, but does not allow any updates. The FILEPOOL BACKUP command fails if any directory control directories in the storage group are



accessed read/write. (When directory control directories are accessed read/write, the FILEPOOL BACKUP command cannot obtain a *share disable lock*.)

2. A file space or storage group cannot be backed up if it is locked because of an incomplete attempt at renaming that file space.
3. This command requires the MAINT 193 disk be accessed.
4. The only valid device types for the input and output files (for example, when using FILEDEF) are tape and disk.
5. Prior to entering the FILEPOOL BACKUP command, you must enter a FILEDEF command to define the output file that is to contain the backup data. Specify BACKUP as the ddname. For example, to define a tape output file, you might enter:

```
filedef backup tap1 sl volid 111111
```

This defines a tape file at virtual address 181 using IBM standard labels. (If you use IBM standard labels, multivolume tape files are supported.) The tape volume ID is 111111. A storage group backup file should always be the first file on the tape.

An example of a FILEDEF command that defines a disk output file is:

```
filedef backup disk sg000003 sgbackup b
```

This defines a CMS disk file named *SG000003 SGBACKUP* on file mode B.

The output file is created with a variable-blocked record format having a record length of 29024 and a block size of 29028 (RECFM VB LRECL 29024 BLKSIZE 29028). The RECFM, LRECL, and BLKSIZE values are specified internally. If you specify your own RECFM, LRECL, or BLKSIZE values on the FILEDEF command, they will be ignored.

For tape devices, if necessary, specify the label options and the tape device specifications such as number of tracks and density. You can also use a LABELDEF command for a number of functions associated with tapes. For example, if the backup spans more than one tape, you can have FILEPOOL BACKUP prompt you with the next tape to enter. Use LABELDEF to tell the system which tapes will be used. Enter:

```
labeldef backup volid ?
```

and answer the prompts for the volume IDs of the backup tapes. Now FILEPOOL BACKUP can tell you which volume to mount. For more information on how to establish volume IDs, see the TAPE command in *z/VM: CMS Commands and Utilities Reference*. For more about the FILEDEF and LABELDEF commands, see *z/VM: CMS Commands and Utilities Reference* and *z/VM: CMS Application Development Guide for Assembler*. For more about tape handling, see *z/VM: CMS User's Guide*.

6. It is often a good idea, before you enter the FILEPOOL BACKUP command, to enter SET FILEWAIT ON.
 

If you do not enter a SET FILEWAIT ON command, the FILEPOOL BACKUP command stops if any other user is writing to the storage group when FILEPOOL BACKUP tries to lock it. When FILEWAIT is ON, the file pool server does not stop the FILEPOOL BACKUP command if it cannot immediately obtain the share disable lock. Instead, the file pool server does not let any new writing activity start. It waits for all writing activity to end and then gets the share disable lock for FILEPOOL BACKUP. FILEPOOL BACKUP will then back up the storage group. When the backup is finished, the server automatically releases the share disable lock.
7. The FILEPOOL BACKUP command creates entries in the LASTING GLOBALV file. Therefore, the virtual machine in which the command processes must have a file mode A accessed in read/write mode. Either a minidisk or a directory in another file pool can be accessed as A. If a read/write file mode A is not available, the FILEPOOL BACKUP will still process but recovery from catastrophic errors during the processing of the command will be more difficult or disruptive. A *catastrophic error* of the FILEPOOL BACKUP command is any error that causes the usual stopping logic to be avoided.

8. The FILEPOOL BACKUP command uses an application named DMS5PR to connect to the \*BLOCKIO facility using IUCV.
9. When you enter this command from your virtual machine, there can be no active IUCV or APPC sessions that were initiated using only CP IUCV or APPC interfaces. Any such non-CMS connections to IUCV will result in error message DMS3545E being issued. SFS usage of APPC and IUCV cannot cause this problem.
10. If an external security manager is being used to provide data security, the virtual machine executing the FILEPOOL BACKUP command must be authorized to read all minidisks allocated to the storage group being backed up.
11. If an external security manager is not being used, you should be prepared to supply the read password for all minidisks in the storage group. FILEPOOL BACKUP processing prompts you for these passwords before it begins backing up the data.

Instead of responding to the prompts as they occur, you can write an exec in which the passwords are placed in the program stack. Stack the passwords in order of the minidisk numbers (MDKnnnnn) one per line. Then enter the FILEPOOL BACKUP command. (The FILEPOOL BACKUP command is implemented as an EXEC.) However, placing minidisk passwords in a CMS file does reduce the level of security of the minidisks.

If the number of entries on the program stack is equal to the number of minidisks in the storage group to be backed up, FILEPOOL BACKUP processing assumes the entries are passwords and tries to use them (the entries are destroyed). This implies the stack should not be used when running the FILEPOOL BACKUP command.

If a password in the program stack is incorrect, the following message is displayed:

```
DMS3541R  Incorrect password supplied for minidisk MDKnnnnn
          at vdev.
          Enter '1' to retry or '0' to cancel.
```

You should respond '0' to cancel the operation. Otherwise, FILEPOOL BACKUP processing will use the next password on the stack, which is likely to be incorrect as well. Depending on your installation, repeated unsuccessful attempts to use a minidisk with incorrect passwords may cause the minidisk to be locked.

12. If the number of entries on the program stack before entering the FILEPOOL BACKUP command is equal to the number of minidisks in the storage group being backed up, the stack is destroyed.
13. If file pool server processing fails while FILEPOOL BACKUP is executing, you will be given the option of waiting until the file pool server is restarted or canceling the command. If you choose to cancel, the storage group will be unavailable for write access when the server is restarted. To make the storage group available for usual multiple user mode access, you have two options:
  - a. After the server is restarted, enter the FILEPOOL BACKUP command again.
  - b. If, for some reason, you do not want to restart the command, enter a FILEPOOL CLEANUP command for the storage group. FILEPOOL CLEANUP will reenables the storage group for update activity.
14. A catastrophic error of the FILEPOOL BACKUP command is any error that causes the usual stopping logic to be avoided. A catastrophic error occurred if you do not see a usual or unsuccessful completion message from the command.

If a catastrophic error has occurred, certain resources may be in an unusable state. If for some reason the backup cannot be immediately rerun, the FILEPOOL CLEANUP command should be used to release as many of these resources as possible.

Until the backup is rerun or CLEANUP done, the following conditions will exist:

- The storage group will be disabled. Users of the storage group will not be able to modify it, but they will be able to read it.
- The minidisks may be linked in READ mode until the session ends.

If the virtual machine was in DOS SVC mode at the time the FILEPOOL BACKUP command was called and a catastrophic error occurs, it is possible the machine will be left on OS SVC mode. Similarly, if CP TRACE I/O was in effect at the time of the catastrophic error, CP TRACE I/O may no longer be in effect. FILEPOOL CLEANUP processing will not reset either of the above items. Depending how DOS SVC mode or CP TRACE I/O was established in the virtual machine, you will need to either re-IPL or log off and log on again.

#### 15. Recovery from Catastrophic Errors without GLOBALV.

If, for any reason, the FILEPOOL BACKUP command failed in its attempt to use CMS GLOBALV to record variables in the execution that failed, you should:

- a. Detach all links to minidisks belonging to the storage group. You can simply log off the virtual machine in which the command was executing. (You can log on again at any time.) Or, you can enter CP DETACH commands for the minidisks.
  - b. Reenable the storage group entering the ENABLE operator command for the storage group. Specify the FOR *owner* operand, substituting the user ID under which the FILEPOOL BACKUP command was operating for *owner*.
16. FILEPOOL BACKUP processing issues DETACH commands internally. If you have a CP TRACE I/O command in effect, FILEPOOL BACKUP processing will temporarily turn it off during a DETACH.
- Because there is no way for FILEPOOL BACKUP processing to determine the options you originally specified on the TRACE I/O command, it reinstates the trace without any options. This causes the virtual machine to stop execution and enter CP command mode when the next I/O interrupt occurs. At this point you can reenter the TRACE I/O command with the correct options.
17. The SFS server internally manages the file pool space as though it were a linear space of physical blocks. These blocks are numbered relative to the beginning of the file pool. Because the FILEPOOL BACKUP process is sensitive to the block numbers, the configuration of the server must be preserved if the restore process is to be successful. In other words, when adding minidisk space or replacing minidisks in the file pool, you must strictly follow the procedures described in [Chapter 9, "Managing Storage," on page 195](#). Otherwise, the backup file may no longer be valid.
18. Acknowledgement Messages.
- Unless you specify the NOACK option, FILEPOOL BACKUP processing issues acknowledgement messages at the start of each stage of the backup process.
19. The FILEPOOL BACKUP command is not supported for execution in the CMS batch facility.
20. If you do not have DFSMS/VM installed on your system, you will receive messages DMS1136E and DMS3635W during FILEPOOL BACKUP processing. You should ignore these messages because they apply only to DFSMS/VM file pools.
21. If the file pool is managed by DFSMS/VM, it may contain migrated files. The FILEPOOL BACKUP command automatically backs up both the primary and the migrated data for the storage group being backed up.

When backing up a storage group which has migrated files in it, the following should be kept in mind:

- a. The virtual machine in which the FILEPOOL BACKUP command processes must have the following authorizations:
  - SFS administration authority for the file pool that contains the storage group being backed up
  - SFS administration authority for the file pool that contains the Migration Level 1 (ML1) directory for the storage group being backed up
  - Read authority to the DFSMS/VM control file DGTVCNTL DATA in directory VMSYS:DFSMS.CONTROL
- b. While the FILEPOOL BACKUP command is executing, the file pool with the migration level 1 directory for the storage group being backed up and the VMSYS file pool must also be available.
- c. The FILEPOOL BACKUP command requires certain CSL routines which reside in the DFSMS/VM-supplied library FSMINT CSLLIB. Therefore, an access to FSMINT CSLLIB is needed if you have

migrated data. This can be accomplished by linking and accessing the disk that contains the DFSMS/VM product code.

- d. If you are using the Migration Level 2 capabilities of DFSMS/VM Function Level 221 (ML2 is defined in the DFSMS/VM control file), IBM Language Environment® with the C Language option is required. The disks containing Language Environment must be accessed prior to invoking the FILEPOOL BACKUP command. If only ML1 is defined in the DFSMS/VM control file, Language Environment is not necessary.
- e. While backing up the migrated data, the FILEPOOL BACKUP command may display some messages that are issued by DFSMS/VM. Such messages are prefix with the characters **FSM** and are documented in [z/VM: DFSMS/VM Messages and Codes](#).
- f. It is not recommended that the migration level 1 directory be used to store data other than DFSMS/VM data. Non-DFSMS/VM data may be lost during FILEPOOL RESTORE processing.

For detailed information about DFSMS/VM, migrated files and managing storage groups, see [z/VM: DFSMS/VM Storage Administration](#) and [z/VM: DFSMS/VM Planning Guide](#).

22. The FILEPOOL LIST BACKUP command may be issued to determinethe contents of a backup file created by the FILEPOOL BACKUP command. For details on the command, see [“FILEPOOL LIST BACKUP”](#) on page 459.
23. If message DMS3202W has been received for storage group 1 and the situation has not been alleviated, FILEPOOL BACKUP processing may complete successfully, but FILEPOOL RESTORE processing may fail with a SFS server abend and message DMS3201E.

### Messages and Return Codes

#### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warning(s) issued.

**8 or greater**

Unusual ending. The return code is the same as the message number of associated error message.

When there is an error condition, the FILEPOOL BACKUP command, in many cases, displays more than one error message. Generally, the first error message displayed describes the immediate cause of the error. This message is often followed by additional error messages that further describe the error. The return code for this command is the message number.

## FILEPOOL CLEANUP

►► FILEPOOL — CLEANUP — *group\_number* — { *filepoolid* }<sup>1</sup>

Notes:

<sup>1</sup> The default is the default file pool ID for your virtual machine. If there is no default file pool ID, the command fails.

### Authorization

Repository File Pool Administrator

### Purpose

Use the FILEPOOL CLEANUP command to correct some of the storage group or administration machine problems caused by a catastrophic error of a FILEPOOL BACKUP or FILEPOOL RESTORE command. A catastrophic error of the FILEPOOL BACKUP or FILEPOOL RESTORE command is any error that causes the stopping logic to be avoided. This command requires administrator authority.

FILEPOOL CLEANUP drops all links to the file pool minidisks and enables storage groups disabled by FILEPOOL BACKUP or FILEPOOL RESTORE at the time of the error. FILEPOOL CLEANUP re-enables the storage group only if it is appropriate to do so.

### Operands

#### *group\_number*

is the number of the storage group to be corrected.

#### *filepoolid*

#### *filepoolid:*

is the ID of the file pool in which the storage group resides. If omitted, FILEPOOL CLEANUP uses the default file pool ID for the virtual machine. If omitted and there is no default file pool for the virtual machine, the command fails.

### Usage Notes

1. You must run the FILEPOOL CLEANUP command in the same virtual machine in which the FILEPOOL BACKUP or FILEPOOL RESTORE command failed.
2. You should use this command if FILEPOOL BACKUP or FILEPOOL RESTORE is not successful and you are unable to repeat the operation.
3. This command requires the MAINT 193 disk be accessed.
4. If you enter this command when it is not needed, a warning message is displayed and the command ends.
5. FILEPOOL CLEANUP fails if you did not have file mode A accessed in read/write mode during the running of the FILEPOOL BACKUP or FILEPOOL RESTORE command.

### Messages and Return Codes

#### Return Codes:

## FILEPOOL CLEANUP

**0**

Successful completion.

**4**

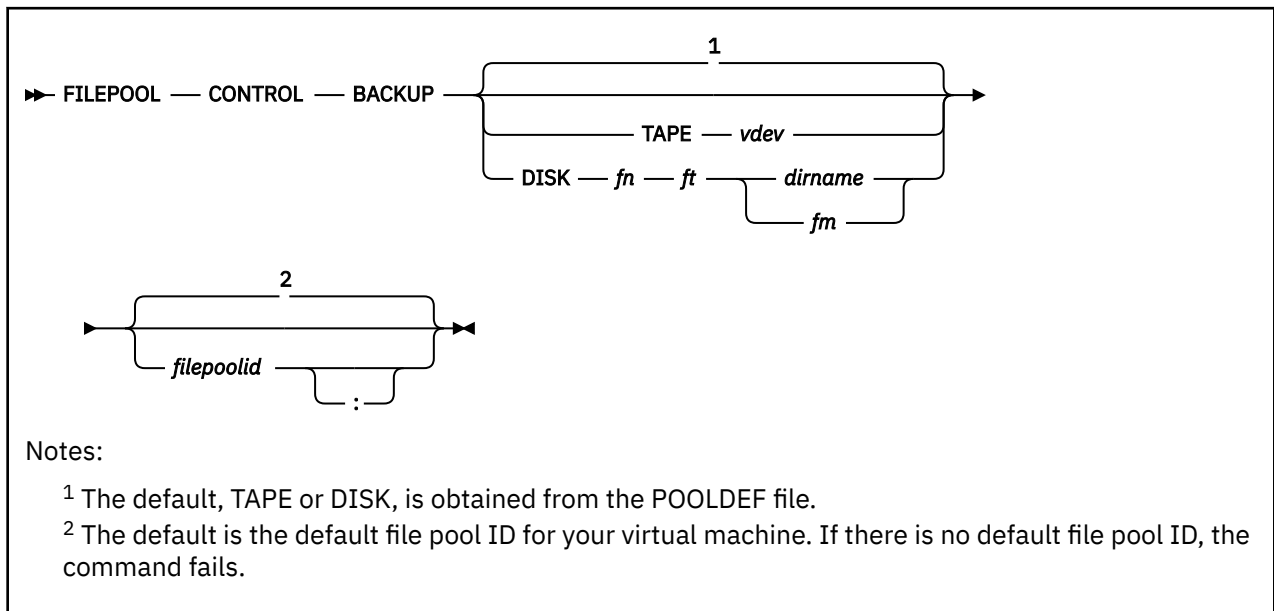
Successful completion, warning(s) issued.

**8 or greater**

Unusual ending. The return code number is the same as the number of the associated error message.

When there is an error condition, the FILEPOOL CLEANUP command, in many cases, displays more than one error message. Generally, the first error message displayed describes the immediate cause of the error. This message is often followed by additional error messages that further describe the error. The return code for this command is the message number.

## FILEPOOL CONTROL BACKUP



### Authorization

Repository File Pool Administrator

### Purpose

The `FILEPOOL CONTROL BACKUP` administrator command creates a backup of the control data. The control data includes the contents of the `POOLDEF` file, the control minidisk, and the catalog storage group (storage group 1). The resultant backup file may be used as input to a `FILESERV START` with the `RESTORE` startup parameter specified, should control data be lost at a later time. Note that this command may only be issued from a virtual machine that has file pool administration authority.

### Operands

#### TAPE *vdev*

identifies the virtual device number *vdev* for a standard label (SL) tape file. The control data backup will be directed to the identified tape device which must be attached to the file pool server. If an assignment for the control data backup file already exists, it is temporarily replaced.

#### DISK *fn ft dirname*

#### DISK *fn ft fm*

identifies a CMS file to which the control data backup will be directed. The file can reside on a minidisk or in another file pool. The *fm* represents one of the following:

- The file mode of a minidisk accessed by the file pool server whose control data is being backed up.
- The file mode of an SFS directory accessed by the file pool server whose control data is being backed up.

The *dirname* represents the fully qualified directory name of an SFS directory where the backup file is to be created. Note that *filepoolid:userid.n* (where you can repeat *n* up to 8 times, each one separated by a period) is the only allowed form of *dirname* in this case. If an assignment for the control data backup file already exists in the SFS file pool server, it is temporarily replaced.

### *filepoolid*

#### ***filepoolid:***

is the ID of the file pool in which the control data to be backed up resides. If you do not specify a file pool ID, the default file pool ID for your virtual machine is used.

## Usage Notes

1. For a list of supported virtual device numbers for tapes, see the description of the FILEDEF command in *z/VM: CMS Commands and Utilities Reference*.
2. If tapes are not mounted beforehand and TAPE is specified, message 3900E is returned in the \$\$\$SFS \$MSG\$ file.
3. This command requires the MAINT 193 disk be accessed.
4. A FILEPOOL CONTROL BACKUP command should be used to start a control data backup at a time when the workload for the SFS file pool server is at a low point.
5. The FILEPOOL CONTROL BACKUP command causes a control data backup to be started by the file pool server. The command will return to the issuer before the backup of the control data completes. Any error conditions found in the file pool server because the incorrect definition of the backup file will cause messages to be issued to the terminal where the command was entered.

A reader file, \$\$\$SFS \$MSG\$, will be sent to the administrator who issued the FILEPOOL CONTROL BACKUP command. There is a possibility you may not get this file because of spool space problems or the network not being available. If you wait for the file longer than the time it normally takes for control data to be backed up, you should take a look at the server console. Error messages after the backup is started, because of errors in writing the backup file, will appear in the \$\$\$SFS \$MSG\$ file and on the SFS operator console. The \$\$\$SFS \$MSG\$ file will also contain messages which will tell you whether the backup was successful, where it was directed, the cause of the error (if any) and where the last successful backup was directed.

6. SFS file pool server processing creates the control data backup file either at the default destination currently in effect, or at the destination explicitly specified on the FILEPOOL CONTROL BACKUP command itself.

The default destination for the backup file is originally determined from the contents of the POOLDEF file.

7. Explicitly redirecting the backup file specifying a new destination on the FILEPOOL CONTROL BACKUP command does not change the current default destination.
8. The BACKUP startup parameter must be in effect for FILEPOOL CONTROL BACKUP to complete successfully.
9. When the backup file resides on a minidisk, the control data backup processing uses a temporary file in the file pool server machine. This file is named \$\$TEMP \$BACKUP. You will only see the \$\$TEMP \$BACKUP file if the server crashed during the backup. If backup is successfully created, the file is erased and the current backup copy appears untouched. If backup processing fails, and the backup file resides on a minidisk, follow the recovery actions for backup error under [“Special Considerations for Control Data Backups to CMS File”](#) on page 106.
10. If the command fails to complete successfully, it can be rerun. If the command is unsuccessful and displays error messages, correct the problem and process the command again.
11. When the file pool server has written the control data to the backup file, it tries to close the backup file. If that file resides in another file pool and there is insufficient DASD storage in the target storage group or an insufficient number of unused blocks in the target file space, the backup stops.  

You can minimize the chance of having the backup stop by ensuring there is adequate physical storage in the storage group and an adequate number of unused blocks in the file space. You should also consider restricting the write activity to the file space, so other users do not consume the blocks intended to hold the control data backup.
12. When the backup file is created in another file pool, the performance degradation to the users of the file pool being backed up is less than it would have been if the files were created on tape or minidisk.



However, the backup process itself takes longer to complete. For more information, see [“Special Considerations for Control Data Backups to File Pool”](#) on page 105.

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS and REXX/VM Messages and Codes](#).

### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warning(s) issued.

**8 or greater**

Unusual ending. The return code is the same as the message number of associated message.

When there is an error condition, the FILEPOOL CONTROL BACKUP command, in many cases, displays more than one error message. Generally, the first error message displayed describes the immediate cause of the error. This message is often followed by additional error messages that further describe the error. The return code for this command is the message number.

Messages:

- DMS1223E There is no default file pool currently defined [RC=1223]
- DMS3221E File definition specified for BACKUP already in use. File pool=*filepoolid* [RC=3221]
- DMS3287E FILEPOOL CONTROL BACKUP command rejected because backup is being performed. File pool=*filepoolid* [RC=3287]
- DMS3288E FILEPOOL CONTROL BACKUP command rejected because NOBACKUP is in effect. File pool=*filepoolid* [RC=3288]
- DMS3289E FILEPOOL CONTROL BACKUP command rejected because the control data backup file definition is currently being changed. File pool=*filepoolid* [RC=3289]
- DMS3296I A FILEPOOL CONTROL BACKUP has been successfully scheduled. File pool=*filepoolid* [RC=0]
- DMS3297E Error encountered while processing the FILEPOOL CONTROL BACKUP command [RC=3297]
- DMS3298E File pool *filepoolid* does not support the *operand* operand on the FILEPOOL command [RC=3298]
- DMS3514E Action *action* incorrect (RC=3514)
- DMS3518E File pool *filepoolid* is unavailable or unknown [RC=3518]
- DMS3532E This userid does not have administrator authority for file pool *filepoolid* [RC=3532]
- DMS3600E Control data backup not defined. File pool=*filepoolid* [RC=3600]
- DMS3601E Invalid device address *vdev* for tape backup. File pool=*filepoolid* [RC=3601]
- DMS3602E Error issuing FILEDEF BACKUP for tape *tape*. File pool=*filepoolid*, Return code=*code* [RC=3602]
- DMS3602E Error issuing FILEDEF BACKUP DISK *fn ft fm*. File pool=*filepoolid*, Return code=*code* [RC=3602]
- DMS3602E Error issuing FILEDEF TBACKUP DISK \$TEMP \$BACKUP *fm*. File pool=*filepoolid*, Return code=*code* [RC=3602]
- DMS3603E File mode *fm* is incorrect for disk backup. File pool=*filepoolid* [RC=3603]
- DMS3604E No minidisk or directory is accessed as mode *fm*. BACKUP processing will fail. File pool=*filepoolid* [RC=3604]
- DMS3605E Error in CSL routine DMSQFMOD. Filemode=*fm*, return code=*code*, reason code=*code*, File pool=*filepoolid* [RC=3605]

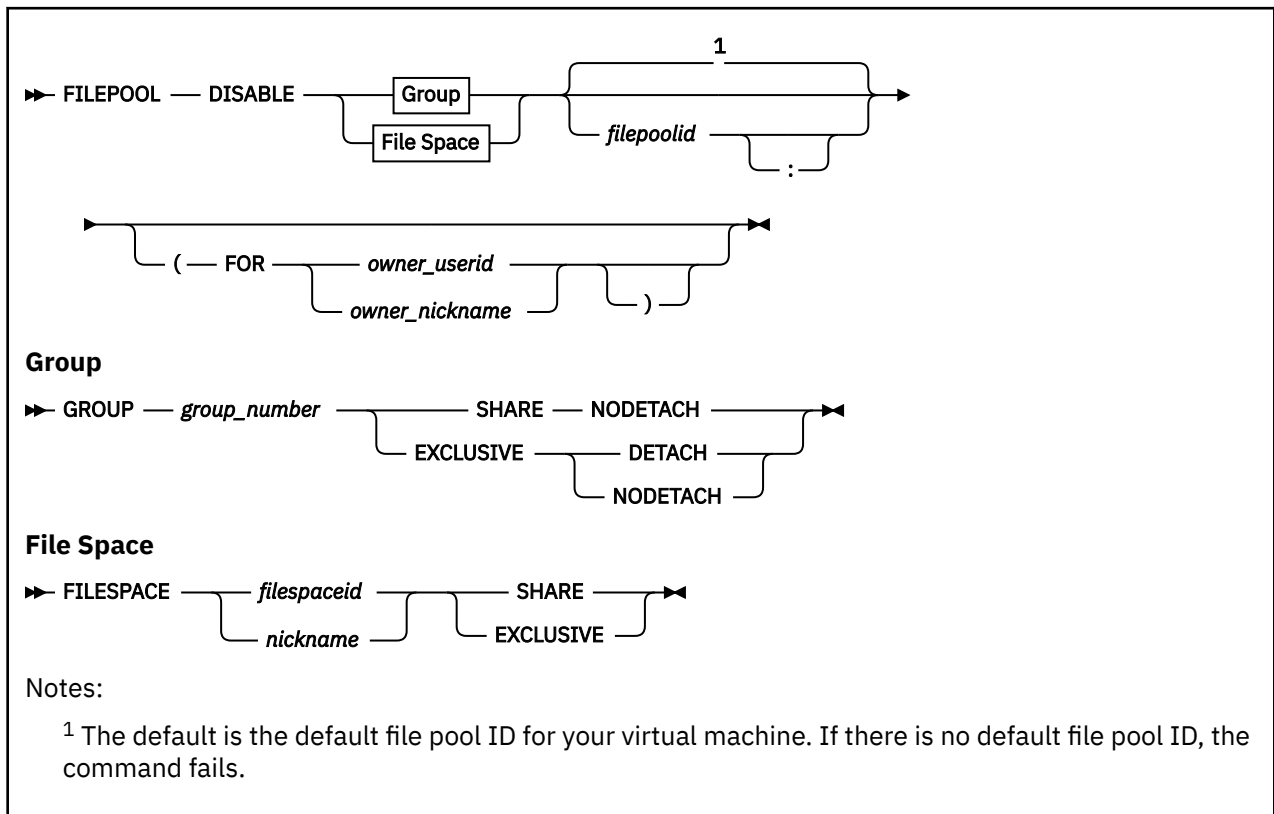
## FILEPOOL CONTROL BACKUP

- DMS3606E File mode *fm* not accessed read/write by server *serverid* for disk backup. File pool=*filepoolid* [RC=3606]
- DMS3607E Directory *dirname* does not exist in file pool *filepoolid*, or server *serverid* not authorized for it. File pool=*filepoolid* [RC=3607]
- DMS3608E Directory *dirid* is incorrect or is not fully qualified. File pool=*filepoolid* [RC=3608]
- DMS3609E File pool *filepoolid* is unavailable or unknown. File pool=*filepoolid* [RC=3609]
- DMS3610E Error in CSL routine DMSEXIDI. Directory=*dirid*, return code=*code*, reason code=*code*, file pool=*filepoolid* [RC=3610]
- DMS3611E Server *serverid* not authorized to create a file in directory *dirid*. File pool=*filepoolid* [RC=3611]
- DMS3612E File pool *filepoolid* cannot be used for backup of file pool. File pool=*filepoolid* [RC=3612]
- DMS3621E Error on *functionname*. Reason code = *code* [RC=3612]

The following messages can be returned in the \$\$\$SFS \$MSG\$ file.

- DMS1141W User filespace threshold [still] exceeded [for file pool *filepoolid*]
- DMS1156S Supervisor error {1;2;} return code *retcode* [,reason code *reascode*]
- DMS1259E File pool *filepoolid* has run out of physical space in the storage group
- DMS3207E Can not lock file *fn ft* in *dirid*
- DMS3209E *Rtname* request failed. Return Code= *code1* Reason code=*code2*
- DMS3222E Tape specified for BACKUP contains current control data backup file
- DMS3231E Backup scheduled for file pool *filepoolid* by *command* command was unsuccessful
- DMS3231I Backup scheduled for file pool *filepoolid* by *command* command was successful Reason code *code*
- DMS3516E No workunits are currently available
- DMS3609E File pool *filepoolid* is unavailable or unknown
- DMS3613I [Last successful] control data backup file was directed to tape device *tape*
- DMS3614I [Last successful] control data backup file was directed to minidisk. File name:*filename* File type:*filetype* File mode:*filemode*
- DMS3615I [Last successful] control data backup file was directed to file pool *filepoolid*. Filename:*filename* File type:*filetype* File mode:*dirid*
- DMS3900E {Open|Close|Write|Read} error DDNAME = *ddname* REASON1=*reason1* REASON2=*reason2*

## FILEPOOL DISABLE



### Authorization

Repository File Pool Administrator

### Purpose

Use the FILEPOOL DISABLE administrator command to lock a storage group or file space for write access (SHARE) or all access (EXCLUSIVE). When all access is prohibited (EXCLUSIVE), the storage group minidisks can also be detached. The disable may optionally be done on behalf of another user by specifying the user ID. This command provides function equivalent to the associated CSL Routines DMSDISFS - Disable File Space and DMSDISSG - Disable Storage Group which are described in [z/VM: CMS Callable Services Reference](#).

### Operands

#### GROUP *group\_number*

indicates the number of a storage group to be disabled. *group\_number* can be any number from 2 to the MAXDISKS value specified for the file pool. The storage group specified must, however, exist (that is, it must have a minidisk assigned to it).

For more information, see [MAXDISKS](#).

#### FILESPACE *filespaceid*

#### FILESPACE *nickname*

identifies the file space to be disabled. *Nickname* may be a nickname for a single user ID or Byte File System (BFS), but it cannot be a nickname for a list of user IDs or Byte File Systems. (Use the NAMES command to define nicknames.)

### SHARE

indicates a *share disable lock* should be acquired on the file space or storage group. A share disable lock allows users to read, but not modify, items in the locked object. Files also cannot be put into migrated status, recalled, or erased by any user.

### EXCLUSIVE

indicates an *exclusive disable lock* should be acquired on the file space or storage group. An exclusive disable lock prevents anyone other than the owner of the lock from reading or modifying the items in the locked object. Files also cannot be put into migrated status, recalled, or erased by other users.

### DETACH

this operand causes the file pool server to detach the minidisks associated with a storage group being disabled. This parameter is valid only when a storage group is being disabled and the EXCLUSIVE operand is specified. The minidisks are re-attached when the storage group is enabled.

### NODETACH

causes the file pool server to keep the links to the storage group minidisks. This operand is valid only when a storage group is being disabled.

### *filepoolid*

#### ***filepoolid:***

identifies the file pool in which the file space or storage group being disabled exists. for your virtual machine is used. If there is no default file pool ID, the command is unsuccessful.

## Options

### **FOR *owner\_userid***

### **FOR *owner\_nickname***

acquires the lock on behalf of another user. A nickname must be a nickname for a single user ID, not a list of user IDs.

If you disable a storage group for another user, the user must have file pool administration authority.

If you disable a file space on behalf of a user other than the owner, the user must have file pool administration authority.

## Usage Notes

1. The lock acquired by the FILEPOOL DISABLE command persists until a FILEPOOL ENABLE administrator command or ENABLE operator command, or DMSENAFS or DMSENASG CSL routine is processed to enable the storage group or file space.
2. The FOR *owner\_userid* option identifies the user ID that owns the lock. If omitted, the lock is acquired on behalf of the administrator. There are several ways to allow users to access the locked object:
  - Enter the FILEPOOL ENABLE administrator command.
  - Enter the ENABLE operator command (with FOR *owner\_userid*).
  - An administrator executes an Enable CSL routine for the object and specifies the proper user ID as the owner of the lock.
  - The lock owner executes an Enable CSL routine for the object.
3. This command requires the MAINT 193 disk be accessed.
4. This command can be unsuccessful because:
  - A lock needed to complete the command is held by some user
  - The file space or storage group contains a directory control directory that is accessed read/write by another user.

Use the QUERY ACCESSORS command to determine who has the directory control directory accessed.

If the command is unsuccessful for either of these reasons, enter the command again at a later time.

5. Callable Services Library (CSL) routines are available in VMLIB that disable and enable storage groups and file spaces. They are:
- DMSDISFS** disables a file space
  - DMSDISSG** disables a storage group
  - DMSENAFS** enables a file space
  - DMSENASG** enables a storage group
- These routines can be called from application programs. See [z/VM: CMS Callable Services Reference](#) for more information about these routines.
6. File pool operator commands can be entered on the file pool server operator console to disable and enable storage groups and file spaces. They are:
- FILEPOOL DISABLE FILESPACE** disables a file space
  - FILEPOOL DISABLE GROUP** disables a storage group
  - FILEPOOL ENABLE FILESPACE** enables a file space
  - FILEPOOL ENABLE GROUP** enables a storage group
7. The FILEPOOL DISABLE command is unsuccessful if entered from an exec or assembler program and the file pool specified is active on the current default work unit (that is, has some work on it which has not been committed or rolled back).
8. If the file space is disabled SHARE or EXCLUSIVE, a disable storage group can be done for SHARE or EXCLUSIVE. This will override any file space disable locks. If the storage group is disabled SHARE, a disable file space can be done for SHARE. If there is an exclusive lock on a storage group, only the owner of that lock can put a lock on any file space within that storage group.

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS and REXX/VM Messages and Codes](#).

### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warning(s) issued.

**8 or greater**

Unusual ending. The return code is the same as the message number of associated message.

When there is an error condition, the FILEPOOL DISABLE command, in many cases, displays more than one error message. Generally, the first error message displayed describes the immediate cause of the error. This message is often followed by additional error messages that further describe the error. The return code for this command is the message number.

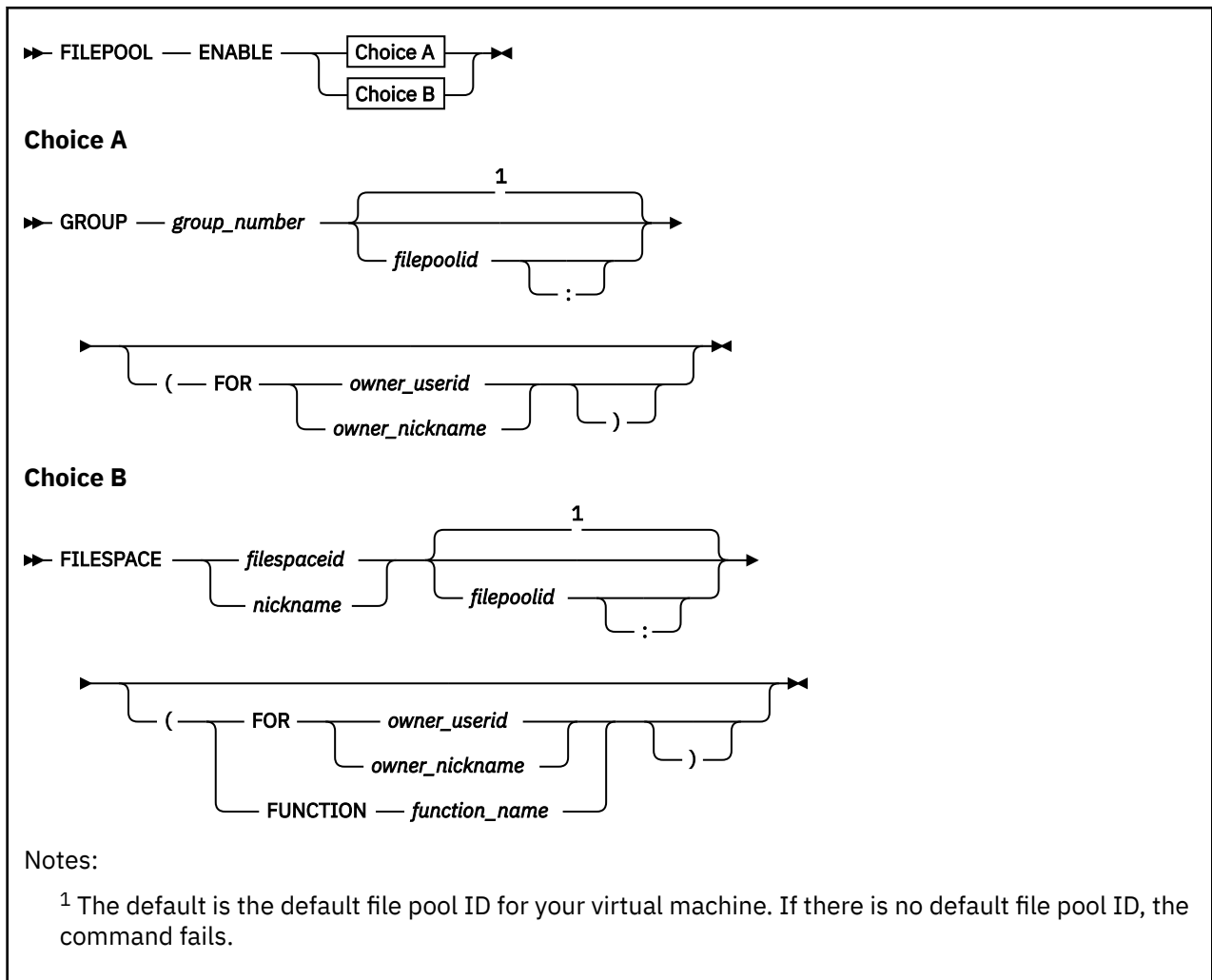
Messages:

- DMS065E *option* option specified twice [RC=065]
- DMS066E SHARE and DETACH are conflicting options [RC=066]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS149E Userid *userid* not valid [RC=149]
- DMS637E Missing nodeid for the AT operand [RC=637]
- DMS647E Userid not specified for {*nickname/owner\_nickname*} in *userid* NAMES file [RC=647]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=647]
- DMS653E Error executing *command* rc=*nn* [RC=653]
- DMS1139E You are not authorized to enter this command [RC=1139]

## FILEPOOL DISABLE

- DMS1209E Nickname *owner\_nickname* resolved to more than one user ID; only one file space can be disabled or enabled at a time [RC=1209]
- DMS1209E Nickname *nickname* resolved to more than one user ID; only one lock owner is allowed [RC=1209]
- DMS1214E {You|*Ownerid*} already {hold|holds} the requested lock [RC=1214]
- DMS1214E File space already locked in *mode1* mode [by you] and you have requested a *mode2* lock [for *owner\_userid*] [RC=1214]
- DMS1214E Storage group already locked in *mode1* mode [by you] and you have requested a *mode2* lock [for *owner\_userid*] [RC=1214]
- DMS1215E File space already locked in *mode1* mode by another user and you have requested a *mode2* lock [for *owner\_userid*] [RC=1215]
- DMS1215E Storage group already locked in *mode1* mode by another user and you have requested a *mode2* lock [for *owner\_userid*] [RC=1215]
- DMS1223E There is no default file pool currently defined [RC=1223]
- DMS2023E File pool *filepoolid* does not support the requested *option* option on the *commandname* command [RC=2023]
- DMS3511E Specified storage group number *storage\_group* is invalid [RC=3511]
- DMS3516E No workunitids currently available [RC=3516]
- DMS3519E Storage group *storage\_group* does not exist or you are not authorized to it [RC=3519]
- DMS3519E File space *filespace* does not exist or you are not authorized to it [RC=3519]
- DMS3519E Storage group *storage\_group* does not exist in file pool *filepoolid* [RC=3519]
- DMS3532E This userid does not have administrator authority, or the FOR owner userid does not have administrator authority for file pool *filepoolid* [RC=3532]
- DMS3555E Error on disable of {storage group *storage\_group* | file space *filespace*|*nickname*} in file pool *filepoolid*. [Reason code = *reasoncode*] [RC=3555]
- DMS3556E Error on workunitid allocation request. Reason code = *reason\_code* [RC=3556]

## FILEPOOL ENABLE



### Authorization

Repository File Pool Administrator

### Purpose

Use the FILEPOOL ENABLE administrator command to remove the lock from a disabled storage group or file space. This command provides function equivalent to the CSL routines DMSENAFS - Enable File Space and DMSENASG - Enable Storage Group, which are described in [z/VM: CMS Callable Services Reference](#).

### Operands

#### GROUP *group\_number*

indicates the number of a storage group to be enabled. You can choose any number from 2 to the MAXDISKS value specified for the file pool for *group\_number*.

For more information, see [MAXDISKS](#).

### **FILESPACE** *filepaceid*

#### **FILESPACE** *nickname*

identifies the file space to be enabled. *Nickname* may be a nickname for a single user ID, or Byte File System, but it cannot be a nickname for a list of user IDs or Byte File Systems. (Use the NAMES command to define nicknames.)

#### *filepoolid*

#### **filepoolid:**

identifies the file pool in which the file space or storage group being enabled exists. for your virtual machine is used.

## Options

### **FOR** *owner\_userid*

#### **FOR** *owner\_nickname*

is the user ID of the *owner* of the lock. This could be the *owner\_userid* specified in the FILEPOOL DISABLE administrator command, the DISABLE operator command, or the DMSDISFS call. *Owner\_nickname* may be a nickname for a single user ID, but it cannot be a nickname for a list of user IDs. (Use the NAMES command to define nicknames.)

### **FUNCTION** *function\_name*

identifies the function that disabled the file space. The FUNCTION operand is only valid with the ENABLE FILESPACE format of the command. The list of valid function names currently consists of one name, RENAME, which is valid only for the FILESPACE version of the command.

This should be done only in extreme cases where the FILEPOOL RENAME command cannot finish and the lock needs to be removed in order to recover.

## Usage Notes

1. The FILEPOOL ENABLE command unlocks the storage group or file space, allowing general access to the objects within.
2. If the ENABLE is done on behalf of another user ID, FOR *owner\_userid* or FOR *owner\_nickname* must be specified.
3. This command requires the MAINT 193 disk be accessed.
4. Callable Services Library (CSL) routines that disable and enable storage groups and file spaces are available in VMLIB. They are:

**DMSDISFS** disables a file space

**DMSDISSG** disables a storage group

**DMSENAFS** enables a file space

**DMSENASG** enables a storage group

These routines can be called from application programs that process in an administration machine or from a user machine if the user is the owner. See [z/VM: CMS Callable Services Reference](#) for more about these routines.

5. SFS operator commands can be entered on the SFS file pool server operator console to disable and enable storage groups and file spaces. They are:

**FILEPOOL DISABLE FILESPACE** disables a file space

**FILEPOOL DISABLE GROUP** disables a storage group

**FILEPOOL ENABLE FILESPACE** enables a file space

**FILEPOOL ENABLE GROUP** enables a storage group

6. The FUNCTION option is only valid with the ENABLE FILESPACE format of the command. When this option is used, the specified file space and its associated storage group that were disabled because a Rename User ID attempt using the FILEPOOL RENAME command, are unlocked.



7. The FILEPOOL ENABLE command is unsuccessful if it is entered from an exec or assembler program, and the file pool specified is active on the current default work unit (that is, has some work on it which has not been committed or rolled back).
8. When data spaces are successfully created and a storage group is disabled and then enabled, the enable processing will be unsuccessful if there is a FBA minidisk not allocated in 8-block increments and not aligned on a 4KB boundary. Message DMS3075E is issued when an enable of a storage group is unsuccessful because an FBA alignment error. Message DMS3074E is issued on the server console to indicate which FBA minidisks are not aligned properly.

Otherwise, if there were no data spaces created, when a storage group is enabled and a minidisk is detected that is not aligned properly, message DMS3074I is issued for each minidisk, and processing continues.

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#)

### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warning(s) issued.

**8 or greater**

Unusual termination. The return code is the same as the message number of associated message.

When there is an error condition, the FILEPOOL ENABLE command, in many cases, displays more than one error message. Generally, the first error message displayed describes the immediate cause of the error. This message is often followed by additional error messages that further describe the error. The return code for this command is the message number.

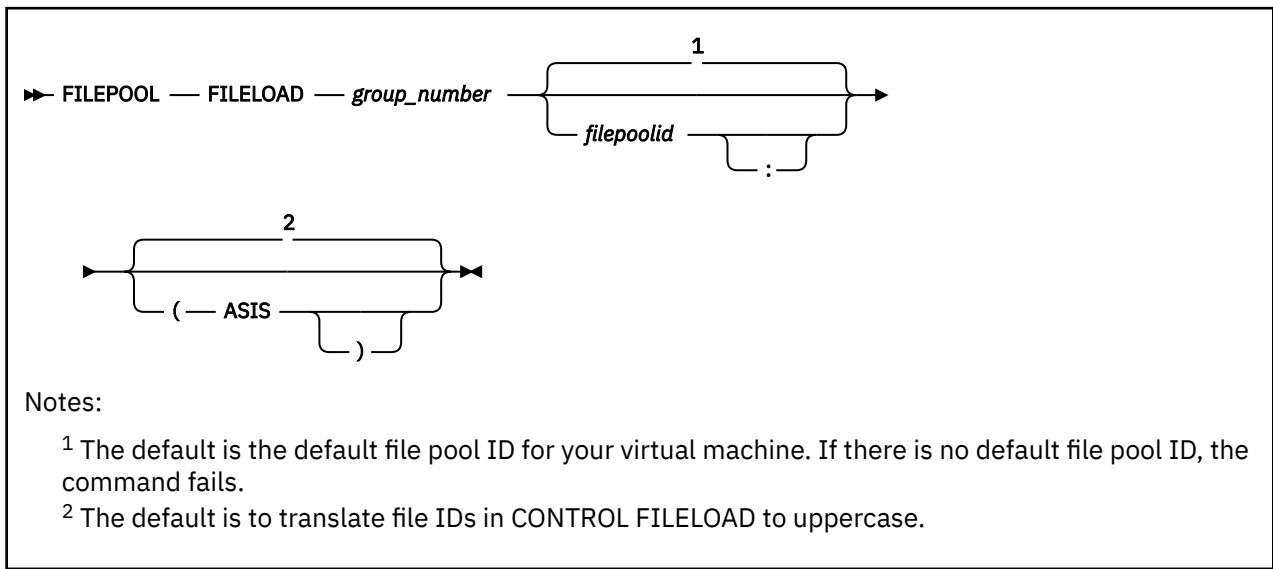
Messages:

- DMS065E *option* option specified twice [RC=065]
- DMS066E FOR and FUNCTION are conflicting options [RC=066]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS149E Userid *userid* not valid [RC=149]
- DMS637E Missing node ID for the AT operand [RC=637]
- DMS647E Userid not specified for [*nickname/owner\_nickname*] in *userid* NAMES file [RC=647]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=647]
- DMS653E Error executing *command rc=nn* [RC=653]
- DMS1139E You are not authorized to enter this command [RC=1139]
- DMS1206W No locks are held by you for file space *filespace* [RC=4]
- DMS1206W No locks are held by *userid* for file space *filespace* [RC=4]
- DMS1206W No locks are held by you for storage group *storage\_group* [RC=4]
- DMS1206W No locks are held by *userid* for storage group *storage\_group* [RC=4]
- DMS1206W No locks are held by function *function* for file space *filespace* [RC=4]
- DMS1209E Nickname [*nickname/owner\_nickname*] resolved to more than one user ID; only one file space can be disabled or enabled at a time [RC=1209]
- DMS1209E Nickname *nickname* resolved to more than one user ID; only one lock owner is allowed [RC=1209]
- DMS1223E There is no default file pool currently defined [RC=1223]

## FILEPOOL ENABLE

- DMS2023E File pool *filepoolid* does not support the requested *option* option on the *commandname* command [RC=2023]
- DMS2533E Function *function\_name* is not a valid function [RC=2533]
- DMS3075E Enable of storage group *group\_number* failed because an FBA alignment error [RC=3075]
- DMS3511E Specified storage group number *storgroupnum* is invalid [RC=3511]
- DMS3516E No workunitids currently available [RC=3516]
- DMS3519E Storage group *storage\_group* does not exist or you are not authorized to it [RC=3519]
- DMS3519E File space *filespace* does not exist or you are not authorized to it [RC=3519]
- DMS3519E Storage group *storage\_group* does not exist in file pool *filepoolid* [RC=3519]
- DMS3532E This userid does not have administrator authority, or the FOR owner userid does not have administrator authority for file pool *filepoolid* [RC=3532]
- DMS3554E Error on enable of {storage group *storage\_group* | file space *filespace|nickname*} in file pool *filepoolid*. [Reason code = *reasoncode*] [RC=3554]
- DMS3556E Error on workunitid allocation request. Reason code = *reason\_code* [RC=3556]

## FILEPOOL FILELOAD



### Authorization

Repository File Pool Administrator

### Purpose

Use the FILEPOOL FILELOAD command to restore base files from a backup file created by FILEPOOL BACKUP. The files to be restored are listed in a control file. This command requires pool administration authority.

### Operands

#### *group\_number*

is the number of the storage group from which the backup file was created.

#### *filepoolid*:

#### *filepoolid*:

is the ID of the file pool in which the files are to be restored. It defaults to the default file pool ID for your virtual machine.

### Options

#### ASIS

prevents FILEPOOL FILELOAD from translating file IDs in CONTROL FILELOAD to uppercase. The default is to translate to uppercase.

### Usage Notes

1. You list the files to be restored in CONTROL FILELOAD A. The records in CONTROL FILELOAD A must have the following format:

```
filename filetype dirname
```

*dirname* is the directory name from which the backup file was taken. If you omit the ID of the file pool in *dirname*, it defaults to the file pool ID used by the FILEPOOL FILELOAD command.

The length of each record in the control file is limited to 256 bytes. Upper and lowercase characters are considered to be equivalent unless the ASIS option is specified.

- Prior to issuing the FILEPOOL FILELOAD command, you must also enter a FILEDEF command to define the input file that contains the backup data. This file must be created by the execution of the FILEPOOL BACKUP command. Specify RESTORE as the ddname. For example, to define a tape input file, you might enter:

```
filedef restore clear
filedef restore tap1 sl 1 volid 111111
```

This removes any existing definition for ddname RESTORE and defines a tape file at virtual address 181 using IBM standard labels. The tape volume ID is 111111. The file is the first on the tape.

An example of a FILEDEF command that defines a disk input file is:

```
filedef restore disk sg000003 sgbackup b
```

This defines a CMS disk file named SG000003 SGBACKUP on file mode B.

The input file must have a variable-blocked record format with a record length of 29024 and a block size of 29028 (RECFM VB LRECL 29024 BLKSIZE 29028). The RECFM, LRECL, and BLKSIZE values are specified internally. If you specify your own RECFM, LRECL, or BLKSIZE values on the FILEDEF command, they will be ignored.

For tape devices, if necessary, specify the label options and the tape device specifications such as number of tracks and density. You can also use a LABELDEF command for a number of functions associated with tapes. For example, if the backup spans more than one tape, you can have the FILEPOOL FILELOAD command display tell you which tape to mount next. Use LABELDEF to tell the system which tapes will be used. Enter:

```
labeldef backup volid ?
```

and answer the prompts for the volume IDs of the backup tapes. Now FILEPOOL BACKUP can tell you which volume to mount. For more information on how to establish volume IDs, see the TAPE command in *z/VM: CMS Commands and Utilities Reference*. For more about the FILEDEF and LABELDEF commands, see *z/VM: CMS Commands and Utilities Reference*. For more about tape handling, see *z/VM: CMS User's Guide*.

- This command requires the MAINT 193 disk be accessed.
- If the file pool ID to be restored by the FILEPOOL FILELOAD command does not match that contained in the backup data (for example, if the file pool ID was changed since the backup file was created), you will be asked if execution should continue. This is done to ensure you are not trying to restore files from the wrong file pool.
- The requested file will be restored to a directory with the same name as the directory where it existed at the time the backup was made. Any existing file in this directory with the same file name and file type will be replaced. If the file does not exist, it will be created. If a directory with the specified *dirname* is not found, the file will not be restored.
- SFS authorizations and aliases are handled as follows:
  - If the restore process results in replacing an existing file with the same *filename filetype dirname*, the authorizations and aliases of the existing file are preserved. Note that they may not necessarily be the same as the authorizations and aliases of the file at the time the backup file was created.
  - If the restore process results in the creation of a new file, the newly created file has no aliases. The newly created file has only those authorizations created as a result of current directory level authorizations (NEWREAD or NEWWRITE authorizations for file control directories, or DIRREAD or DIRWRITE authorizations for directory control directories). So, the authorizations and aliases that existed at the time the backup file was created are not restored.
  - If *filename filetype dirname* represents an alias to a base file, the alias will not be restored. Use the CREATE ALIAS command to recreate it. For more information, see *z/VM: CMS Commands and Utilities Reference*.

7. The command will also work if the user's file space whose files you are restoring has been moved to a different storage group since the backup file was created. (The *group\_number* parameter is used for input file verification only.)
8. If a particular file cannot be restored because of an error condition, you will be informed with a message. If at all possible, the restore process for the rest of the files will continue.
9. The FILEPOOL FILELOAD command internally verifies the syntax of the entries in file CONTROL FILELOAD. If an entry is syntactically incorrect, a message will be displayed and the file will not be restored. If the number of the detected syntax errors reaches 20, the FILEPOOL FILELOAD command will be canceled.
10. Duplicate entries in file CONTROL FILELOAD are ignored. For example, if you have more than one entry in file CONTROL FILELOAD with the same *filename filetype dirname*, the file will be restored only one time.
11. A *catastrophic error* is any error of the FILEPOOL FILELOAD command that causes the usual stopping logic to be avoided. A catastrophic error occurred if you do not see a usual or unsuccessful completion message from the command. If such a condition occurs, the FILEPOOL FILELOAD command can be rerun.

If a catastrophic error has occurred, and the virtual machine was in DOS SVC mode at the time the FILEPOOL FILELOAD command was called, it is possible the machine will be left in OS SVC mode. Depending on how DOS SVC mode was established in the virtual machine, you will need to either re-IPL or log off and log on again.

12. The FILEPOOL FILELOAD command is not supported for execution in the CMS batch facility.
13. If the file pool is managed by DFSMS/VM, it may contain migrated files. The FILEPOOL FILELOAD command automatically restores the migrated data into the appropriate DFSMS/VM repositories.

When restoring files which were migrated at the time the backup file was created, the following should be kept in mind:

- a. The virtual machine in which the FILEPOOL FILELOAD command processes must have the following authorizations:
  - Administration authority for the file pool that contains the storage group into which files are being restored
  - Administration authority for the file pool that contains the Migration Level 1 (ML1) directory for the storage group
  - Read authority to the DFSMS/VM control file DGTVCNTL DATA in directory VMSYS:DFSMS.CONTROL
- b. While the FILEPOOL FILELOAD command is executing, the file pool with the Migration Level 1 directory for the storage group into which files are being restored and the VMSYS file pool must also be available.
- c. The FILEPOOL FILELOAD command requires certain CSL routines which reside in the DFSMS/VM-supplied library FSMINT CSLLIB. Therefore, an access to FSMINT CSLLIB is needed if you have migrated data. This can be accomplished by linking and accessing the disk that contains the DFSMS/VM product code.
- d. If you are using the Migration Level 2 capabilities of DFSMS/VM Level 221 (ML2 is defined in the DFSMS/VM control file), IBM Language Environment with the C Language option is required. The disks containing Language Environment must be accessed prior to invoking the FILEPOOL FILELOAD command. If only ML1 is defined in the DFSMS/VM control file, Language Environment is not necessary.
- e. While restoring the migrated data, the FILEPOOL FILELOAD command may display some messages issued by DFSMS/VM. Such messages are prefixed with the characters **FSM** and are documented in *z/VM: DFSMS/VM Messages and Codes*.
- f. If the migration level 1 directory does not exist it will be recreated by the FILEPOOL FILELOAD processing. This will be done only if the file space that is to contain the migrated data exists. (This

file space is specified in the DFSMS/VM control file. For more information see [z/VM: DFSMS/VM Storage Administration](#).)

For detailed information about DFSMS/VM, migrated files and managing storage groups, see [z/VM: DFSMS/VM Storage Administration](#) and [z/VM: DFSMS/VM Planning Guide](#).

14. The input to FILEPOOL FILELOAD for a damaged minidisk in a file pool can be obtained by issuing the FILEPOOL LIST MINIDISK command. See [“Replacing Storage Group 2 through n Minidisks”](#) on page [132](#) if you are replacing the data in a single file pool minidisk.
15. The FILEPOOL LIST BACKUP command may be issued to determine the contents of a backup file created by the FILEPOOL BACKUP command. For details, see [“FILEPOOL LIST BACKUP”](#) on page [459](#).
16. The FILEPOOL FILELOAD of Byte File System (BFS) files is not recommended because the following restrictions apply:
  - file IDs specified in the CONTROL FILELOAD must be in the CMS system generated short name format and
  - requested files must exist in the file pool.

If you would like the same type of function as the FILEPOOL FILELOAD command you will need to use the FILEPOOL UNLOAD command for backing up BFS file spaces, and the FILEPOOL RELOAD command with the FILES parameter. This will allow you to restore files in a similar manner to FILEPOOL FILELOAD.

### Messages and Return Codes

#### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warning(s) issued.

**8 or greater**

Unusual stopping. The return code is the same as the message number of associated error message.

When there is an error condition, the FILEPOOL FILELOAD command, in many cases, displays more than one error message. Generally, the first error message displayed describes the immediate cause of the error. This message is often followed by other error messages that describe more about the error. The return code for this command is the message number.

# FILEPOOL FORMAT AUDIT

► FILEPOOL — FORMAT — AUDIT ◄

## Authorization

File Pool Administrator

## Purpose

Use the FILEPOOL FORMAT AUDIT command to format the security audit data created by server processing. The formatted output is placed in a file you can print or display. For more information on using this command, see [“Auditing Security”](#) on page 143.

## Usage Notes

1. The command does not call server processing and does not access the file pool. It does not require any file pool authorizations. You can enter the command from any virtual machine that has read access to the security audit trace file.
2. This command requires the MAINT 193 disk be accessed.
3. CRR audit records traces the following commands that can be used to intervene into CRR activity:
  - CRR recovery server operator commands:
    - CRR ERASE LU
    - CRR ERASE LUWID
    - CRR RESYNC
  - File pool operator commands:
    - ERASE LUNAME
    - FORCE PREPARED
4. The FILEPOOL FORMAT AUDIT command prompts for selective audit file processing if desired. This will let you only format audit records that match any of certain criteria:
  - Only CRR audit records or exclude the CRR audit records
  - User IDs that are being checked for authority
  - User IDs that own objects
  - Types of authorization required
  - Server function codes
  - Date range
  - Time range
  - Authority check results (successful, unsuccessful, or successful because special authority)

FILEPOOL FORMAT AUDIT will create a temporary file named \$\$\$TEMP \$\$\$INPUT on the first read/write file mode in the CMS search order. The file is erased when the FILEPOOL FORMAT AUDIT command completes.
5. The FILEPOOL FORMAT AUDIT command allows the user to either format CRR audit records from the audit file or to exclude the CRR audit records from being formatted.
6. You can avoid the selectivity prompts by creating a control file. The control file must have a fixed-length, 80-byte record format (RECFM F, LRECL 80). Prior to invoking the FILEPOOL FORMAT AUDIT command enter a FILEDEF command to associate your control file with the ddname INPUTCTL.

The control statements can be in any order. One of the following keywords must be the first word on each record. Only one keyword is allowed per record. Following the keyword, you would specify valid parameters:

### **AUTHREQ**

Follow this keyword with up to 10 numbers that represent various kinds of authorization checks.

### **CRRIGNOR**

This keyword has no parameters. It excludes CRR audit records from being formatted.

### **CRRONLY**

This keyword has no parameters. It lets you to format and write only CRR audit records.

### **DATE**

Follow this keyword with either one date, or two dates (to represent a range of dates). Use the form mm/dd/yy for the dates.

### **DUMPALL**

This keyword has no parameters. It causes all audit data to be formatted. If you specify DUMPALL, all other control statements are ignored. DUMPALL includes records associated with CRR recovery server processing.

### **FILEREQ**

Follow this keyword with up to 10 numbers that represent kinds of requests made to the file pool server.

### **OWNERID**

Follow this keyword with up to six user IDs. The user IDs represent the owners of objects that someone was trying to access.

### **RESULTS**

Follow this keyword with number 1, 2, or 3. Specify 1 for unsuccessful requests, 2 for successful requests, 3 for requests that were successful only because of special authority.

### **TIME**

Follow this keyword with two time values, representing a range of times. Use the form hh:mm:ss for the times.

### **USERID**

Follow this keyword with up to six user IDs. The user IDs represent users that were trying to access the object.

For a complete description of the keyword control statements, see [“Keyword Control Statements for FILEPOOL FORMAT AUDIT”](#) on page 149.

7. Prior to entering the FILEPOOL FORMAT AUDIT command, enter a FILEDEF command to associate the security audit file with ddname INPUT. The file must have been created by server security audit trace processing. (For information about the AUDIT startup parameter, see Chapter 20, “File Pool Server Startup Parameters,” on page 339.) The file is assumed to have the following characteristics (FILEPOOL FORMAT AUDIT ignores any RECFM, LRECL, or BLOCK specification on the FILEDEF command):

```
RECFM = VB      LRECL = 384      BLOCK = 4096
```

8. Formatted output is written to DDNAME OUTPUT. The user must enter a FILEDEF command for DDNAME OUTPUT before running the FILEPOOL FORMAT AUDIT command. The FILEPOOL FORMAT AUDIT command creates an output file with the following characteristics (the command will ignore any RECFM, LRECL, or BLOCK specification on the FILEDEF command):

```
RECFM = FA or FBA  LRECL = 133
```

The output to DDNAME=OUTPUT is similar to the following. The first output record shown below (beginning with AUDITPOINT) is split into two lines so that it will fit on the page.



```

AUDITPOINT=nnnnn AUTHORITY REQUIRED=nnn FP FUNCTION CODE=nnn AUTHORITY CHECKING DONE
BY=nnn DATE=mm/dd/yy TIME=hh:mm:ss
RESULTS OF AUTHORITY CHECK=snnnnn USERID OF REQUESTOR=xxxxxxx OID OF RESOURCE=xxxxxx
QUALIFIED OBJECT NAME=filename filetype dirname

FORCE PREPARED      FILEPOOL:filepool LUWID:nnnnnnn.1111111.iiiiiiiiiii.ssss
TOKEN:ttttttt
mm/dd/yy hh:mm:ss REQUESTING USERID:userid COMMIT END USERID:userid RESULT:xxxxxxx
Transaction tag: transaction_tag_field

```

For an explanation of the output format, see [“Output Format of FILEPOOL FORMAT AUDIT”](#) on page 160.

9. If the FILEPOOL FORMAT AUDIT command fails, it can be rerun.
10. Messages are written to the virtual machine console.
11. When the FORCE PREPARED command is entered, the resulting audit file contains a token displayed as a hexadecimal value and has the label *TOKEN:*. You can match this token with the token (called Taskid) from the output of the QUERY PREPARED command.
12. If the LPP option on the CP SPOOL command is 0 (LPP OFF), the number of lines per page for the formatted output will be 58. If the LPP value is greater than 0, the number of lines per page for the formatted output will be equal to the LPP value. For more information, see [z/VM: CP Commands and Utilities Reference](#).

## Messages and Return Codes

### Return Codes:

- 0** Successful.
- 8** Unsuccessful.

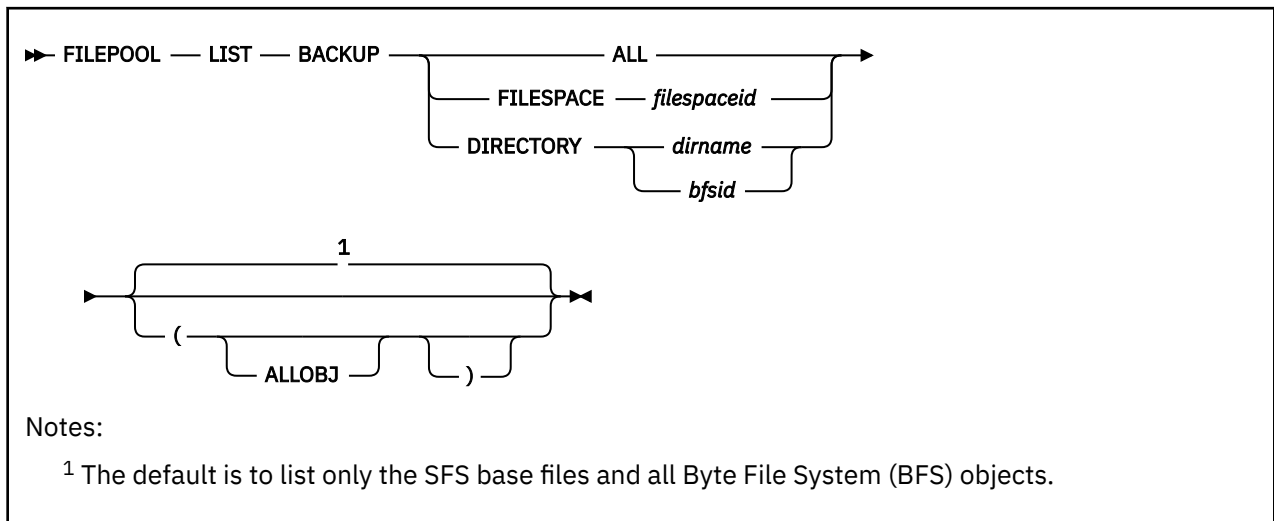
### Messages:

- DMS3450R Enter CRR selections: '1' to Ignore, '2' for Only, or just press enter to skip this selection
- DMS3476E Input keyword *keyword* not valid
- DMS3477E Number of parameters in xxxxxxxx
- DMS3487R Enter AUDIT selections: 1 (All) or 2 (Select)
- DMS3478E {AUTHREQ|FILEREQ} parameters must be numeric and less than 256
- DMS3479E Syntax error in one of the {DATE|TIME} parameters
- DMS3480E RESULTS parameter can only be 1, 2, or 3
- DMS3481E The audit file does not contain the audit data
- DMS3482W The audit file is empty
- DMS3485I FILEPOOL processing begun at *time on date*
- DMS3486I FILEPOOL processing ended at *time on date*
- DMS3487R Enter AUDIT selections: 1 (All) or 2 (Select)
- DMS3488R Enter up to 6 userids or just press enter to skip this selection
- DMS3489R Enter up to 6 ownerids or just press enter to skip this selection
- DMS3490R Enter up to 10 authorization types or just press enter to skip this selection
- DMS3491R Enter up to 10 file pool server function codes or just press enter to skip this selection
- DMS3492R Enter a date range (mm/dd/yy) or just press enter to skip this selection
- DMS3493R Enter a time range (hh:mm:ss) or just press enter to skip this selection
- DMS3493R Enter a time range (hh:mm:ss) or just press enter to skip this selection

## FILEPOOL FORMAT AUDIT

- DMS3494R Enter authority check results wanted: 1 (Unsuccessful), 2 (Successful), 3 (Successful because special authority), or just press enter to skip this selection
- DMS3495E Write error from CMS EXECIO on file \$\$TEMP \$\$INPUT. RC=*rc*
- DMS3496E An error occurred for CMS FILEDEF INPUTCTL DISK \$\$TEMP \$\$INPUT. RC=*rc*
- DMS3497E Input is not in the correct format
- DMS3498R You have not entered any special selections. Enter 9 (quit audit file processing) or just press enter to process all the audit records
- DMS3499R Enter 1 (if DDNAME=INPUTCTL is already available), 9 (to quit audit file processing), or just press enter if you want to be prompted for audit processing
- DMS3900E {Open|Close|Write|Read} error DDNAME = *ddname*. REASON1 = *reason1*. REASON2 = *reason2*

## FILEPOOL LIST BACKUP



### Authorization

Repository File Pool Administrator

### Purpose

The FILEPOOL LIST BACKUP command lists the SFS base files and Byte File System (BFS) pathnames contained in a user storage group backup or unload file created by the FILEPOOL BACKUP or FILEPOOL UNLOAD command.

SFS objects include SFS:

- Directories
- Base files
- Aliases
- External Objects

BFS objects include BFS:

- Block Special files
- Character Special files
- Directories
- External Links
- Regular files
- FIFO
- Symbolic Link

### Operands

#### ALL

indicates all base files and BFS pathnames in the input file are to be listed.

#### FILESPACE *userid*

indicates all base files and BFS pathnames for a file space in the input file are to be listed. The file space is identified by a *filepaceid*.

**DIRECTORY** *dirname***DIRECTORY** *bfsid*

indicates all base files for a directory in the input file are to be listed. The *dirname* represents the fully qualified directory name of an SFS directory whose objects are to be listed. Note that *filepoolid:userid.n1.n2...n8* is the only allowed form for *dirname*. The *bfsid* identifies the Byte File System.

**Options****ALLOBJ**

indicates all objects are to be listed for SFS file spaces and that both the long pathname and CMS short name are to be listed for BFS file spaces.

**Usage Notes**

1. This command requires the MAINT 193 disk be accessed.
2. Authorization is implied through the ownership of the backup or unload file. The file pool which was backed up or unloaded is not referenced during the FILEPOOL LIST BACKUP command processing, so authorization to a file pool is not relevant.
3. The only valid device types for the input and output files (for example, when using FILEDEF) are tape and disk.
4. Prior to entering the FILEPOOL LIST BACKUP command, you must enter FILEDEF commands to define the input backup or unload file and the output file that will contain the list of objects in the backup or unload file. Specify BACKUP as the ddname for the input file, and LISTBKUP as the ddname for the output file. For example, to define a tape input file, you might enter:

```
filedef backup tap1 sl volid 111111
```

This defines a tape file at virtual address 181 using IBM standard labels. IBM standard labels allow multivolume tape file support. The tape volume id is 111111.

An example of a FILEDEF command that defines a disk output file is:

```
filedef listbkup disk report file c
```

This defines a CMS disk file named REPORT FILE on file mode C.

The RECFM, LRECL, and BLKSIZE values of the input and output files are specified internally. If you specify your own RECFM, LRECL, or BLKSIZE values on the FILEDEF command, they will be ignored.

The input file containing the backup or unload data must be created by the execution of the FILEPOOL BACKUP or FILEPOOL UNLOAD command.

For tape devices, if necessary, you should specify the label options and the tape device specifications such as number of tracks and density. A LABELDEF command may also be supplied if desired. For more information about the FILEDEF and LABELDEF commands, see *z/VM: CMS Commands and Utilities Reference* and *z/VM: CMS Application Development Guide for Assembler*. For more about tape handling, see *z/VM: CMS User's Guide*.

5. When a selection is made for a directory, the object listing does not include any objects in any of its subdirectories.
6. If you want to use the output created by this command as input to the FILEPOOL FILELOAD command (for only SFS objects) or the FILEPOOL RELOAD FILES command (for SFS or BFS objects), do not use the ALLOBJ option.
7. All BFS pathnames will be appended with a slash ('/'). This is necessary for FILEPOOL RELOAD FILES to denote where the pathname ends. The slash will be removed from the pathname by FILEPOOL RELOAD FILES processing.
8. A catastrophic error of the FILEPOOL LIST BACKUP command occurred if you do not see a usual or unsuccessful completion message from the command.

If a catastrophic error has occurred, you may want to enter the command again to see if it completes successfully. There might have been a temporary problem that caused the original error. If you get a catastrophic error again, or if you decide not to enter the command again, the following condition may exist:

- If the virtual machine was in DOS SVC mode at the time the FILEPOOL LIST BACKUP command was issued, it is possible that the machine will be left in OS SVC mode. You will need to either re-IPL or log off and log on again to reset to DOS SVC mode.
9. If the FILEPOOL LIST BACKUP ALL is issued for a storage group containing both SFS and BFS file spaces, the heading(s) for the file space(s) being listed will be included with the file space data as needed.

## Responses

If the ALLOBJ option is not specified, the structure of the information displayed in the output file looks like the following:

**Note:** The Storage Group ... will not be displayed if the input file is for a FILEPOOL UNLOAD of a file space.

```
Time hh:mm:ss      Filepool List Backup - fpoolid
Date mm/dd/yy      Listing Selection - selection
                   Storage Group - groupnumber
```

```
Backup File Created:
mm:dd:yy AT hh:mm:ss
```

**Note: For SFS you would also see**

```
Filename  Filetype  Dirname
-----  -
filename filetype dirname
```

**Note: For BFS you would see instead**

```
Path name
-----
pathname
```

The following is the structure of the information that can be displayed in the object list output file if the ALLOBJ option is specified:

```
Time hh:mm:ss      Filepool List Backup - fpoolid
Date mm/dd/yy      Listing Selection - selection
                   Storage Group - groupnumber
```

```
Backup File Created:
mm:dd:yy AT hh:mm:ss
```

**Note: For SFS you would also see**

```
Type          Filename  Filetype  Dirname
-----  -
type          filename filetype dirname
ALIAS_BASE    filename filetype dirname
DFSMS ID      dfsms_idenfier
Authorizations : userid authtype
```

**Note: For BFS you would see instead**

```
Type  CMS short name          Path name
-----  -
type  filename filetype          pathname
```

## Time and Date

is the local time and date the FILEPOOL LIST BACKUP command was issued.

### Filepool List Backup

identifies the command issued and the file pool ID associated with the backup data.

### Storage Group

identifies the user storage group number. This will not be displayed if the input file was the unload file of a file space.

### Listing Selection

identifies the selection parameter where *selection* is either ALL, FILESPACE, or DIRECTORY.

### Backup File Created

gives the date and time the BACKUP file was created.

### Type

identifies the type of file as follows:

For SFS:

#### Type

#### Meaning

#### BASE

Base file

#### ALIAS

Alias to a base file

#### BASE\*

DFSMS/VM Migrated base file

#### EXTRNL

External Object

#### DIRECTORY

Directory

For BFS:

#### Type

#### Meaning

#### B

Block special BFS file

#### C

Character special BFS file

#### D

BFS directory

#### E

BFS external link

#### F

BFS regular file

#### P

BFS FIFO

#### S

BFS symbolic link

### Filename

identifies the file name of an object in the backup file.

### Filetype

identifies the file type of an object in the backup file.

### Dirname

identifies the fully qualified SFS directory name of an object in the backup file.

**ALIAS\_BASE**

identifies the fully qualified base file name of an object in a backup file, when the preceding object is a file with a type of Alias.

**DFSMS ID**

is the DFSMS/VM identifier for an object in the backup file that is in migrated storage, (Migration Level 1 (ML1) or Migration Level 2 (ML2)). It is displayed when the preceding object is a file with a type of migrated.

**Authorizations**

identifies the user IDs authorized to access an object in the backup file, and the type of authorization that each user ID is allowed to the object. The following are the authorization descriptions:

- R - Read
- W - Write
- NR - Directory FILECONTROL New Read
- NW - Directory FILECONTROL New Write
- DR - Directory DIRCONTROL Directory Read
- DW - Directory DIRCONTROL Directory Write

**CMS short name**

identifies the CMS file name and file type of an object in the backup file that is unique within a BFS to each file.

**Path name**

identifies the path name of an object in the backup file.

**Examples**

1. If you want a list of the base files in the file space identified by user ID SMITH, enter the following command:

```
FILEPOOL LIST BACKUP FILESPACE SMITH
```

The following is an example of a response:

```
Time hh:mm:ss      Filepool List Backup - VMSYSU
Date dd/mm/yy      Listing Selection - FILESPACE
                   Storage Group - 2

Backup File Created:
    dd/mm/yy AT hh:mm:ss
```

```
Filename  Filetype  Dirname
-----
REQMNTS  ADDED     VMSYSU:SMITH
REQMNTS  DEFINED   VMSYSU:SMITH.PROJECT.TASKS
STATUS   REPORT    VMSYSU:SMITH.PROJECT.STATUS
MANAGERS REPORT    VMSYSU:SMITH.REPORTS
```

2. If you need a list of all the objects backed up for the file space identified by user ID SMITH, enter the following command:

```
FILEPOOL LIST BACKUP FILESPACE SMITH (ALLOBJ)
```

The following is an example of a response:

For SFS:

```
Time hh:mm:ss      Filepool List Backup - VMSYSU
Date dd/mm/yy      Listing Selection - FILESPACE
                   Storage Group - 2
```

# FILEPOOL LIST BACKUP

Backup File Created:  
dd/mm/yy AT hh:mm:ss

Type	Filename	Filetype	Dirname
-----	-----	-----	-----
DIRECTORY			VMSYSU:SMITH
Authorizations	: WOLFKM	W	
	WOLFKM	NW	
BASE	REQMNTS	ADDED	VMSYSU:SMITH
Authorizations	: WOLFKM	W	
DIRECTORY			VMSYSU:SMITH.PROJECT
DIRECTORY			VMSYSU:SMITH.PROJECT.TASKS
BASE*	REQMNTS	DEFINED	VMSYSU:SMITH.PROJECT.TASKS
DFSMS ID	00020000	00FBA236	
Authorizations	: WHITEGM	R	
	WOLFKM	W	
DIRECTORY			VMSYSU:SMITH.PROJECT.STATUS
BASE	STATUS	REPORT	VMSYSU:SMITH.PROJECT.STATUS
DIRECTORY			VMSYSU:SMITH.REPORTS
Authorizations	: SOMMERTW	R	
BASE	MANAGERS	REPORT	VMSYSU:SMITH.REPORTS
Authorizations	: SOMMERTW	R	
ALIAS	PROJSTAT	REPORT	VMSYSU:SMITH.REPORTS
ALIAS_BASE	STATUS	REPORT	VMSYSU:SMITH.PROJECT.STATUS

## For BFS:

Time hh:mm:ss Filepool List Backup - VMSYSU  
Date dd/mm/yy Listing Selection - FILESPACE  
Storage Group - 2

Backup File Created:  
dd/mm/yy AT hh:mm:ss

Type	CMS short name	Path name
-----	-----	-----
D		/.. /VMBFS:VMSYSU:SMITH/
F	2	0 /.. /VMBFS:VMSYSU:SMITH/REQMNTS ADDED /
D	3	0 /.. /VMBFS:VMSYSU:SMITH/PROJECT/
D	4	0 /.. /VMBFS:VMSYSU:SMITH/REPORTS /
F	5	0 /.. /VMBFS:VMSYSU:SMITH/REPORTS/MANAGERS REPORT/
F	6	0 /.. /VMBFS:VMSYSU:SMITH/REPORTS/PROJSTAT REPORT /
D	7	0 /.. /VMBFS:VMSYSU:SMITH/PROJECT/STATUS/
F	8	0 /.. /VMBFS:VMSYSU:SMITH/PROJECT/STATUS/STATUS REPORT/
D	9	0 /.. /VMBFS:VMSYSU:SMITH/PROJECT/TASKS/
F	10	0 /.. /VMBFS:VMSYSU:SMITH/PROJECT/TASKS/REQMNTS DEFINED/

3. If you need a list of all the objects backed up for the SFS directory represented by VMSYSU:SMITH.REPORTS, enter the following command:

```
FILEPOOL LIST BACKUP DIRECTORY VMSYSU:SMITH.REPORTS (ALLOBJ
```

The following is an example of a response:

Time hh:mm:ss Filepool List Backup - VMSYSU  
Date dd/mm/yy Listing Selection - DIRECTORY  
Storage Group - 2

Backup File Created:  
dd/mm/yy AT hh:mm:ss

Type	Filename	Filetype	Dirname
-----	-----	-----	-----
DIRECTORY			VMSYSU:SMITH.REPORTS



```

Authorizations : SOMMERTW R
BASE           MANAGERS REPORT   VMSYSU:SMITH.REPORTS
Authorizations : SOMMERTW R
ALIAS         PROJSTAT REPORT   VMSYSU:SMITH.REPORTS
ALIAS_BASE    STATUS   REPORT   VMSYSU:SMITH.PROJECT.STATUS

```

4. If you need a list of the base files backed up in a user storage group in VMSYSU, enter the following command:

```
FILEPOOL LIST BACKUP ALL
```

The response resembles that for a file space, but includes all the base files backed up for all the user IDs and Byte File Systems (BFS) found in the VMSYSU user storage group.

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

### Return Codes:

#### 0

Successful completion.

#### 4

Successful completion, warning(s) issued.

#### 8 or greater

Unusual stopping. The return code is the same as the message number of associated message.

When there is an error condition, the FILEPOOL LIST BACKUP command, in many cases, displays more than one error message. Generally, the first error message displayed describes the immediate cause of the error. This message is often followed by additional error messages that further describe the error. The return code for this command is the message number.

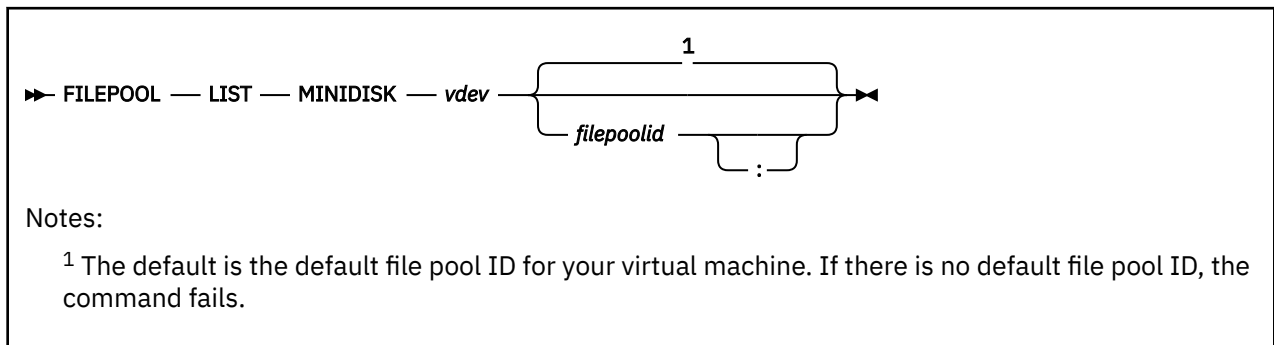
### Messages:

- DMS3221E File definition specified for LISTBKUP is identical to file definition for BACKUP [RC=3221]
- DMS3438I FILEPOOL LIST BACKUP successful [RC=0]
- DMS3438E FILEPOOL LIST BACKUP unsuccessful [RC=3438]
- DMS3439W No objects found for FILEPOOL LIST BACKUP FILESPACE *userid* [RC=4]
- DMS3439W No objects found for FILEPOOL LIST BACKUP DIRECTORY *dirname* [RC=4]
- DMS3451E Input file specified for FILEPOOL LIST BACKUP is not a backup file for file pool *filepoolid* [RC=3451]
- DMS3514E Action *action* incorrect. Second parameter must be *keyword* [RC=3514]
- DMS3514E Action *action* incorrect. Third parameter must be *keyword* [RC=3514]
- DMS3514E Action *action* incorrect. Fourth parameter must be *keyword* [RC=3514]
- DMS3520E Error detected in routine *DFSMS routine\_name* [RC=3520]
- DMS3521E No FILEDEF specified for *ddname* file [RC=3521]
- DMS3522E BACKUP file record was not generated by backup [RC=3522]
- DMS3526E BACKUP File is inconsistent [RC=3526]
- DMS3527E *ddname* file device type is not tape or disk [RC=3527]
- DMS3528E Close for *ddname* file failed [RC=3528]
- DMS3529E Irrecoverable I/O error on *ddname* file [RC=3529]
- DMS3530E Unexpected end of file on *ddname* file [RC=3530]
- DMS3531E Insufficient virtual storage [RC=3531]

## FILEPOOL LIST BACKUP

- DMS3557E Error on *ddname* file open [RC=3557]
- DMS3570E Error on CMS DMSFREE or DMSFRET request [RC=3570]
- DMS3608E Directory *dirname* is not fully qualified [RC=3608]
- DMS3621E Error on *functionname*. Reason code = *code*. [RC=3621]

## FILEPOOL LIST MINIDISK



### Authorization

Repository File Pool Administrator

### Purpose

The FILEPOOL LIST MINIDISK command lists the SFS or Byte File System (BFS) files that have at least one 4KB block of data on a specific file pool storage group minidisk.

The FILEPOOL LIST MINIDISK command takes as input a virtual device address identifying a minidisk in a user storage group. The output of the command is a file containing a list of all the base files in that file pool storage group that have at least one 4KB block of data on the minidisk identified by the input virtual device number. This provides the support to recover only the affected files when a minidisk in a file pool user storage group is damaged. The output can be used as input to the FILEPOOL FILELOAD command or FILEPOOL RELOAD FILES command to restore a list of files.

Both SFS files and Byte File System (BFS) files are supported.

### Operands

#### *vdev*

is the virtual device number of a storage group minidisk.

#### *filepoolid*

#### *filepoolid:*

is the ID of the file pool in which the minidisk resides.

### Usage Notes

1. This command requires the MAINT 193 disk be accessed.
2. The only valid device types for the input and output files (for example, when using FILEDEF) are tape and disk.
3. Prior to issuing the FILEPOOL LIST MINIDISK command, you must enter a FILEDEF command to define the output file that will contain the list of base files. Enter the FILEDEF command with LISTMDSK as the ddname. For example, to define a disk output file you might enter:

```
filedef listmdsk disk report file c
```

This defines a CMS disk file named REPORT FILE on file mode C.

4. The FILEPOOL LIST MINIDISK command obtains a *share disable lock* for the storage group in which the minidisk resides. A share disable lock lets users read from the storage group but does not allow any updates. The FILEPOOL LIST MINIDISK command fails if any DIRCONTROL directories in the storage group are accessed in read/write status. (When DIRCONTROL directories are accessed read/write, the FILEPOOL LIST MINIDISK command cannot obtain a *share disable lock*. The QUERY

ACCESSORS command displays information about current accessors of DIRCONTROL directories.) If the storage group is already locked by the administrator issuing the FILEPOOL LIST MINIDISK command, or if it is locked *share* by another administrator, processing will continue with that lock, and the lock will be preserved after the command completes. If the storage group is locked *exclusive* by another administrator, message DMS3585E is issued, and command processing stops. The QUERY FILEPOOL DISABLE command shows who has disable locks for a storage group.

5. The QUERY FILEPOOL MINIDISK command can be used to get the virtual address of a minidisk in a file pool's storage group.
6. The output file containing the list of base files may reside only on a disk or tape. For tape, use IBM standard labels if you want multivolume tape file support.
7. The output file containing the list of base files contains only non-migrated files because the data for migrated files is not on the specified virtual device.
8. For a damaged file pool minidisk, the FILEPOOL LIST MINIDISK command can be used to get a listing of damaged files. The output list file can be used as the input file for the FILEPOOL FILELOAD command or the FILEPOOL RELOAD FILES command in which case the output file must be renamed to CONTROL FILELOAD or CONTROL RELOAD. See [“Replacing Storage Group 2 through n Minidisks” on page 132](#) for the steps needed to replace all the files in a single file pool minidisk.
9. A catastrophic error of the FILEPOOL LIST MINIDISK command occurred if you do not see a usual or unsuccessful completion message from the command.

If a catastrophic error has occurred, you may want to enter the command again to see if it completes successfully. There might have been a temporary problem that caused the original error. If you get a catastrophic error again, or if you decide not to enter the command again, the following conditions may exist:

- a. The storage group may be disabled. To enable the storage group, use the ENABLE operator command, the FILEPOOL ENABLE administrator command, or the Enable Storage Group (DMSENAGS) CSL routine. For more information, see [“ENABLE” on page 415](#) and [“FILEPOOL ENABLE” on page 447](#). For more information on the DMSENASG CSL routine, see [z/VM: CMS Callable Services Reference](#).
  - b. If the virtual machine was in DOS SVC mode at the time the FILEPOOL LIST MINIDISK command was entered, it is possible the machine will be left in OS SVC mode. You will need to either re-IPL or log off and log on again to reset to DOS SVC mode.
10. When a Byte File System (BFS) file is listed in the output:
    - *filename* and *filetype* are the file's CMS short file name.
    - *dirname* is the BFS's fully qualified top directory name (for example, VMSYSU:BFSPROJA).

## Responses

The output list is a list of fully qualified file names for all base files that have at least one block of data allocated on the specified minidisk.

The records in the output list have the following format:

```
filename filetype dirname
```

where:

```
filename filetype
is the name of an SFS file or the CMS short name
of a BFS file.

dirname is the fully qualified directory
name for SFS files, or bfsid for BFS files.
```

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warning(s) issued.

**8 or greater**

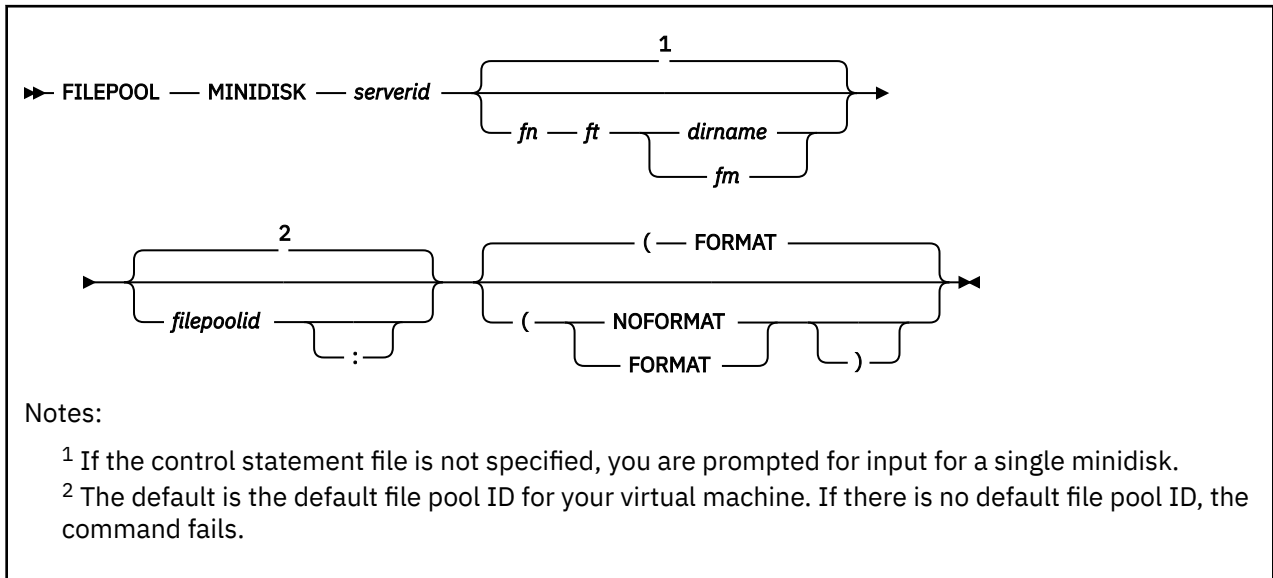
Unusual stopping. The return code is the same as the message number of associated message.

When there is an error condition, the FILEPOOL LIST MINIDISK command, in many cases, displays more than one error message. Generally, the first error message displayed describes the immediate cause of the error. This message is often followed by additional error messages that further describe the error. The return code for this command is the message number.

Messages:

- DMS0649E Extraneous parameter(s) *parameter*. [RC=649]
- DMS1223E There is no default file pool currently defined. [RC=1223]
- DMS1240E You are not authorized to connect to file pool *filepoolid* [RC=1240]
- DMS3209E *Request* request failed. Return code = *return\_code* Reason code = *reason\_code* [RC=3209]
- DMS3298E File pool *filepoolid* does not support the *operand* operand on the FILEPOOL command [RC=3298]
- DMS3438E FILEPOOL LIST MINIDISK command unsuccessful [RC=3438]
- DMS3438I FILEPOOL LIST MINIDISK command successful [RC=0]
- DMS3452E Incorrect file pool catalog interface level, level = *level*. [RC=3452]
- DMS3514E Action *action* incorrect. Must be LIST MINIDISK [RC=3514]
- DMS3515E *input\_parameter* is an incorrect parameter. [RC=3515]
- DMS3516E No workunitids currently available. [RC=3516]
- DMS3518E File pool *filepoolid* is unavailable or unknown. [RC=3518]
- DMS3519E Storage group *group* does not exist in file pool *filepoolid*. [RC=3519]
- DMS3521E No FILEDEF specified for *ddname* file [RC=3521]
- DMS3527E *ddname* file device type is not tape or disk. [RC=3527]
- DMS3528E CLOSE for *ddname* file failed. [RC=3528]
- DMS3529E Irrecoverable I/O error on *ddname* file. [RC=3529]
- DMS3531E Insufficient virtual storage [RC=3531]
- DMS3532E This userid does not have administrator authority for file pool *filepoolid* [RC=3532]
- DMS3557E Error on *ddname* file open. [RC=3557]
- DMS3585E Cannot access storage group *storgroup* in the file pool *filepoolid*. Conflicting lock outstanding. [RC=3585]
- DMS3621E Error on called *CMS command/macro* Reason code = *reason\_code*. [RC=3621]
- DMS3628E Inconsistent catalog information. [RC=3628]
- DMS3638I No file data blocks were found for virtual device *vdev*. [RC=0]
- DMS3639E Virtual address *vdevis* not part of a user storage group. in file pool *filepoolid* [RC=3639]

## FILEPOOL MINIDISK



### Authorization

Repository File Pool Administrator

### Purpose

Use the FILEPOOL MINIDISK command to add minidisks to storage groups in a file pool while in multiple user mode and to update the appropriate POOLDEF file in the server machine. This command requires file pool administration authority.

### Operands

#### *serverid*

is the user ID of the SFS server machine.

#### *fn ft dirname*

#### *fn ft fm*

identifies the input control statement file that contains one or more minidisk definition control statements. The format of a minidisk definition control statement is shown on [Minidisk Definition Control Statement](#). These control statements will be added to the *filepoolid* POOLDEF file in the *serverid* machine. If the input control statement file is contained in an SFS directory, specify *fn ft dirname*. If the input control statement file is contained on a minidisk or accessed directory, specify *fn ft fm*. If you do not specify the file name, file type, and file mode or fully qualified directory name, you will be prompted for input for a single minidisk.

#### *filepoolid*

#### *filepoolid:*

is the file pool ID of the file pool for which the new minidisk will be added. If you do not specify a file pool ID, the default file pool ID of your virtual machine will be used.

### Options

#### FORMAT

causes the minidisks to be formatted and reserved when they are added to the file pool. A block size of 4096 is used for the minidisk. FORMAT must not be used for a remote file pool.

**NOFORMAT**

is used when the new minidisks have already been formatted and reserved. NOFORMAT must be used when you add minidisks to a remote file pool.

**Minidisk Definition Control Statement:** This control statement identifies a new minidisk for a new or existing storage group. The format of a minidisk definition control statement is:

```
DDNAME=MDKnnnnn VDEV=vdev GROUP=grpnum BLOCKS=blocks
```

where:

**MDKnnnnn**

is the ddname the FILEPOOL command processing will use.

The *nnnnn* can range from 3 through 32767 and must be assigned in ascending consecutive numeric sequence. For example, it is an error to use DDNAME=MDK00005 before DDNAME=MDK00004 is used. DDNAME=MDK00005 must be the next DDNAME used after DDNAME=MDK00004. The MAXDISKS value is the upper limit of what can be specified for *nnnnn*. MAXDISKS is a control statement in the control statement file required by the FILESERV GENERATE command. See FILESERV GENERATE command for more information on the MAXDISKS control statement.

If you are not sure what the next available MDKnnnnn is, use the QUERY FILEPOOL MINIDISK command to find the highest MDK number. Then use the next number for *nnnnn* field. For a complete description, see [“QUERY FILEPOOL MINIDISK” on page 586](#).

**VDEV=vdev**

identifies the virtual device number for the minidisk being added to the file pool.

**GROUP=grpnum**

identifies the storage group (*grpnum*) in the file pool to which the minidisk is being added. The storage group may or may not already exist. Storage group numbers do not need to be used in consecutive numeric sequence. *grpnum* can range from 1 to 32767 but cannot exceed the MAXDISKS value specified for the file pool.

**BLOCKS=blocks**

is an optional parameter handled automatically just as it is during FILESERV GENERATE. For more information on the BLOCKS parameter, see [“FILESERV GENERATE” on page 513](#).

**Usage Notes**

1. The minidisks to be added must have been added to the directory of the *serverid* virtual machine and the directory placed online before invoking FILEPOOL MINIDISK.
2. This command requires the MAINT 193 disk be accessed.
3. GROUP=1 is used exclusively for the file pool catalogs. During file pool generation, storage group 1 is allocated at least one minidisk.
4. GROUP=2 is used for user data. During file pool generation, storage group 2 is allocated at least one minidisk.
5. GROUP=3-32767 can be optionally used for user data. These groups may or may not have been allocated minidisks during file pool generation.
6. The GROUP values 3-32767 do not need to be assigned in ascending consecutive numeric sequence. For example, GROUP=5 can be assigned before GROUP=4.
7. If successful, FILEPOOL MINIDISK will search the accessed R/W minidisks in the server machine for the first occurrence of the *filepoolid* POOLDEF file. FILEPOOL MINIDISK will then update the *filepoolid* POOLDEF file with the new minidisk control statements.

During this processing, FILEPOOL MINIDISK creates a temporary file, \$STEMP \$POOLDEF, on the same file mode as the *filepoolid* POOLDEF file. If FILEPOOL MINIDISK processing is successful, the temporary file is erased.

8. If the BACKUP startup parameter has been specified, there must be a current, valid designation for the backup.
9. If a control definition statement file name is specified on the command, and the file is changed while FILEPOOL MINIDISK is in progress, it may yield unpredictable results.
10. If the FORMAT option is specified or defaulted, and ESM is not installed, the user may be prompted for passwords for establishing links to the minidisks.<sup>8</sup> All prompts for the passwords will be done initially unless there are not enough free virtual addresses. In the latter case, prompts for passwords may be spread over time.
11. Before entering the FILEPOOL MINIDISK command, you should ensure all FBA storage group minidisks are allocated in 8-block increments and aligned on a 4KB boundary. FILEPOOL MINIDISK command processing will cause an error until the minidisks are realigned on 4KB boundaries.
12. For every FBA minidisk not aligned on a 4KB boundary, FILEPOOL MINIDISK processing will cause an error message DMS3074E. This message is only issued up to 10 times even if there are more than 10 minidisks that are to be added.
13. If the BACKUP startup parameter has been specified, the FILEPOOL MINIDISK command causes a control data backup to be initiated by the file pool server. The command will return to the error before the backup of the control data completes. Any error conditions found in the file pool server because the incorrect definition of the backup file will cause messages to be issued to the terminal where the command was entered.

A reader file, \$\$\$SFS \$MSGS, will be sent to the administrator who entered the FILEPOOL MINIDISK command only if the BACKUP startup parameter was specified. There is a possibility you may not get this file because of spool space problems or the network not being available. If you wait for the file longer than the time it normally takes for control data to be backed up, you should take a look at the server console. Error messages after the backup is initiated, because of errors in writing the backup file, will appear in the \$\$\$SFS \$MSGS file and on the file pooloperator console. The \$\$\$SFS \$MSGS file will also contain messages which will tell you if the backup was successful and where it was directed or if the backup was not successful and the cause of the error.

If the BACKUP startup parameter is specified, the minidisks will not be available for use until the control data has been successfully backed up.

14. Error situations that arise during the sequence of steps that include the FILEPOOL MINIDISK function require different administrator actions based on which step was unsuccessful. The actions are as follows:
  - If any command error occurs, the error situation should be corrected and the FILEPOOL MINIDISK command entered again.
  - If the file pool server stops as a result of an error, the action you take depends on whether message DMS3922I was displayed on the file poolserver console when the error occurred. Message DMS3922I is:

```
DMS3922I n minidisk(s) were added to the
file pool
```

Take recovery actions as follows:

- If the server stopped and message DMS3922I is not displayed, and if there are messages indicating the error, take the appropriate action and rerun the command. Otherwise, if there are no messages, start the server and rerun the command.
- If the server stopped and message DMS3922I is displayed, you need to edit the POOLDEF file. If the POOLDEF file does not contain the control statements for all the minidisks you added, you need to add them from the \$\$TEMP \$POOLDEF file. To do this, edit the \$\$TEMP \$POOLDEF file and delete all of the lines *except* for the ones with the values you assigned to the minidisk or

<sup>8</sup> In a RACF® or other External Security Manager (ESM) protected environment, prompts for passwords may not occur. Refer to the appropriate product documentation for how minidisk linking authorization is performed.



minidisks that were to be added. Then, copy the \$\$TEMP \$POOLDEF file to the bottom of the POOLDEF file which adds the statements to the POOLDEF file for the new minidisks.

After you have updated the POOLDEF file, you need to start the SFS server again unless the BACKUP startup parameter is specified. If the BACKUP startup parameter is specified, you must stop the FILESERV BACKUP command before starting the SFS server again. If you get a message indicating backup has been started, but the server errors prior to completing the control data backup, the server cannot be restarted until control data is backed up using the FILESERV BACKUP command. The minidisks are available for use after the server is successfully started.

## Responses

```
DMS3425R  Enter MDK number (nnnnn), virtual device
           address (vvvv), storage group number
           (ggggg) for a minidisk to be added.
           Use format nnnnn vvvv ggggg
```

Enter the information for a single minidisk to be added. Refer to Minidisk Definition Control Statement for an explanation of MDK number, virtual address, and storage group number. Only use the *nnnnn* part of the MDK*nnnnn*. Do not use 'MDK' characters preceding the MDK number. The numbers need not be padded with zeros. As an example, if you need to add a new minidisk with virtual device address 306 in storage group 2, and you know the next sequential MDK number is 3, your response to the prompt would be:

```
3 306 2
```

Entering data that does not fit in the format of *nnnnn vvvv ggggg* will cancel the execution of the command.

If the FORMAT option is specified or defaulted, the second prompt that is unsuccessful will get the following:

```
DMS3426I  The following minidisks will be
           formatted and reserved:
DMS3426I  MDKnnnnn vvvv ggggg
DMS3427R  Format will erase all files on above minidisks.
           Do you wish to continue? Enter 1 (Yes) or 0 (No).
```

If you wish to continue, enter **1**. To stop FILEPOOL MINIDISK processing, enter **0**. This response cannot be stacked.

Message DMS3426I will display each minidisk being added when the FORMAT option is specified or defaulted.

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warning(s) issued.

**8 or greater**

Unusual ending. The return code is the same as the message number of associated message.

When there is an error condition, the FILEPOOL MINIDISK command, in many cases, displays more than one error message. Generally, the first error message displayed describes the immediate cause of the error. This message is often followed by additional error messages that further describe the error. The return code for this command is the message number.

## Messages:

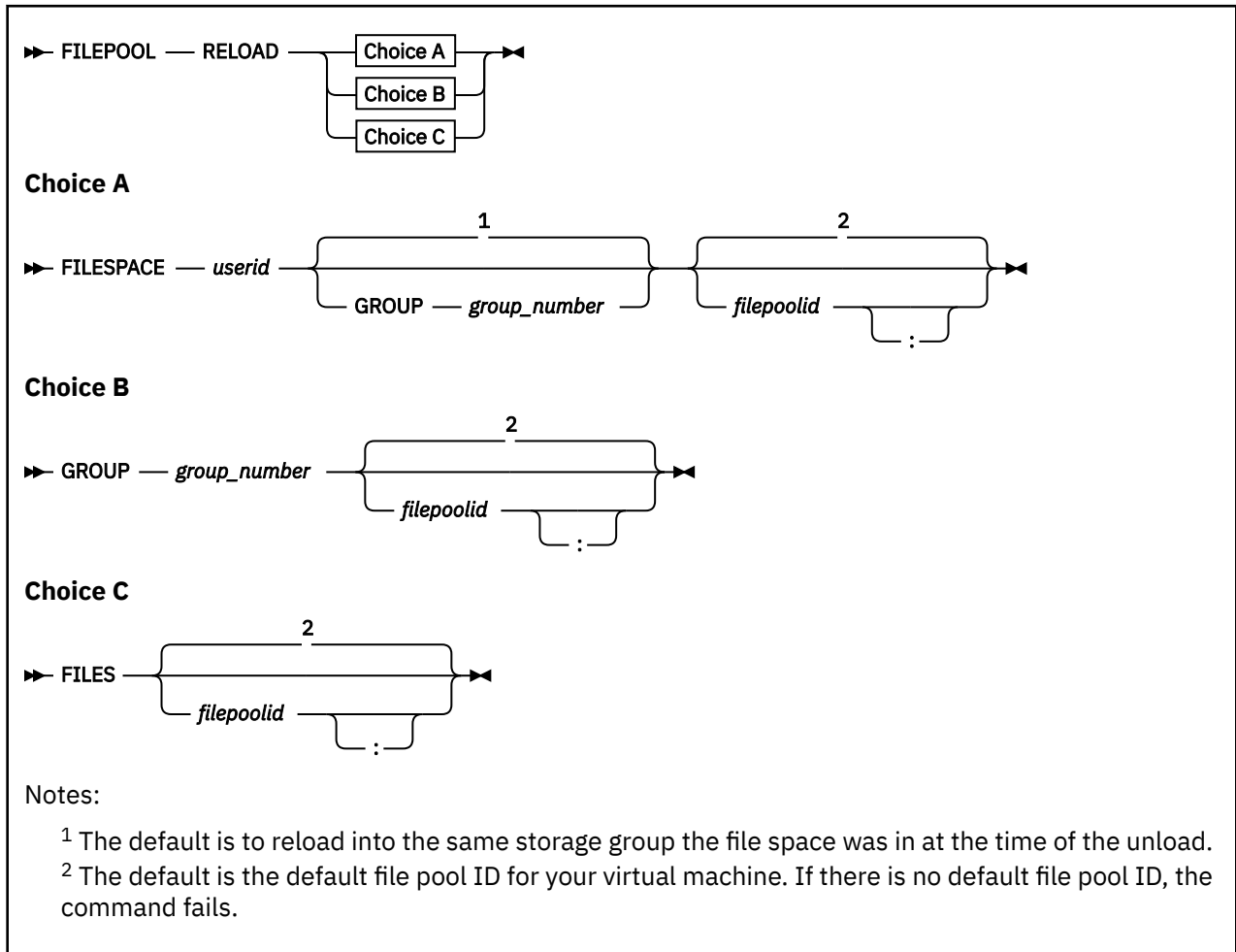
- DMS0167E Free storage management error, internal error code *code* [RC=0167]
- DMS1027E Only one option may be specified [RC=1027]
- DMS1115E Incorrect control statement *control-statement* [in *fn ft fm*] [RC=1115]
- DMS1116E Incorrect value *value* for *parameter* [in *fn ft fm*] [RC=1116]
- DMS1118E No available filemode for FORMAT and RESERVE [RC=1118]
- DMS1132E Incorrect number of operands [RC=1132]
- DMS1135E No control statements exist in *fn ft fm* [RC=1135]
- DMS1176E Virtual storage capacity exceeded for file pool *filepoolid* [RC=1176]
- DMS1184E {File *fn ft fm*|Directory *dirname*} not found or you are not authorized for it [RC=1184]
- DMS1223E There is no default file pool currently defined [RC=1223]
- DMS1291E There are no unused work units available [RC=1291]
- DMS1313E Duplicate virtual device *vdev* specified [RC=1313]
- DMS3052E *response* is incorrect, please reenter the correct response [RC=3052]
- DMS3074E Minidisk at *cuu* is on an FBA disk, and not properly aligned [RC=3074]
- DMS3298E File pool *filepoolid* does not support the *operand* operand on the FILEPOOL command [RC=3298]
- DMS3404W File pool limit of *nnnnn* minidisks has been reached [RC=4]
- DMS3409E Requested number of blocks for *ddname* is incorrect [RC=3409]
- DMS3420E Return code *code* from *command* command for minidisk with virtual device number *vdev* [RC=3420]
- DMS3421E Error copying POOLDEF file. Reason code = *code* [RC=3421]
- DMS3422E No available virtual device number for CP LINK command [RC=3422]
- DMS3424W A warning was issued from *routine\_name* routine, return code was *return-code* reason code was *reason-code*. Processing continues [RC=4]
- DMS3429E An error occurred in *name* {command|routine}, return code was *return\_code*, [reason code was *reason-code* [for file pool *filepoolid*]] [RC=3429]
- DMS3430E Error {writing|opening|closing} POOLDEF file. Reason code = *reason-code* [RC=3430]
- DMS3431E FILEPOOL MINIDISK command rejected because FILEPOOL MINIDISK is being performed. [file pool = *filepoolid*] [RC=3431]
- DMS3432E FILEPOOL MINIDISK command rejected because minidisk that is to be added with virtual device number *vdev* already exists in the file pool [RC=3432]
- DMS3433E Data in *fn ft fm* may have changed [RC=3433]
- DMS3434E FORMAT option is not valid for a remote file pool server *serverid*, NOFORMAT must be specified [RC=3434]
- DMS3435E *value* is incorrect for the *parameter* parameter on the *command\_name* command [RC=3435]
- DMS3518E File pool *filepoolid* is not available [RC=3518]
- DMS3531E Insufficient virtual storage [RC=3531]
- DMS3532E This user ID does not have administrator authority [RC=3532]
- DMS3556E Error on workunitid allocation request. Reason code = *code* [RC=3556]
- DMS3600E Control data backup file not defined. [file pool = *filepoolid*] [RC=3600]
- DMS3629E Directory *dirname* is incorrect or is not fully qualified [RC=3629]
- DMS3909E The DDNAME = *ddname* is out of sequence [RC=3909]
- DMS3910E Incorrect storage group number for DDNAME = *ddname* [RC=3910]

- DMS3913E Filepool CONTROL disk is incorrect size. Reason = *n* [RC=3913]
- DMS3917E Filepool cannot have more than *n* minidisks [RC=3917]
- DMS3926E The IUCV limit for the virtual machine was exceeded [RC=3926]
- DMS3927E Open error {CONNECT|DISKID} on minidisk, DDNAME *ddname* DASD *vdev*, RETCODE *nn* [RC=3927]

The following messages can be returned in the \$\$\$SFS \$MSG\$ file.

- DMS1141W User filespace threshold [still] exceeded [for file pool *filepoolid*]
- DMS1156S Supervisor error {1;|2;} return code *retcode* {,reason code *reascode*}
- DMS1259E File pool *filepoolid* has run out of physical space in the storage group
- DMS3207E Can not lock file *fn ft* in *dirname* Reason Code *code*
- DMS3209E *Rtname* request errored. Return Code= *code1* Reason Code=*code2*
- DMS3222E Tape specified for BACKUP contains current control data backup file
- DMS3231E Backup scheduled for file pool *filepoolid* by *command* command was unsuccessful
- DMS3231I Backup scheduled for file pool *filepoolid* by *command* command was successful
- DMS3516E No workunits are currently available
- DMS3609E File pool *filepoolid* is unavailable or unknown
- DMS3613I [Last successful] control data backup file was directed to tape device *tape*
- DMS3614I [Last successful] control data backup file was directed to minidisk. File name: *filename* File type: *filetype* File mode: *filemode*
- DMS3615 [Last successful] control data backup file was directed to file pool *filepoolid*. Filename: *filename* File type: *filetype* File mode: *filemode*
- DMS3900E {Open|Close|Write|Read} error DDNAME=*ddname* REASON1=*reason1* REASON2=*reason2*

## FILEPOOL RELOAD



### Authorization

Repository File Pool Administrator

### Purpose

The FILEPOOL RELOAD command uses the output of the FILEPOOL UNLOAD command to recreate the contents of SFS or Byte File System (BFS), file space(s) in another storage group or file pool, or to restore the contents of user file space(s) in the same storage group or file pool. This provides a function similar to what FILESERV MOVEUSER provides, without the need to stop the server, and without the restriction that the file space be moved within the same file pool. Also provided is a function similar to FILEPOOL RESTORE without the restriction that the storage group minidisk configuration remain unchanged since the time of the backup.

For more information on the effects of restoring a user storage group, see [“Restoring User Data”](#) on page 115.

### Operands

#### FILESPACE *userid*

is the identifier of the target file space for the reload. If a file space does not exist, the *userid* is automatically ENROLLED in the target filepool.

**GROUP *group\_number***

is the number of the target storage group for the reload. If GROUP is not specified in conjunction with the FILESPACE operand, the default is to reload into the same storage group that the file space was in at the time of the unload.

**FILES**

indicates a set of files input in a separate control file (CONTROL RELOAD), and any authorizations granted to them, will be reloaded. If the restore replaces an existing file, its authorizations and aliases are preserved. If the target does not exist it will be created. For more information see the usage note for this operand (number [“24”](#) on page 480).

***filepoolid******filepoolid:***

is the ID of the file pool in which the file space will be reloaded. If not specified, the default file pool ID for the virtual machine will be used. If there is no default file pool ID, the command will be unsuccessful.

**Usage Notes**

1. This command requires the MAINT 193 disk be accessed.
2. FILEPOOL RELOAD reloads the contents of the file created by the FILEPOOL UNLOAD command into an SFS or BFS file space or storage group.

The file created by the FILEPOOL UNLOAD command contains information on all the files and directories in the file space(s) as well as the following:

- a. For SFS file spaces:
  - aliases defined in the file space(s) being unloaded
  - authorizations granted on the objects in the file space(s) being unloaded
  - if *ALL* was specified on the FILEPOOL UNLOAD command, aliases in other file spaces in the file pool that refer to base files in the file space(s) being unloaded
  - if *ALL* was specified on the FILEPOOL UNLOAD command, authorizations granted by other users in the file pool to the user ID identifying the file space(s) being unloaded
  - External Objects
  - Any data that was migrated by DFSMS/VM at the time of the unload
- b. For BFS file spaces:
  - Block special files
  - Character special files
  - External Links
  - BFS Regular files
  - Sockets
  - FIFOs
  - Symbolic Links
  - Any data that was migrated by DFSMS/VM at the time of the unload
3. FILEPOOL RELOAD restores the file space or storage group that was unloaded to the file space or storage group specified on the FILEPOOL RELOAD command.
4. If you want to reload a subset of files into an existing file space or storage group, the FILEPOOL RELOAD FILES command may be used.
5. If you want to reload a single file space from a file created by FILEPOOL UNLOAD GROUP, enter FILEPOOL RELOAD FILESPACE for that user ID.
6. The only valid device types for the input and output files (for example, when using FILEDEF) are tape and disk.

7. Prior to issuing the FILEPOOL RELOAD command, you must enter a FILEDEF command to define the input file that contains the unloaded data. Specify RELOAD as the ddname. For example, to define a tape input file, you might enter:

```
filedef reload tap1 sl volid 111111
```

This defines a tape file at virtual address 181 using IBM standard labels. The tape volume id is 111111.

An example of a FILEDEF command that defines a disk input file is:

```
filedef reload disk userx unload b
```

This defines a CMS disk file named USERX UNLOAD on file mode B.

The RECFM, LRECL, and BLKSIZE values of the input file are specified internally. If you specify your own RECFM, LRECL, or BLKSIZE values on the FILEDEF command, they will be ignored.

For tape devices, if necessary, you should specify the label options and the tape device specifications such as number of tracks and density. A LABELDEF command may also be supplied if desired. For more about the FILEDEF and LABELDEF commands, see [z/VM: CMS Commands and Utilities Reference](#) and [z/VM: CMS Application Development Guide for Assembler](#). For more about tape handling, see [z/VM: CMS User's Guide](#).

8. To use the FILEPOOL RELOAD command to restore a user storage group, the following scenario should be followed:
- Enter FILEDEF to define the output file for unload.
  - Enter FILEPOOL UNLOAD to unload the user storage group.
  - Enter FILEDEF to define the input file for reload.
  - Enter FILEPOOL RELOAD to reload the user storage group.

Below is an example in which storage group 2 is backed up and restored using FILEPOOL UNLOAD and FILEPOOL RELOAD commands. In this example, the UNLOAD file is stored on a minidisk.

```
FILEDEF UNLOAD DISK STGRP2 UNLOAD A  
FILEPOOL UNLOAD GROUP 2 (ALL  
FILEDEF RELOAD DISK STGRP2 UNLOAD A  
FILEPOOL RELOAD GROUP 2
```

9. To move a user file space from one storage group to another within the same file pool, the following scenario should be followed:
- Enter a FILEDEF to define the output file for unload.
  - Enter a FILEPOOL UNLOAD to unload the user file space (specify ALL).
  - Enter a FILEDEF to define the input file for reload.
  - Enter a FILEPOOL RELOAD to reload the user file space into the new storage group.

Below is an example in which the file space for user SCHANCK is moved from its storage group 2 to storage group 3. In this example, the UNLOAD file is stored on a minidisk.

```
FILEDEF UNLOAD DISK SCHANCK UNLOAD A  
FILEPOOL UNLOAD FILESPACE SCHANCK (ALL  
FILEDEF RELOAD DISK SCHANCK UNLOAD A  
FILEPOOL RELOAD FILESPACE SCHANCK GROUP 3
```

10. To move a user file space from a storage group in one file pool to a storage group in another file pool, the following scenario should be followed:
- Enter a FILEDEF to define the output file for unload.
  - Enter a FILEPOOL UNLOAD to unload the user file space (do not specify ALL).
  - Enter a FILEDEF to define the input file for reload.
  - Enter a FILEPOOL RELOAD to reload the user file space into the new storage group.

- e. Enter a DELETE USER, to delete the user file space from the source file pool (note that this step is optional; if the intent is to replicate the file space instead of move it, the file space does not have to be deleted).

Below is an example in which the file space for user SCHANCK is moved from its original storage group on file pool SERVER5 to storage group 5 on file pool SERVER8. In this example, the UNLOAD file is stored on a tape.

```
FILEDEF UNLOAD TAP1 SL VOLID 11111
FILEPOOL UNLOAD FILESPACE SCHANCK SERVER5
FILEDEF RELOAD TAP1 SL VOLID 11111
FILEPOOL RELOAD FILESPACE SCHANCK GROUP 5 SERVER8
DELETE USER SCHANCK SERVER5 (NOCONFIRM)
```

11. FILEPOOL RELOAD conditionally restores authorizations, as follows:
  - When the file space or storage group is reloaded in the same file pool, authorizations granted on the contents of the file space are restored. If *ALL* was specified at the time of the FILEPOOL UNLOAD, authorizations granted to the *filespaceid* being reloaded are also restored.
  - When the file space or storage group is moved to a storage group in a different file pool, only authorizations granted on the contents of the file space are restored.
12. FILEPOOL RELOAD conditionally restores aliases, as follows:
  - When the file space or storage group is reloaded to the same file pool, aliases defined in the file space(s) are restored. If *ALL* is specified at the time of the FILEPOOL UNLOAD, aliases defined in other file spaces that refer to base files in the file space(s) being reloaded, are restored.
  - When the file space(s) are moved to a storage group in a different file pool, aliases defined within the file space(s) are restored.
13. If the destination storage group for reload is not large enough to accommodate the specified data from the unload file, the reload will be unsuccessful. If this happens, increase the size of the storage group and re-run the command.
14. If the FILEPOOL RELOAD does not complete successfully, it can be re-run. If you are reloading an entire storage group, message DMS3455I will be issued as each file space in the storage group is successfully reloaded. As an alternative to running the FILEPOOL RELOAD again for the entire storage group it may be faster to reload the remaining file spaces instead. Similarly, if you are reloading a list of files or a file space, you may want to see which files have not been reloaded, and enter the FILEPOOL RELOAD FILES for the remaining files.
15. FILEPOOL RELOAD obtains a *disable exclusive* lock for the file space or storage group being reloaded, except when the FILES operand is specified. In this case, the usual SFS file-sharing rules apply (i.e. a single writer).
16. Before entering the FILEPOOL RELOAD command, for a storage group or file space you should consider entering SET FILEWAIT ON. If you do not enter a SET FILEWAIT ON command, FILEPOOL RELOAD stops if any other user is reading from or writing to the file space when FILEPOOL RELOAD tries to lock it. When FILEWAIT is ON, the file pool server does not stop the FILEPOOL RELOAD command if it cannot immediately obtain the lock. Instead, the file pool server does not let any new activity start. It waits for all activity to end and then gets the lock for FILEPOOL RELOAD. FILEPOOL RELOAD will then reload the file space. When the reload completes, the server automatically relinquishes the lock.
17. A catastrophic error of the FILEPOOL RELOAD command is any error that causes the usual ending logic to be avoided. A catastrophic error occurred if you do not see a usual or unsuccessful completion message from the command.

If a catastrophic error occurs, certain resources may be in an inconsistent state. You may attempt to re-enter the command. If the reload cannot be immediately rerun, the following conditions may exist:

- The file space or storage group may be partially reloaded. The objects reloaded before the error will be intact, and the rest of the file space will be missing. Message DMS3455I will be issued for each file space after it is completely reloaded.

- The file space may be disabled. To enable the file space, use the ENABLE operator command, the FILEPOOL ENABLE command, or the Enable File Space (DMSENAFS) CSL routine.

(See “ENABLE” on page 415, “FILEPOOL ENABLE” on page 447, and *z/VM: CMS Callable Services Reference*, for more information).

- If the virtual machine was in DOS SVC mode at the time the FILEPOOL RELOAD command was called, it is possible the machine will be left in OS SVC mode. If so, you will need to either re-IPL or log off and log on again.
18. FILEPOOL RELOAD is not supported for execution in the CMS batch facility.
  19. If the FILEPOOL RELOAD command is issued to a file pool server at a service level prior to VM/ESA Version 2 Release 1, message DMS3298E will be issued, and the command will not be processed.
  20. A file space or storage group cannot be reloaded if it is already locked SHARE or EXCLUSIVE by another user.
  21. If the file space is managed by DFSMS/VM, the unload file may contain migrated file data. The FILEPOOL RELOAD command automatically reloads both the primary and the migrated data for the file space.
  22. When reloading a file space which has migrated files in it, the following should be kept in mind:
    - a. If an object cannot be reloaded because of an error condition, a message will be displayed and the reload process will stop. The only exception to this is the case where the DFSMS/VM environment is not available. If there are migrated files to be reloaded, and the DFSMS environment is not available, reload will continue for the non-migrated files.
    - b. If the DFSMS/VM environment is not available when the first attempt is made to reload a migrated file, all migrated files will be skipped. Reload will continue for non-migrated files.
    - c. If you are moving file spaces or storage groups which contain DFSMS/VM migrated data to a new storage group or file pool which is not managed by DFSMS, you need to recall the migrated data prior to issuing the FILEPOOL UNLOAD command. Otherwise, the migrated data will not be restored in the new file space or storage group.
    - d. The virtual machine in which the FILEPOOL RELOAD command executes must have the following authorizations:
      - File Pool administration authority for the file pool that contains the file space being reloaded.
      - File Pool administration authority for the file pool that contains the migration level 1 directory for the file space being reloaded.
      - Read authority to the DFSMS/VM control file.
    - e. While the FILEPOOL RELOAD command is executing, the file pool with the migration level 1 directory for the file space being reloaded, and DFSMS/VM, must also be available.
    - f. The FILEPOOL RELOAD command requires certain CSL routines which reside in the DFSMS/VM-supplied library FSMINT CSLLIB. Therefore, an access to FSMINT CSLLIB is needed if you have migrated data.
    - g. While reloading the migrated data, the FILEPOOL RELOAD command may display some messages issued by DFSMS/VM. Such messages are prefix with the characters **FSM** and are documented in *z/VM: DFSMS/VM Messages and Codes*.
- For detailed information about DFSMS/VM, migrated files and managing storage groups, see *z/VM: DFSMS/VM Storage Administration* and *z/VM: DFSMS/VM Planning Guide*.
23. If you move users to new storage groups, and you use FILEPOOL BACKUP and RESTORE to back up your file pool(s), you should back up the two storage groups after completing the FILEPOOL RELOAD process.
  24. The following usage notes apply only to FILEPOOL RELOAD with the FILES parameter specified:
    - a. The file spaces targeted for reloading files must be enrolled in the file pool targeted for reload.
    - b. List the files to be restored in CONTROL RELOAD A.

There are 2 control record formats:



i) *filename filetype dirname*

*filename filetype* is the name of an SFS file or the CMS short name of a BFS file. (The CMS short name identifies the CMS file name and file type of an object in the backup file unique within a BFS to each file.)

*dirname* is the fully qualified directory name for SFS files, or the *bfsid* for BFS files.

If you omit the ID of the file pool in *dirname*, it defaults to the file pool ID used by the FILEPOOL RELOAD FILES command.

ii) *fully qualified pathname* for BFS only

*fully qualified pathname* is in the format: */../VMBFS:filepoolid:filespaceid/pathname*

If a blank is the last character of the pathname, a '/' or X'00' must be appended to delimit the end of the pathname. A '/' or X'00' are valid as ending delimiters in any case, and will not be considered to be part of the pathname. The *filepoolid* and *filespaceid* may not default in the fully qualified pathname.

- c. The CONTROL RELOAD file may contain only 1 format of control record. In other words, control records containing *fully qualified pathnames* for BFS files may not be intermixed with control records containing the *filename filetype dirname* format. Whatever the format is for the first record of the file is the expected format for the remaining records in the file. Any records that do not conform will be ignored.
- d. When the CMS short name is used for BFS files, the file data can only be replaced. The catalog data for the file must be intact. Note that if a minidisk in a user storage group is corrupted, the catalog data for the corrupted file data is still intact.
- e. If the target directory for an SFS file does not exist, it will be created by FILEPOOL RELOAD processing.
- f. If the target directory for a BFS pathname does not exist, it will not be created by filepool reload processing. The directory will be created, however, if a BFS pathname for a directory is specified in the CONTROL RELOAD file.
- g. The pathname of the parent directory of a BFS object to be reloaded may not contain symbolic links. If a symbolic link is encountered during pathname resolution for the directory, the file will not be reloaded.
- h. The length of each record in the control file is limited to 1025 bytes.
- i. SFS file IDs and BFS Pathnames are not translated to uppercase.
- j. The FILEPOOL RELOAD FILES command internally verifies the syntax of the entries in file CONTROL RELOAD. If an entry is syntactically incorrect, a message will be displayed and the file will not be restored. If the number of the detected syntax errors reaches 20, the FILEPOOL RELOAD FILES command will be canceled.
- k. Duplicate entries in file CONTROL RELOAD are ignored. For example, if you have more than one entry in file CONTROL RELOAD with the same *filename filetype dirname* or *pathname*, the file will be restored only one time.
- l. The FILEPOOL RELOAD FILES command restores aliases and authorizations as follows:
  - If the restore process results in replacing an existing file, the authorizations and aliases of the existing file are preserved. If there were other authorizations and aliases at the time of the unload, they will also be restored.
  - If the restore process results in the creation of a new file, the authorizations and aliases that existed at the time of the backup will be restored.
  - If the *filename filetype dirname* represents an alias to a base file, the alias will be recreated.
- m. The input to FILEPOOL RELOAD for a damaged minidisk in an SFS or BFS file space storage group can be obtained by issuing the FILEPOOL LIST MINIDISK command. See [“Replacing Storage Group 2 through n Minidisks”](#) on page 132 if you are replacing the data in a single file pool minidisk.

- n. The FILEPOOL LIST BACKUP command may be entered to determine the contents of a file created by the FILEPOOL UNLOAD command. For details on the command, see [“FILEPOOL LIST BACKUP”](#) on page 459.

### Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

#### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warnings issued.

**8 or greater**

Unusual stopping. The return code is the number of the associated message.

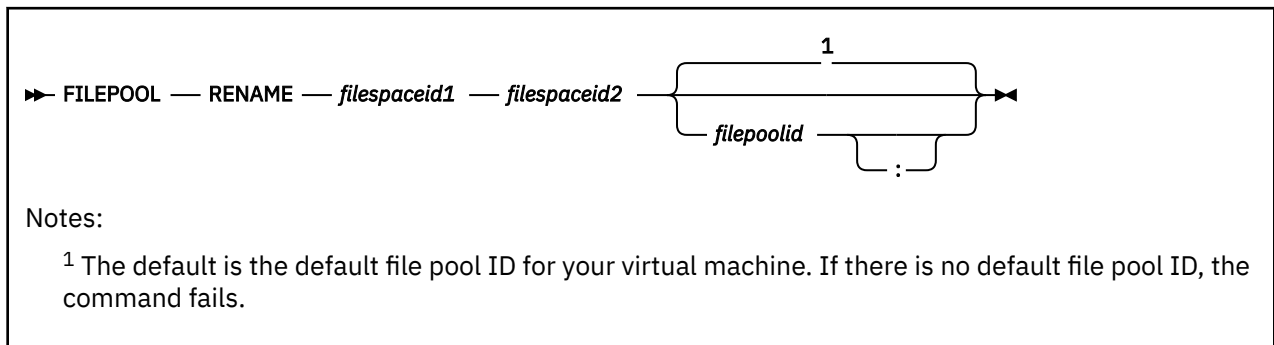
The FILEPOOL RELOAD command can display more than one error message. Generally, the first error message describes the immediate cause of the error. The first message may be followed by additional error messages that further describe the error.

Messages:

- DMS0386E Missing Operands
- DMS0649E Extraneous parameter(s)
- DMS1136E Unable to gain access to library *library\_name*
- DMS1223E There is no default file pool currently defined
- DMS1240E You are not authorized to connect to file pool *filepoolid*
- DMS3208E Unexpected return code *returncode* returned from routine *routine*
- DMS3209E *Rtnname* request failed. Return code = *returncode*. Reason code = *reasoncode*
- DMS3284E DDNAME=RELOAD input file not found
- DMS3298E File pool *filepoolid* does not support the RELOAD operand on the FILEPOOL command
- DMS3438E FILEPOOL RELOAD unsuccessful
- DMS3438I FILEPOOL RELOAD successful
- DMS3452E Incorrect file pool catalog interface level, level = *level*
- DMS3453E Filespace *filespaceid* not found in reload file
- DMS3454E The input file contains the unload data for a filespace. FILEPOOL RELOAD cannot be performed at the storage group level
- DMS3455I The reload of file space *filespaceid* is starting
- DMS3455I The reload of file space *filespaceid* is complete
- DMS3508W Data for user *filespaceid* in storage group *nn* has not been reloaded. User is currently enrolled in a different storage group
- DMS3511E Specified storage group number *nn* invalid
- DMS3512E Incorrect option *option* specified.
- DMS3514E Action (*action*) incorrect. Must be RELOAD
- DMS3515E *Parameter* is an incorrect parameter
- DMS3516E No workunitids currently available
- DMS3518E File pool *filepoolid* is unavailable or unknown
- DMS3519E Storage group *nn* does not exist or you are not authorized to it
- DMS3521E No FILEDEF specified for RELOAD file

- DMS3522E RELOAD file record was not generated by unload
- DMS3523E Reload of migration level *n* files in storage group *nn* failed
- DMS3526E Reload file is inconsistent
- DMS3530E Unexpected end of file on RELOAD file
- DMS3527E RELOAD file device type is not tape or disk
- DMS3528E CLOSE for RELOAD file errored.
- DMS3529E Irrecoverable I/O error on RELOAD file
- DMS3531E Insufficient virtual storage
- DMS3532E This userid does not have administrator authority for file pool *filepoolid*
- DMS3557E Error on RELOAD file open
- DMS3561E Storage group *nn* in filepool *filepoolid* has no associated filespaces
- DMS3562E Storage group *nn* in filepool *filepoolid* has not been modified
- DMS3562E File space *filepaceid* in filepool *filepoolid* has not been modified
- DMS3585E Cannot access storage group *nn* in file pool *filepoolid*. Conflicting lock outstanding
- DMS3585E Cannot access file space *filepaceid* in file pool *filepoolid*. Conflicting lock outstanding
- DMS3594R Storage group *nn* in file pool *filepoolid* will be replaced by storage group *nn* in file pool *filepoolid* in reload file created at *hh:mm:ss* on *mm:dd:yy*. Enter '1' to continue or '0' to cancel
- DMS3594R File space *filepaceid* in storage group *nn* in file pool *filepoolid* will be replaced by file space *filepaceid* in file pool *filepoolid* in reload file created at *hh:mm:ss* on *mm:dd:yy*. Enter '1' to continue or '0' to cancel
- DMS3594R Files from storage group *nn* in filepool *filepoolid* will replace files in filepool *filepoolid* from reload file created at *hh:mm:ss* on *mm:dd:yy*. Enter '1' to continue or '0' to cancel
- DMS3594R Files from file space *filepaceid* in filepool *filepoolid* will replace files in filepool *filepoolid* from reload file created at *hh:mm:ss* on *mm:dd:yy*. Enter '1' to continue or '0' to cancel
- DMS3594R File space *filepaceid* in storage group *nn* in file pool *filepoolid* will be replaced from file pool *filepoolid* in reload file created at *hh:mm:ss* on *mm:dd:yy*. Enter '1' to continue or '0' to cancel.
- DMS3616W File *filename filetype dirname* cannot be restored
- DMS3616W File cannot be restored: *pathname*
- DMS3617I File *filename filetype dirname* successfully restored
- DMS3617I File successfully restored: *pathname*
- DMS3618W File *filename filetype dirname* not found in reload file
- DMS3618W File not found in the reload file: *pathname*
- DMS3620I *n* files restored
- DMS3621E Error on *rtname*. Reason code = *reasoncode*
- DMS3622E Control Reload file not found
- DMS3623E OPEN|READ|CLOSE of control file errored. Reason Code = *reasoncode*
- DMS3629E Directory *dirname* is incorrect or is not fully qualified
- DMS3630E *filepoolid* not equal to target filepool *filepoolid*
- DMS3636E Storage group *nn* in filepool *filepoolid* contains migrated data. DFSMS/VM encountered an error verifying the execution environment
- DMS3636E Filespace *filepaceid* in filepool *filepoolid* contains migrated data. DFSMS/VM encountered an error verifying the execution environment
- DMS3641W Path name incorrect or not fully qualified: *pathname*
- DMS3641W Symbolic link encountered in path name prefix: *pathname*

## FILEPOOL RENAME



### Authorization

Repository File Pool Administrator

### Purpose

Use the FILEPOOL RENAME administrator command to rename an SFS or BFS file space to a new file space name.

To determine whether a user is enrolled in the file pool, use the CMS QUERY ENROLL command (see [z/VM: CMS Commands and Utilities Reference](#)).

### Operands

#### *filespaceid1*

is the existing file space being renamed.

For SFS file spaces, the directory structure, files, migrated files, authorities, and aliases of the file space are not affected; they are transferred from *filespaceid1* to *filespaceid2*. Authorities granted to, and locks created by *filespaceid1* are transferred to *filespaceid2*.

For BFS file spaces (created using the BFS option on ENROLL USER) the directory structure and all objects of the file space are not affected; they are transferred from *filespaceid1* to *filespaceid2*. Locks created by *filespaceid1* are transferred to *filespaceid2*.

For both types of file spaces, authorities granted through the use of an External Security Manager (ESM) are not transferred.

A nickname cannot be used for *filespaceid1*.

*filespaceid1* does not have to be explicitly enrolled in the file pool, but there must be some evidence it is a user of the file pool, such as its having explicit authorizations or holding current locks.

#### *filespaceid2*

is the new file space name. *filespaceid2* cannot already be enrolled in the file pool.

A nickname cannot be used for *filespaceid2*.

Do not rename to a user ID that begins with a plus (+) or a minus (-) or that contains a colon (:), a period (.), or a slash (/). These characters are used as separator characters in SFS directory IDs, of which the file space ID is a part. Also, BFS file space IDs may not contain the slash (/) or X'00' characters.

#### *filepoolid*

#### *filepoolid:*

identifies the file pool. The default is the default file pool ID for your virtual machine.

## Usage Notes

### Common Notes for SFS and BFS

1. This command requires the MAINT 193 disk be accessed.
2. An error message is issued if the file pool server does not support the FILEPOOL RENAME command.
3. Explicit locks, created on SFS files and directories or on the Byte File System (BFS) files in the file space by the CREATE LOCK command or the DMSCRLOC (SFS Create Lock) CSL routine, do not prevent the renaming of a file space.
4. The file pool server tries to obtain exclusive locks (whose owner is the function RENAME) on the file space and its storage group. If the locks are not obtained, the command is unsuccessful.
5. The FILEPOOL RENAME command is unsuccessful if a directory control directory owned by *userid1* is accessed R/W by any other user. The QUERY ACCESSORS command shows the user that currently has the directory accessed (see [“QUERY ACCESSORS \(SFS only\)”](#) on page 547).

Renaming is not affected by a directory control directory accessed R/O by another user. However, the user does not see the new owner’s name until he releases all accesses to the directory and then accesses it again.

6. When the renaming of a file space is unsuccessful, remove any locks by using the FUNCTION RENAME parameter of the
  - FILEPOOL ENABLE FILESPACE administrator command
  - The DMSNAFS (Enable File Space) CSL routine
  - Or the ENABLE operator command.

Do this only when FILEPOOL RENAME cannot finish and the locks need to be removed in order to recover.

DMSNAFS is described in *z/VM: CMS Callable Services Reference*.

7. Certain commands—the DISABLE GROUP command, FILEPOOL DISABLE GROUP command, and the DMSDISSG - Disable Storage Group routine—are incompatible with the FILEPOOL RENAME command: they are unsuccessful if they are entered while a file space is being renamed, and FILEPOOL RENAME is unsuccessful if the storage group has been locked by one of them. The file pool administrator can use the DMSNASG - Enable Storage Group CSL routine or the FILEPOOL ENABLE GROUP command to unlock a storage group, and the file pool operator can use the ENABLE command.
8. It is a good idea to back up user data before using this command.
9. To rename an administrator: (1) revoke administrator authority from the old user ID; (2) do the renaming; and (3) grant administrator authority to the new user ID. A warning message is issued if an administrator is renamed by this command. For further information on administrator authority see [“Administration Authority”](#) on page 139.
10. Locks held by *filespaceid2* are deleted by FILEPOOL RENAME. This can cause problems if *filespaceid2* tries to delete the locks or to perform operations that depend on their existence.
11. Because DFSMS/VM uses user IDs beginning with *DFSMS* you should avoid enrolling a user ID that begins with *DFSMS*. Also if you do enroll such a user, DFSMS/VM will not manage anything in that user's file space. See *z/VM: DFSMS/VM Storage Administration* for more information.
12. There can be no uncommitted work on the file space being renamed. The FILEPOOL RENAME command is unsuccessful if any object owned by *filespaceid1* is in use.
 

However, another user can have a file control directory accessed that is owned by *filespaceid1*. The information in the user machine that has the directory accessed is updated to reflect the new name.
13. A file space can be renamed regardless of any explicit locks on files and directories in the file space (locks created by the CREATE LOCK command or the DMSCRLOC - Create Lock CSL routine).
14. If an error occurs while an explicitly enrolled user is being renamed, some file pool changes can be committed. Process the command again using the same operands to complete the renaming; otherwise locks may remain in effect.

## FILEPOOL RENAME

When the command cannot be re-processed, release the locks and back off any changes to *filespaceid1*'s file space:

- a. Use the FUNCTION RENAME option on the file pool administrator command FILEPOOL ENABLE FILESPACE, SFS administrator CSL routine DMSENAFS, or the file pool operator command ENABLE FILESPACE to delete any locks on the file space and its storage group. (See usage note “6” on page 485). When the results indicate locks remain from the renaming, continue with step “14.b” on page 486.
- b. Use the FILEPOOL RESTORE command to restore the storage group. This is the only way to restore *filespaceid1*'s complete directory structure and objects.

**Note:** A storage group or a file space cannot be backed up until the locks are released and the storage group is restored. If the storage group is not restored, the file space being renamed may be left incomplete.

### Notes for SFS only

1. Users retain their authorities and aliases to the files and directories of a renamed file space. However, they must use the new user ID when accessing and manipulating objects in the file space.
2. Inconsistencies can occur if *filespaceid1* is accessing any of its directories when its file space is renamed. The directories remain accessed, but their names change and they are owned by *filespaceid2*. The upshot is that *filespaceid1* can no longer use the directories or objects in them without explicit authority.
3. When *filespaceid2* is accessing the *filespaceid1* file space before the renaming, after the renaming, CMS does not recognize *filespaceid2* as the owner of the directories it had accessed. This can cause errors, and it may be necessary for *filespaceid2* to re-IPL CMS.
4. When *filespaceid1* and *filespaceid2* have been granted different authorities to the same object, FILEPOOL RENAME gives *userid2* the higher authority.

### Notes for BFS only

1. Mounted file systems are not affected by FILEPOOL RENAME until the next IPL.
2. Users retain the same permissions to files and directories in the renamed BFS as they had to the original BFS; however, they must use the new name to MOUNT the BFS. In addition, symbolic links to objects in the renamed BFS must change to use the new name.

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warnings issued.

**8 or greater**

Unusual ending. The return code is the number of the associated message.

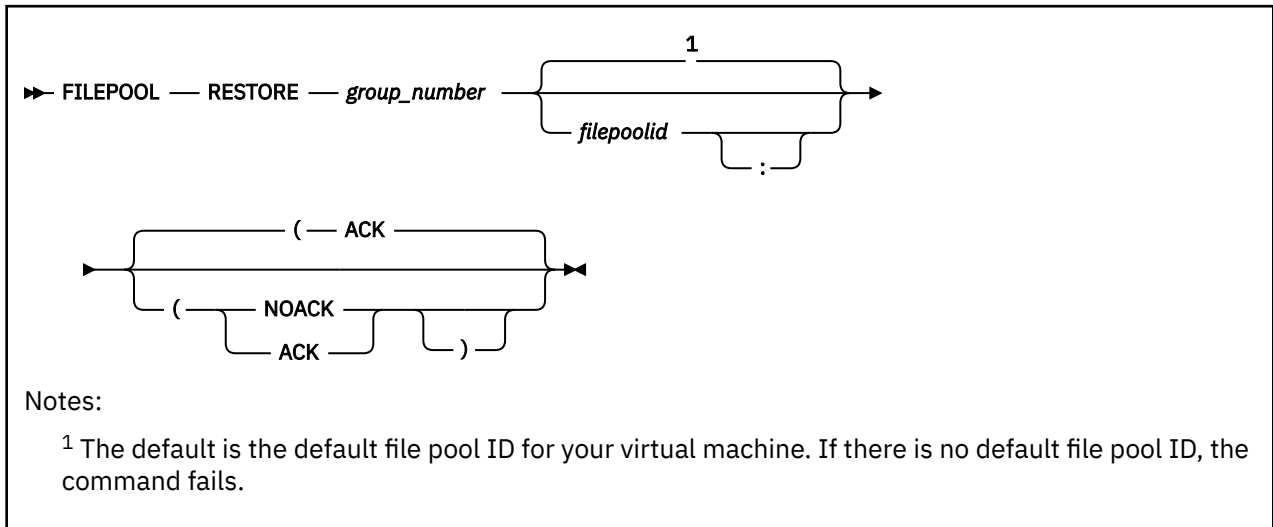
The FILEPOOL RENAME command can display more than one error message. Generally, the first error message describes the immediate cause of the error. The first message may be followed by additional error messages that further describe the error.

Messages:

- DMS149E Userid *filespaceid* not valid [RC=149]
- DMS1166E Userid *filespaceid* is already enrolled [RC=1166]
- DMS1167E Userid *filespaceid* is not enrolled [RC=1167]

- DMS1173E Userid *filespaceid* can not be renamed because the file space is currently in use [RC=1173]
- DMS1215E File space or storage group already locked in {EXCLUSIVE | SHARE} mode by another user. Rename of userid was unsuccessful. [RC=1215]
- DMS2023E File pool *filepoolid* does not support the requested FILEPOOL RENAME command [RC=2023]
- DMS2531E Rename of userid *filespaceid* was partially successful. Re-process the command with the same operands to complete the request [RC=2513]
- DMS2532W An administrator has been renamed The new user ID *userid2* does not have administrator authority [RC=4]
- DMS2544W User ID *userid1* does not exist in file pool *filepoolid*. [RC=4]
- DMS3516E No *workunitids* currently available. [RC=3516]
- DMS3555E Error renaming *filespaceid* in file pool *filepoolid*. Reason code = *reason* [RC=3555]
- DMS3726W The new user ID *userid2* has administrator authority [RC=4]

## FILEPOOL RESTORE



### Authorization

Repository File Pool Administrator

### Purpose

Use the FILEPOOL RESTORE command to load the specified file pool storage group from a copy of the storage group data created by FILEPOOL BACKUP processing. The storage group and associated catalog information for all SFS file spaces and Byte File Systems (BFS) file spaces still associated with this storage group are restored to their status at the point the restore file was created. Note that this command can only be entered from a virtual machine that has file pool administration authority. For more information on the effects of restoring a user storage group, see [“Restoring User Data” on page 115](#).

Only storage groups containing user file data can be loaded using this command. Storage group 1, which contains catalog data, cannot be loaded using this command.

### Operands

#### *group\_number*

is the number of the storage group to be restored. It must be the same as the number of the storage group whose backup information is on the restore file.

#### *filepoolid*

#### *filepoolid:*

is the ID of the file pool in which the storage group to be restored resides. If omitted, the default file pool ID for your virtual machine is used.

### Options

#### **ACK**

acknowledge the start of each stage of the restore process.

#### **NOACK**

suppress acknowledgement messages.

### Usage Notes

1. This command requires the MAINT 193 disk be accessed.
2. While the request is running, the storage group being restored will not be available to any other user.



3. Users enrolled in the storage group being restored will not be able to connect to the file pool until the restore completes successfully. This applies to users connecting for access to both SFS and BFS file pool information.
4. The only valid device types for the input and output files (for example, when using FILEDEF) are tape and disk.
5. Prior to issuing the FILEPOOL RESTORE command, you must enter a FILEDEF command to define the input file that contains the backup data. This file must have been created by the execution of the FILEPOOL BACKUP command. Specify RESTORE as the ddname. For example, to define a tape input file, you might enter:

```
FILEDEF RESTORE TAP1 SL VOLID 111111
```

This defines a tape file at virtual address 181 using IBM standard labels. The tape volume id is 111111. The file is the first on the tape.

An example of a FILEDEF command that defines a disk input file is:

```
FILEDEF RESTORE DISK SG000003 SGBACKUP B
```

This defines a CMS disk file named *SG000003 SGBACKUP* on file mode B.

The input file must have a variable-blocked record format with a record length of 29024 and a block size of 29028 (RECFM VB LRECL 29024 BLKSIZE 29028). The RECFM, LRECL, and BLKSIZE values are specified internally. If you specify your own RECFM, LRECL, or BLKSIZE values on the FILEDEF command, they will be ignored.

For tape devices, if necessary, specify the label options and the tape device specifications such as number of tracks and density. You can also use a LABELDEF command for a variety of functions associated with tapes. For example, if the backup spans more than one tape and you want to have the FILEPOOL RESTORE command display which tape to mount next, first enter:

```
LABELDEF BACKUP VOLID ?
```

You will then be prompted for volume IDs of the backup tapes. When you later enter a FILEPOOL RESTORE command, the messages displayed will instruct you which volume to mount. For more information on how to establish volume IDs, see the TAPE command in [z/VM: CMS Commands and Utilities Reference](#). For more information about the FILEDEF and LABELDEF commands, see [z/VM: CMS Commands and Utilities Reference](#). For information about tape handling, see [z/VM: CMS User's Guide](#).

6. Before entering the FILEPOOL RESTORE command, you should consider entering SET FILEWAIT ON.
 

The FILEPOOL RESTORE command obtains an *exclusive disable lock* for the storage group being restored. An exclusive disable lock prevents all user access to the storage group.

If you do not enter a SET FILEWAIT ON command, the FILEPOOL RESTORE command stops if any other user is writing to or reading from the storage group when FILEPOOL RESTORE tries to lock it. When FILEWAIT is ON, the file pool server does not stop the FILEPOOL RESTORE command if it cannot immediately obtain the exclusive disable lock. Instead, the file pool server does not allow any new reading or writing activity to start. It waits for all activity to end and then gets the exclusive disable lock for FILEPOOL RESTORE. FILEPOOL RESTORE will then restore the storage group. When the restore completes, the server automatically relinquishes the exclusive disable lock.
7. The FILEPOOL RESTORE command will cause an error even with file wait if **any** DIRCONTROL directory in the storage group is accessed at all.
8. The FILEPOOL RESTORE command creates entries in the LASTING GLOBALV file. Therefore, the virtual machine in which the command processes must have a file mode A accessed in read/write mode. Either a minidisk, a directory in a storage group other than the one to be restored, or a directory in another file pool can be accessed as A.

If a read/write file mode A is not available, the FILEPOOL RESTORE will still run but recovery from catastrophic errors during the processing of the command will be more difficult or disruptive. A

*catastrophic error* of the FILEPOOL RESTORE command is any error that causes the usual ending logic to be avoided.

9. The FILEPOOL BACKUP command uses an application named DMS5PR to connect to the \*BLOCKIO facility using IUCV.
10. When you enter this command from your virtual machine, there can be no active IUCV or APPC sessions that were initiated using only CP IUCV or APPC interfaces. Any such non-CMS connections to IUCV will result in an error message DMS3545E being issued. SFS usage of APPC and IUCV cannot cause this problem.
11. If an external security manager is being used to provide data security, the virtual machine running the FILEPOOL RESTORE command must be authorized to write to all minidisks allocated to the storage group being restored.
12. If an external security manager is not being used, you should be prepared to supply the write passwords for all minidisks in the storage group. FILEPOOL RESTORE processing prompts you for these passwords before it begins restoring the data.

Instead of responding to the prompts as they occur, you can write an exec in which the passwords are placed in the program stack. Stack the passwords in order of the minidisk numbers (MDKnnnnn) one per line. Then enter the FILEPOOL RESTORE command. (The FILEPOOL RESTORE command is implemented as an exec.) However, placing minidisk passwords in a CMS file does reduce the level of security of the minidisks.

If minidisks have been added to the storage group since the restore file was created, you may supply passwords for only the minidisks in the restore file, or you may supply passwords for all the minidisks in the current user storage group. In the latter case, the extra minidisk passwords are ignored.

If the number of entries on the program stack is equal to the number of minidisks to be restored, the command assumes the entries are passwords. This implies the stack should not be used when running the FILEPOOL RESTORE command. If the number of entries on the program stack is equal to the number of minidisks to be restored, FILEPOOL RESTORE processing assumes the entries are passwords and tries to use them. (The entries are destroyed.)

If a password in the program stack is incorrect, the following message is displayed:

```
DMS3541R  Incorrect password supplied for minidisk
          MDKnnnnn at vdev.
          Enter '1' to retry or '0' to cancel.
```

You should respond '0' to cancel the operation. Otherwise, FILEPOOL RESTORE processing will use the next password on the stack, which is likely to be incorrect as well. Depending on your installation, repeated unsuccessful attempts to use a minidisk with incorrect passwords may cause the minidisk to be locked.

13. If more than one storage group in the same file pool are being restored, they must not be restored at the same time by issuing FILEPOOL RESTORE commands from two different administrator user IDs. Because the FILEPOOL RESTORE command updates the catalog data, multiple FILEPOOL RESTORE commands could interfere with each other, causing them to be unsuccessful. Always restore storage groups one at a time.
14. If the FILEPOOL RESTORE command processing is not successful, it may be impossible to reference the storage group. A message will be displayed at stopping if this is the case. After such an error, a successful restore must be accomplished before the storage group can be used again.
15. If file pool server processing is unsuccessful while the FILEPOOL RESTORE command is being processed, you will be given the option of waiting until the file pool server is available or of canceling the command. If you choose to cancel the FILEPOOL RESTORE command, the storage group will remain disabled. To enable the storage group, you must enter the FILEPOOL RESTORE command after the file pool server becomes available.
16. A *catastrophic error* is any error of the FILEPOOL RESTORE command that causes the usual stopping logic to be avoided. A catastrophic error occurred if you do not see a usual or unsuccessful completion message from the command.

If a catastrophic error has occurred, certain resources may be in an unusable state. If for some reason the FILEPOOL RESTORE command cannot be immediately rerun, you should enter the FILEPOOL CLEANUP command to release as many of these resources as possible.

Until you run FILEPOOL RESTORE again or FILEPOOL CLEANUP, the following conditions will exist:

- The storage group will be permanently disabled for read and write access. Users of the storage group will not be able to reference it. FILEPOOL CLEANUP processing will not enable the storage group. Only a successful restore will make it available.
- The minidisks may be linked in WRITE mode until the session ends.

Two items that may be permanently changed if the FILEPOOL RESTORE command catastrophically errors:

- DOS SVC Mode

The machine will be in OS SVC mode.

- TRACE I/O

If the user is tracing I/O using CP TRACE I/O, it is possible the trace will not be in effect.

#### 17. Recovery from Catastrophic Errors without GLOBALV

If, for any reason, the FILEPOOL RESTORE command was unsuccessful in its attempt to use CMS GLOBALV to record variables in the command that was unsuccessful (as indicated by an error message), you must detach all links to minidisks that belong to the storage group. You can do this simply by logging off the virtual machine in which the FILEPOOL RESTORE command was running. (You can log on again at any time.) Or, you can enter CP DETACH commands for the minidisks.

The FILEPOOL RESTORE command must be reentered to successfully restore the storage group before users will be allowed to reference it again. In this case, you should **not** manually enable the storage group (for example, using the ENABLE operator command), because the data within it is not valid.

#### 18. Because the FILEPOOL RESTORE command runs CP DETACH commands internally, if the user has a CP TRACE I/O command in effect, it will be temporarily turned off during the DETACH.

Because there is no way for the command to determine any options originally specified by the user, the trace will be reinstated without any. The virtual machine will stop and enter CP command mode when the next I/O interrupt occurs. At this point the user may reenter the TRACE I/O command with the correct options.

#### 19. Obsolete Backup Files

Between the times of backup and restore, you can make various modifications to the definition of the storage group. These modifications make the backup file obsolete.

Normally, you would make a new backup after any modification that altered the definition of the storage group. (This manual describes when such a backup is necessary.) The new backup would be used on any subsequent restore.

However, if it is necessary for you to use an obsolete or backup file that is not at the latest level for a restore, you should be aware of the following:

- Backups Made Prior to FILESERV GENERATE

To recover from some kinds of data losses, it is necessary to generate the file pool again and restore all user storage groups. **If you generate the file pool and modify a storage group in any way prior to restoring it, the storage group backup made before the generation is unusable.** Restoring the storage group from that backup would cause unpredictable results. The restore might succeed, but the data in the storage group or catalog data related to the storage group would be corrupt. Your only recourse would be to generate the file pool again and restore all the storage groups using backups made prior to the preceding generation.

- New File Pool ID

If the file pool ID was changed since the restore, you will be informed of the new and old IDs and will be asked if execution should continue. Because file pool IDs are seldom changed, this message means one of the following:

- You are trying to restore a backup file from the wrong file pool.
- You are doing a secret restore and have temporarily changed the file pool ID. If the displayed file pool IDs correspond to the secret file pool ID and the usual file pool ID, you can allow the process to continue.

- New Minidisks

If additional minidisks have been added to the storage group, the user will be warned they will not be restored and asked if processing should continue. Note that any data on these minidisks will be effectively erased when the catalog information for the storage group is restored.

- New Minidisk Address

If the virtual device number for a minidisk has changed, the user will be warned of the change and asked if processing should continue.

- Minidisk Size Changed

If the new size is greater than the old, processing will continue. If the new size is less than the old, the restore will end unusually.

- New File Space in Storage Group

The file space will be dropped from the storage group and all data (files and directories) in the storage group belonging to that user will be erased. The issuer of the command will be told if this occurs.

A new backup of the storage group should always be done as soon as new file spaces are added or moved to the group to avoid this possibility.

- File Space Moved to Different Storage Group

If a file space which was in the storage group when the backup file was created is currently in another storage group, that file space will not be restored to this storage group. The file space remains intact in the other storage group. The issuer of the command will be informed.

Note that for certain sequences of restore of user storage groups involving moved file spaces using backup files built before the move, it is possible these file spaces will be removed from the file pool. A backup of the new user storage group should always be done as soon as file spaces are moved into the storage group.

If restore **must** be done using a backup file built before the move, the new storage group should be restored first.

- File Space in Backup, but not in File Pool

If there is a file space in the backup file that is not in the file pool, the file space is restored to the file pool. No notification is given to the issuer of the FILEPOOL RESTORE command. This lets you recover file spaces inadvertently deleted. If you do not want the file space and its data to exist in the file pool, you can enter a DELETE USER command after the FILEPOOL RESTORE command completes.

This condition would occur for any file spaces renamed (FILEPOOL RENAME command) since the backup file was created. For any such file spaces, you can enter a FILEPOOL RENAME command after the FILEPOOL RESTORE command completes.

### 20. Acknowledgement Messages

Unless the user explicitly suppresses them with the NOACK option, the FILEPOOL RESTORE command will issue acknowledgement messages. These messages are issued at the start of each stage of the restore process and can be used to keep track of how far processing has progressed.

### 21. FILEPOOL RESTORE command processing generates a large amount of the file pool server log activity. This log activity may cause the server to start an unscheduled control data backup which

may require intervention by the server operator. For more information, see [“Control Data Backups Started Automatically”](#) on page 105.

22. The FILEPOOL RESTORE command is not supported for execution in the CMS batch facility.
23. The FILEPOOL LIST BACKUP command lists the contents of the backup file created by the FILEPOOL BACKUP command.
24. If the file pool is managed by DFSMS/VM, it may contain migrated files. The FILEPOOL RESTORE command automatically restores both the primary and the migrated data for the storage group.

When restoring a storage group which has migrated files in it, the following should be kept in mind:

- a. The virtual machine in which the FILEPOOL RESTORE command processes must have the following authorizations:
  - File pool administration authority for the file pool that contains the storage group being restored
  - File pool administration authority for the file pool that contains the migration level 1 directory for the storage group being restored
  - Read authority to the DFSMS/VM control file DGTVCNTL DATA in directory VMSYS:DFSMS.CONTROL
- b. While the FILEPOOL RESTORE command is executing, the file pool with the migration level 1 directory for the storage group being restored and the VMSYS file pool must also be available.
- c. The FILEPOOL RESTORE command requires certain CSL routines which reside in the DFSMS/VM-supplied library FSMINT CSLLIB. Therefore, an access to FSMINT CSLLIB is needed if you have migrated data. This can be accomplished by linking and accessing the disk that contains the DFSMS/VM product code.
- d. If you are using the Migration Level 2 capabilities of DFSMS/VM Level 221 (ML2 is defined in the DFSMS/VM control file), IBM Language Environment with the C Language option is required. The disks containing Language Environment must be accessed prior to invoking the FILEPOOL RESTORE command. If only ML1 is defined in the DFSMS/VM control file, Language Environment is not necessary.
- e. While restoring the migrated data, the FILEPOOL RESTORE command may display some messages issued by DFSMS/VM. Such messages are prefix with the characters **FSM** and are documented in *z/VM: DFSMS/VM Messages and Codes*.
- f. It is not recommended that the migration level 1 directory be used to store data other than DFSMS/VM data. Non-DFSMS/VM data may be lost during FILEPOOL RESTORE processing.
- g. If the migration level 1 directory does not exist it will be recreated by the FILEPOOL RESTORE processing. This will be done only if the file space that is to contain the migrated data exists. (This file space is specified in the DFSMS/VM control file. For more information see *z/VM: DFSMS/VM Storage Administration*.)
- h. If the FILEPOOL RESTORE process is unsuccessful during the restore of the migrated data, the storage group will be left disabled. You should correct the problem and rerun the command. If for some reason you are prevented from doing that, you can enable the storage group, using the ENABLE operator command or FILEPOOL ENABLE administrator command, and allow access to the primary SFS data. But by doing so, you may lose some of the migrated data.

**Attention:** You should never attempt to enable the storage group if the restore of the primary file pool data was not successful, that is, before message DMS3509I is displayed.

For detailed information about DFSMS/VM, migrated SFS files and managing storage groups, see *z/VM: DFSMS/VM Storage Administration* and *z/VM: DFSMS/VM Planning Guide*.

25. SFS authorizations reside with the files and directories being authorized. Therefore, all restored files and directories have the authorizations and authorization user IDs which existed when the backup file was created.
26. Byte File System (BFS) owner (User ID (UID) and Group ID (GID)) and permission information resides with the files and directories. Therefore, all restored files and directories have the owners and permission information which existed when the backup file was created.

## Messages and Return Codes

### Return Codes

**0**

Successful completion.

**4**

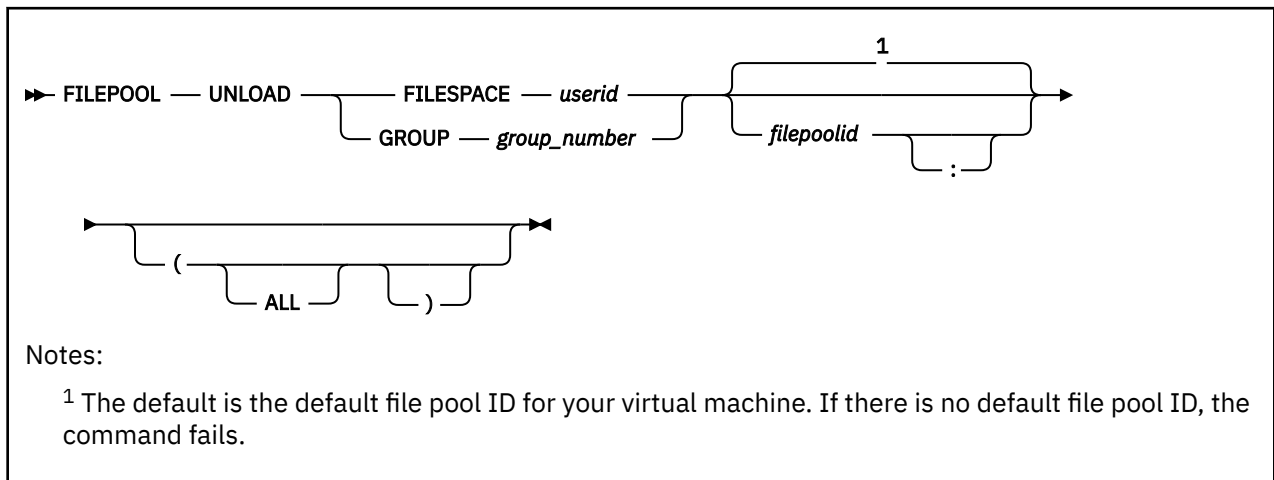
Successful completion, warning(s) issued.

**8 or greater**

Unusual stopping. The return code is the same as the message number of associated error message.

When there is an error condition, the FILEPOOL RESTORE command usually displays more than one error message. In most cases, the first error message displayed describes the immediate cause of the error. This message is often followed by other error messages that describe more about the error. The return code for this command is the number of the message that describes the immediate cause of the error.

## FILEPOOL UNLOAD



### Authorization

Repository File Pool Administrator

### Purpose

Use the FILEPOOL UNLOAD command to create a file that contains a copy of all data in an SFS or Byte File System (BFS) file space or storage group and all associated file pool catalog data. Then FILEPOOL RELOAD command can use the resultant file (from UNLOAD) to recreate the file space or storage group.

FILEPOOL UNLOAD in conjunction with FILEPOOL RELOAD can be used to recover data. The data can be in an SFS or BFS by specifying the reload to the same file pool and storage group or file space if user data loss occurs.

The same is true for moving data. An SFS or BFS file space or storage group can be moved to another storage group in the same or a different file pool. This can be done with the FILEPOOL UNLOAD command followed by a FILEPOOL RELOAD command.

### Operands

#### **FILESPACE *userid***

identifies the file space to be unloaded.

#### **GROUP *group\_number***

identifies the storage group to be unloaded.

#### ***filepoolid***

#### ***filepoolid:***

is the ID of the file pool in which the file space or storage group resides. If not specified, the default file pool ID for your virtual machine will be used.

### Options

#### **ALL**

If the *ALL* option is specified, all aliases that other users in the file pool have on the file space(s) being unloaded and all authorizations granted by other file pool users to the file space(s) being unloaded will be unloaded. If *ALL* is not specified, these will not be included.

## Usage Notes

1. The file created by the FILEPOOL UNLOAD command contains information on all the files and directories in the file space(s) as well as the following:
  - a. For SFS file spaces:
    - aliases defined in the file space(s) being unloaded
    - authorizations granted on the objects in the file space(s) being unloaded
    - if *ALL* is specified, aliases in other file spaces in the file pool that refer to base files in the file space(s) being unloaded
    - if *ALL* is specified, authorizations granted by other users in the file pool to the user ID identifying the file space(s) being unloaded
    - External Objects
    - Any data that was migrated by DFSMS/VM at the time of the unload
  - b. For BFS file spaces:
    - Block special files
    - Character special files
    - External Links
    - BFS Regular files
    - FIFOs
    - Symbolic Links
    - Any data that was migrated by DFSMS/VM at the time of the unload
2. The FILEPOOL UNLOAD command obtains a *share disable lock* for the storage group or file space being unloaded. This allows users to read from but not write to the locked storage group or file space.
3. The *ALL* option should not be specified if the UNLOAD is being performed to move a file space or storage group to a different file pool. The *ALL* option causes a performance degradation.
4. Conversely, the *ALL* option should be specified if you are using the FILEPOOL UNLOAD command to create a backup of a storage group or file space, or if you are using it to move a file space or storage group within a file pool.
5. Before entering the FILEPOOL UNLOAD command, you should consider issuing SET FILEWAIT ON. If you do not enter a SET FILEWAIT ON command, FILEPOOL UNLOAD terminates if any other user is reading from or writing to the file space or storage group when FILEPOOL UNLOAD tries to lock it. When FILEWAIT is ON, the file pool server does not terminate the FILEPOOL UNLOAD command if it cannot immediately obtain the lock. Instead, the file pool server does not let any new activity start. It waits for all activity to end and then gets the lock for FILEPOOL UNLOAD. FILEPOOL UNLOAD will then unload the file space. When the unload completes, the server automatically relinquishes the lock.
6. The only valid device types for the input and output files (for example, when using FILEDEF) are tape and disk.
7. Prior to issuing the FILEPOOL UNLOAD command, you must enter a FILEDEF command to define the output file that is to contain the unload data. Specify UNLOAD as the ddname. For example, to define a tape output file, you might enter:

```
filedef unload tap1 sl volid 111111
```

This defines a tape file at virtual address 181 using IBM standard labels. The tape volume id is 111111.

An example of a FILEDEF command that defines a disk output file is:

```
filedef unload disk userx unload b
```

This defines a CMS disk file named *USERX UNLOAD* on file mode B.



The RECFM, LRECL, and BLKSIZE values of the output file are specified internally. If you specify your own RECFM, LRECL, or BLKSIZE values on the FILEDEF command, they will be ignored.

The output file is created with a variable-blocked record format having a record length of 29024 and a blocksize of 29028 (RECFM VB LRECL 29024 BLKSIZE 29028).

For tape devices, if necessary, you should specify the label options and the tape device specifications such as number of tracks and density. A LABELDEF command may also be supplied if desired.

For more about the FILEDEF and LABELDEF commands, see *z/VM: CMS Commands and Utilities Reference* and *z/VM: CMS Application Development Guide for Assembler*. For more about tape handling, see *z/VM: CMS User's Guide*.

8. A catastrophic failure of the FILEPOOL UNLOAD command occurred if you do not see a normal or unsuccessful completion message from the command.

If a catastrophic failure has occurred, you may enter the command again. If you do not enter the command again, the following condition may exist:

- a. The file space may be disabled. To enable the file space, use the ENABLE operator command, the FILEPOOL ENABLE command, or the Enable File Space (DMSNAFS) CSL routine (see “ENABLE” on page 415, “FILEPOOL ENABLE” on page 447, and *z/VM: CMS Callable Services Reference*, for more information).
  - b. If the virtual machine was in DOS SVC mode at the time the FILEPOOL UNLOAD command was issued, it is possible that the machine will be left in OS SVC mode. You will need to either re-IPL or log off and log on again to reset to DOS SVC mode.
9. The FILEPOOL UNLOAD command is not supported for execution in the CMS batch facility.
  10. If the FILEPOOL UNLOAD command is issued to a server at a service level prior to VM/ESA Version 2 Release 1 message DMS3298E will be displayed, and the command will not be executed. Use the FILEPOOL BACKUP command for backing up these storage groups, or FILESERV MOVEUSER for moving users.
  11. The output of the FILEPOOL UNLOAD command may be used as input to the FILEPOOL RELOAD FILES command if you need to restore a subset of the files in an existing file space or storage group.
  12. If the file pool is managed by DFSMS/VM, it may contain migrated files. The FILEPOOL UNLOAD command automatically unloads both the primary and the migrated data for the file space being unloaded.

When unloading a file space which has migrated files in it, the following should be kept in mind:

- a. The virtual machine in which the FILEPOOL UNLOAD command executes must have the following authorizations:
  - SFS administration authority for the file pool that contains the file space being unloaded
  - SFS administration authority for the file pool that contains the migration level 1 directory for the file space being unloaded
  - Read authority to the DFSMS/VM control file DGTVCNTL DATA in directory VMSYS:DFSMS.CONTROL.
- b. The FILEPOOL UNLOAD command requires certain CSL routines which reside in the DFSMS/VM-supplied library FSMINT CSLLIB. Therefore, an access to FSMINT CSLLIB is needed if you have migrated data. This can be accomplished by linking and accessing the disk that contains the DFSMS/VM product code.
- c. While the FILEPOOL UNLOAD command is executing, the file pool with the migration level 1 directory for the file space being unloaded and the VMSYS file pool must be available.
- d. While unloading the migrated data, the FILEPOOL UNLOAD command may display some messages that are issued by DFSMS/VM. Such messages are prefixed with the characters **FSM** and are documented in *z/VM: DFSMS/VM Messages and Codes*.

For detailed information about DFSMS/VM, migrated SFS files and managing storage groups, see *z/VM: DFSMS/VM Storage Administration* and *z/VM: DFSMS/VM Planning Guide*.

13. BFS files containing more than  $2^{31}-1$  records (bytes) will not be supported by FILEPOOL UNLOAD processing. If such a file is encountered, FILEPOOL UNLOAD processing will stop.

### Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in *z/VM: CMS Commands and Utilities Reference*.

#### Return Codes:

**0**

Successful completion.

**4**

Successful completion, warnings entered.

**8 or greater**

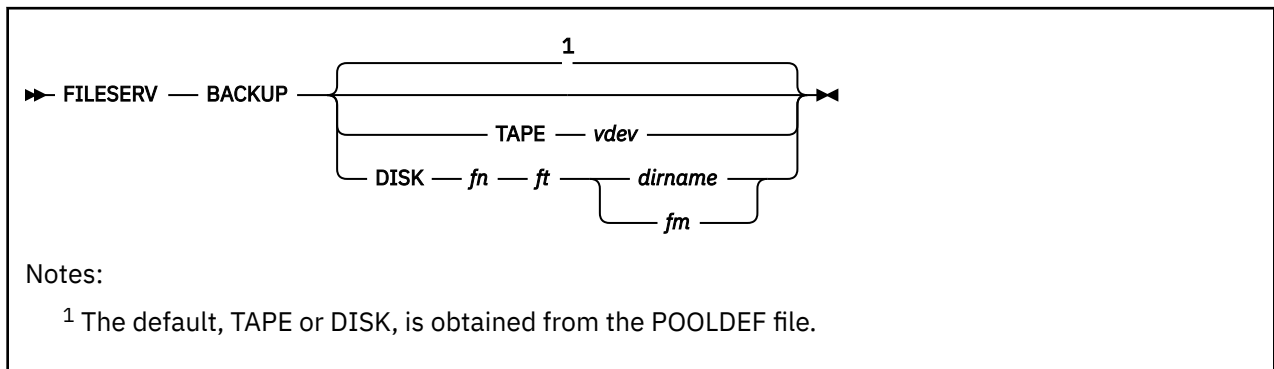
Unusual stopping. The return code is the number of the associated message.

Messages:

- DMS0386E Missing Operands
- DMS0649E Extraneous parameter(s)
- DMS1136E Unable to gain access to library *library\_name*
- DMS1223E There is no default file pool currently defined
- DMS1240E You are not authorized to connect to file pool *filepoolid*
- DMS3209E *Rtname* request failed. Return code = *returncode*. Reason code = *reasoncode*
- DMS3298E File pool *filepoolid* does not support the UNLOAD operand on the FILEPOOL command
- DMS3438E FILEPOOL UNLOAD unsuccessful
- DMS3438I FILEPOOL UNLOAD successful
- DMS3452E Invalid file pool catalog interface level, level = *level*
- DMS3511E Specified storage group number *nn* invalid
- DMS3512E Invalid option *option* specified.
- DMS3514E Action (*action*) invalid. Must be UNLOAD
- DMS3515E *parameter* is an invalid parameter
- DMS3516E No workunitids currently available
- DMS3518E File pool *filepoolid* is unavailable or unknown
- DMS3519E Storage group *nn* does not exist or you are not authorized to it
- DMS3519E File Space *filespaceid* does not exist or you are not authorized to it
- DMS3521E No FILEDEF specified for RELOAD file
- DMS3527E RELOAD file device type is not tape or disk
- DMS3528E CLOSE for RELOAD file failed
- DMS3529E Unrecoverable I/O error on RELOAD file
- DMS3531E Insufficient virtual storage
- DMS3532E This userid does not have administrator authority for file pool *filepoolid*
- DMS3557E Error on RELOAD file open
- DMS3561E Storage group *nn* in file pool *filepoolid* has no associated file spaces
- DMS3585E Cannot access storage group *nn* in file pool *filepoolid*. Conflicting lock outstanding.
- DMS3585E Cannot access file space *filespaceid* in file pool *filepoolid*. Conflicting lock outstanding.
- DMS3621E Error on *rtname*. Reason code = *reasoncode*

- DMS3628E Inconsistent catalog information
- DMS3636E Storage group *nn* in file pool *filepoolid* contains migrated data. DFSMS/VM encountered an error verifying the execution environment
- DMS3636E File space *filespaceid* in file pool *filepoolid* contains migrated data. DFSMS/VM encountered an error verifying the execution environment

## FILESERV BACKUP



### Authorization

Repository File Pool Server Machine

### Purpose

Use the FILESERV BACKUP command to start a server in dedicated maintenance mode to back up the *control data*. The *control data* includes the contents of the POOLDEF file, the control minidisk, and the catalog storage group (storage group 1). For more information on file pool recovery facilities, see [Chapter 7, “Recovery Procedures,”](#) on page 101.

### Operands

#### TAPE *vdev*

identifies the virtual device number *vdev* for a standard label (SL) tape file. The control data backup will be directed to the identified tape device.

#### DISK *fn ft dirname*

#### DISK *fn ft fm*

identifies a CMS file to which the control data backup will be directed. The file can reside on a minidisk or in another file pool. The *dirname* represents the fully qualified directory name of an SFS directory where the backup file is to be created. Note that *filepoolid:userid.n1.n2...n8* is the only allowed form of *dirname* in this case. If an assignment for the control data backup file already exists in the file pool server, it is temporarily replaced. The *fm* represents one of the following:

- The file mode of a minidisk accessed by the file pool server whose control data is being backed up.
- The file mode of an SFS directory accessed by the file pool server whose control data is being backed up.

### Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. For a list of supported virtual device numbers for tapes, see the description of the FILEDEF command in *z/VM: CMS Commands and Utilities Reference*.
3. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV BACKUP will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.
4. All messages are written to the server machine console.
5. For CRR recovery servers, IBM does not recommend doing backups of control data (that is, use the NOBACKUP startup parameter).

6. If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.
7. The FILESERV BACKUP command locates the file pool minidisks by using a POOLDEF file. The POOLDEF file it uses is the first file named *filepoolid* POOLDEF in its search order.  
The *filepoolid* POOLDEF file may reside on a CMS minidisk or it can reside in another file pool.
8. Backup data will be written to the file identified on the DDNAME=BACKUP record in the *filepoolid* POOLDEF file or to the file identified on the FILESERV BACKUP command itself.
9. The BACKUP startup parameter must be in effect for FILESERV BACKUP to complete successfully.
10. FILESERV BACKUP is not the only way a control data backup can be started. A backup will also be scheduled automatically when the file pool logs fill to 80% full, or may be scheduled by issuing the operator commands BACKUP or STOP BACKUP, or by issuing the administrator command FILEPOOL CONTROL BACKUP. The BACKUP startup parameter must be in effect for any control data backup to succeed.
11. When the backup file resides on a minidisk, FILESERV BACKUP processing uses a temporary file during the backup. This file is named \$\$TEMP \$BACKUP. If FILESERV BACKUP processing completes successfully, the file is erased. If FILESERV BACKUP processing fails and the backup file resides on a minidisk, see “Special Considerations for Control Data Backups to CMS File” on page 106 for the appropriate recovery actions.
12. You do not need to enter a FILEDEF for ddname=BACKUP. The server creates the proper environment for the backup processing.
13. If the command fails to complete successfully, it can be rerun. If the command fails and displays error messages, correct the problem and process the command again.
14. If the control backup is directed to tape, it is assumed the tape will be ready when the backup starts. If it is not, message DMS113S will be issued and control backup will stop.

## FILESERV CRRLOG

► FILESERV — CRRLOG — *vdev1* — *vdev2* ◄

### Authorization

CRR Server Machine

### Purpose

Use the FILESERV CRRLOG command to format and update the CRR log minidisks. It lets you reformat the CRR logs, change their sizes, or change their location.

If you are adding the CRR logs for the first time or reconfiguring the CRR logs, you will need to add or update the MDISK control statement for the CRR logs in the z/VM system directory entry for the server virtual machine before invoking this command.

**Note:** The FILESERV CRRLOG command is similar to the FILESERV LOG command. (See “FILESERV LOG” on page 523.) The difference is the FILESERV CRRLOG command formats CRR log minidisks and the FILESERV LOG command formats the SFS log minidisks.

**Attention:** Because FILESERV CRRLOG formats the CRR log minidisks, any existing CRR log information will be destroyed.

### Operands

#### *vdev1*

is the virtual device number of the first CRR log minidisk. It must be specified.

#### *vdev2*

is the virtual device number of the second CRR log minidisk. It must be specified.

**Note:** CMS FORMAT and RESERVE commands are entered for each CRR log minidisk when the startup parameter FORMAT is in effect. A blocksize of 4096 is used for the CRR log minidisk. If the startup parameter NOFORMAT is in effect, CMS FORMAT and RESERVE commands are not entered for the CRR log minidisk. However, the CRR log minidisks will still be initialized, and all log data will be lost as a result of the FILESERV CRRLOG command with NOFORMAT specified. The FORMAT option is required if the minidisks specified have not previously been formatted and reserved as CRR log minidisks.

### Usage Notes

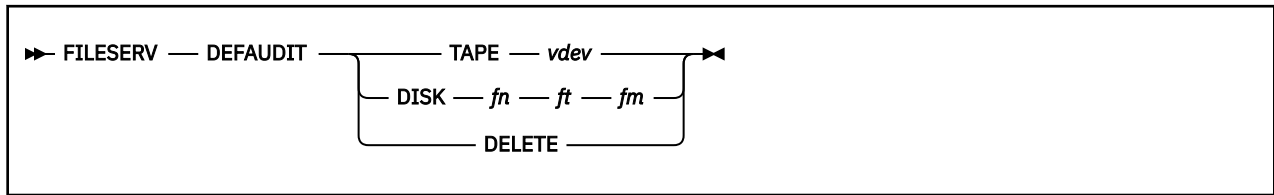
1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. All messages are written to the server machine console.
3. Both *vdev1* and *vdev2* must be the same size and device type but they must be on different physical devices.
4. The *vdev1* value specifies the virtual address of the first CRR log minidisk entry (DDNAME=CRR1) and the *vdev2* specifies the virtual address of the second CRR log minidisk entry (DDNAME=CRR2) in the *filepoolid* POOLDEF file.
5. This command can also be used to change the LU name of the CRR recovery server.
6. If the command fails to complete successfully, it can be rerun. If the command fails and displays error messages, correct the problem and process the command again.
7. If you are replacing only one log minidisk, there is no need to use the FILESERV CRRLOG command. Instead, follow the instructions on “[Replace One CRR Log Minidisk](#)” on page 328.

8. When defining a server to be an alternate CRR recovery server, use the FILESERV DEFCRRLOG command to get the CRR log definitions into the POOLDEF file of the alternate CRR recovery server.

**Attention:** Do not enter the FILESERV CRRLOG command because it formats the CRR logs and destroys any existing CRR log data. This means the newly defined alternate CRR recovery server cannot complete any logical units of work that were in resynchronization with the previous CRR recovery server. For more information about defining an alternate CRR recovery server, see [“Designate an Alternate CRR Recovery Server”](#) on page 318.

9. The FILESERV CRRLOG command should be preceded by the FILESERV DEFCRRLOG command, if you are:
  - Adding CRR logs to the server
  - Changing one or both CRR log virtual device numbers
10. The CRR and LUNAME *luname* startup parameters are required when the FILESERV CRRLOG command is used. The LUNAME *luname* startup parameter establishes the fully qualified LU name of the CRR recovery server. For more information about the LUNAME *luname* startup parameter, see [LUNAME](#).

## FILESERV DEFAUDIT



### Authorization

File Pool Server Machine

### Purpose

Use the FILESERV DEFAUDIT command to add, change, or delete the assignment of the security audit output file for a file pool server. Command processing updates the appropriate POOLDEF file but does not access the other file pool minidisks. For more information about auditing security, see [“Auditing Security”](#) on page 143.

### Operands

#### TAPE *vdev*

identifies the virtual device number (*vdev*) for a standard label (SL) tape file. Standard labels allow multivolume tape file support. The output of subsequent security audits will be directed to the identified tape device. If an assignment for the security audit output file already exists, it is replaced.

#### DISK *fn ft fm*

identifies a CMS file to which the output of subsequent security audits will be directed. The file can reside on minidisk or in another file pool. If an assignment for the security audit output file already exists, it is replaced.

#### DELETE

indicates the current assignment for the security audit output file should be eliminated.

### Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. This command only updates the POOLDEF file, it does not start server processing.
3. This usage note is applicable only if this is a file pool server; or if this is a CRR recovery server and you did not follow IBM's recommendation to not do control data backups.

After using the FILESERV DEFAUDIT command, you should enter the FILESERV BACKUP command to do a control data backup. This will ensure the updates just made to the *filepoolid* POOLDEF file are backed up. Then, the next time you do a control restore (for example, if you enter FILESERV START with the RESTORE startup parameter), you will get the current version of the POOLDEF file.

4. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV DEFAUDIT will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.
5. All messages are written to the server machine console.
6. If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.
7. The *filepoolid* POOLDEF file may reside on a CMS minidisk with other file pool definition files for other server machines.



8. This command can update the *filepoolid* POOLDEF file and should be run in the server machine that owns the file. If the command is run in another machine, conflicts with server processing may occur.
9. A temporary file named \$\$TEMP \$POOLDEF is created on the first R/W file mode in the CMS search order. The temporary file is erased if the command completes successfully.
10. If FILESERV DEFAUDIT is successful, the *filepoolid* POOLDEF file is updated. A record is added, replaced, or deleted as follows:
  - If a tape file is specified, the *filepoolid* POOLDEF file is updated with the following entry:

```
DDNAME=AUDIT TAPE VDEV=vdev
```

Any existing DDNAME=AUDIT record is deleted.

- If a disk file is specified, the *filepoolid* POOLDEF file is updated with the following entry:

```
DDNAME=AUDIT DISK FN=filename FT=filetype FM=filemode
```

Any existing DDNAME=AUDIT record is deleted.

- If DELETE is specified, the DDNAME=AUDIT record is deleted.

For example, if you enter:

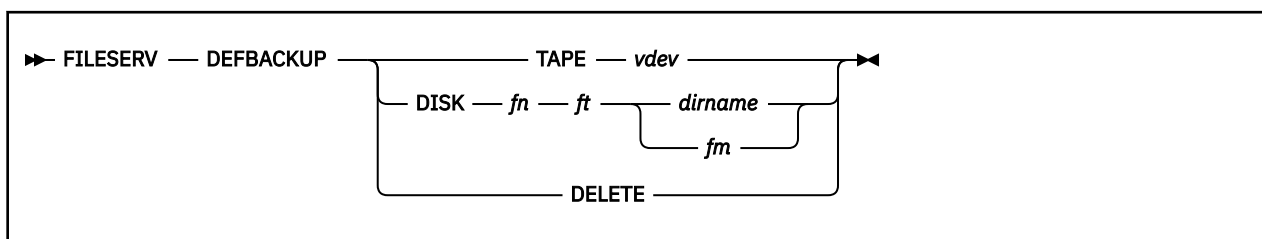
```
fileserv defaudit disk filepool audit a
```

the POOLDEF file will look something like this:

```
MAXUSERS=100
MAXDISKS=100
DDNAME=CONTROL          VDEV=301
DDNAME=LOG1             VDEV=302
DDNAME=LOG2             VDEV=303
DDNAME=MDK00001        VDEV=304      GROUP=1
DDNAME=MDK00002        VDEV=305      GROUP=2
DDNAME=BACKUP          DISK FN=FILEPOOL FT=BACKUP FM=B
DDNAME=AUDIT           DISK FN=FILEPOOL FT=AUDIT  FM=A
```

11. If the command does not complete successfully, it can be rerun. An error message is displayed. Correct the problem and process the command again.

## FILESERV DEFBACKUP



### Authorization

Repository File Pool Server Machine

### Purpose

Use the FILESERV DEFBACKUP command to add, change, or delete the assignment of the control data backup file for the file pool server. Command processing updates the appropriate POOLDEF file but does not access the other file pool minidisks. For more information on SFS and CRR recovery facilities, see Chapter 7, “Recovery Procedures,” on page 101.

### Operands

#### TAPE *vdev*

identifies the virtual device number (*vdev*) for a standard label (SL) tape file. All subsequent control data backups will be directed to the identified tape device. If an assignment for the control data backup file already exists, it is replaced.

#### DISK *fn ft dirname*

#### DISK *fn ft fm*

identifies a CMS file to which the control data backup will be directed. The file can reside on a minidisk or in another file pool. The *dirname* represents the fully qualified directory name of an SFS directory where the backup file is to be created. Note that *filepoolid:userid.n* (where you can repeat *n* up to 8 times, each one separated by a period) is the only allowed form of *dirname* in this case. If an assignment for the control data backup file already exists in the file pool server, it is temporarily replaced. The *fm* represents one of the following:

- The file mode of a minidisk accessed by the file pool server whose control data is being backed up.
- The file mode of an SFS directory accessed by the file pool server whose control data is being backed up.

#### DELETE

indicates the current assignments for the control data backup file should be eliminated.

### Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. For a list of supported virtual device numbers for tapes, see the description of the FILEDEF command in *z/VM: CMS Commands and Utilities Reference*.
3. This command only updates the POOLDEF file, it does not start server processing.
4. For CRR recovery servers, IBM does not recommend doing backups of control data (that is, use the NOBACKUP startup parameter).
5. This usage note is applicable only if this is a file pool server; or if this is a CRR recovery server and you did not follow IBM's recommendation to not do control data backups.

After using the FILESERV DEFBACKUP command, you should enter the FILESERV BACKUP command to do a control data backup. This will ensure the updates just made to the POOLDEF file are backed up. Then, the next time you do a control data restore (for example, if you enter FILESERV START with the RESTORE startup parameter), you will get the current version of the POOLDEF file.

6. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV DEFBACKUP will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.
7. All messages are written to the server machine console.
8. If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.
9. The *filepoolid* POOLDEF file can reside on a CMS minidisk with other file pool definition files for other server machines.
10. This command can update the *filepoolid* POOLDEF file and should be run in the server machine that owns the file. If the command is run in another machine, conflicts with server processing may occur.
11. You do not need to enter a FILEDEF for ddname=BACKUP. The server creates the proper environment for the backup processing.
12. A temporary file named \$\$TEMP \$POOLDEF is created on the first R/W file mode in the CMS search order. The file is erased if the command completes successfully.
13. Do not specify \$\$TEMP \$BACKUP as the backup file. When making a control data backup to a minidisk, the server uses \$\$TEMP \$BACKUP as a temporary file.
14. If FILESERV DEFBACKUP is successful, the first *filepoolid* POOLDEF file found in the CMS search order is updated with the new backup file information as follows:

- If a tape file is specified, the *filepoolid* POOLDEF file is updated with the following entry:

```
DDNAME=BACKUP TAPE VDEV=vdev
```

Any existing DDNAME=BACKUP entry is replaced and any existing BKDIRID and BKDIREXT entries are erased.

- If a disk file is specified, the *filepoolid* POOLDEF file is updated as follows:

If *dirid* is simply a file mode, the *filepoolid* POOLDEF file is updated with the following entry:

```
DDNAME=BACKUP DISK FN=filename FT=filetype FM=filemode
```

Any existing DDNAME=BACKUP entry is replaced and any existing BKDIRID and BKDIREXT entries are erased.

If *dirid* is a fully qualified directory name, the *filepoolid* POOLDEF file is updated with the following entries:

```
DDNAME=BACKUP DISK FN=filename FT=filetype FM=  
BKDIRID=filepoolid:userid.n1.n2...n8
```

Note that if the fully qualified directory name cannot fit in the BKDIRID entry, you may have one or more BKDIREXT entries.

Any existing DDNAME=BACKUP entry is replaced. The following rules apply:

- If a DDNAME=BACKUP DISK FN=*filename* FT=*filetype* FM=*filemode* control record is present in the POOLDEF file, and *filemode* is not blank, you cannot have BKDIRID and BKDIREXT control records.
- There can be at most one BKDIRID control record.
- If BKDIREXT control record(s) exist, they must follow a BKDIRID record.
- The file pool ID and user ID in the fully qualified directory name are required.

For example, if you enter:

```
fileserv defbackup disk filepool backup b
```

the POOLDEF file will look something like:

```
MAXUSERS=100
MAXDISKS=100
DDNAME=CONTROL          VDEV=301
DDNAME=LOG1             VDEV=302
DDNAME=LOG2             VDEV=303
DDNAME=MDK00001        VDEV=304      GROUP=1
DDNAME=MDK00002        VDEV=305      GROUP=2
DDNAME=BACKUP  DISK  FN=FILEPOOL  FT=BACKUP  FM=B
```

Or, if you enter (all on one line)

```
fileserv defbackup disk filepool backup
serverb:vmsysu.bkup
```

The POOLDEF file will look something like:

```
MAXUSERS=100
MAXDISKS=100
DDNAME=CONTROL          VDEV=301
DDNAME=LOG1             VDEV=302
DDNAME=LOG2             VDEV=303
DDNAME=MDK00001        VDEV=304      GROUP=1
DDNAME=MDK00002        VDEV=305      GROUP=2
DDNAME=BACKUP  DISK  FN=FILEPOOL  FT=BACKUP  FM=
BKDIRID=SERVERB:VMSYSU.BKUP
```

- If DELETE is specified, any existing DDNAME=BACKUP, BKDIRID and BKDIREXT records are deleted.

15. If you want to define the backup file in an SFS directory with a fully qualified name too long to type on the FILESERV DEFBACKUP command, you will have to edit the POOLDEF file and create the necessary BKDIRID and BKDIREXT control statements according to the rules described in the previous usage note.

For example, if you want to define your backup file with the file name FILEPOOL, file type BACKUP and create it in directory SERVERB:VMSYSU.DIRECTORYLEVEL01.DIRECTORYLEVEL02.DIRECTORYLEVEL03.DIRECTORYLEVEL04.DIRECTORYLEVEL05.DIRECTORYLEVEL06.DIRECTORYLEVEL07.DIRECTORYLEVEL08 your POOLDEF file should look something like:

```
MAXUSERS=100
MAXDISKS=100
DDNAME=CONTROL          VDEV=301
DDNAME=LOG1             VDEV=302
DDNAME=LOG2             VDEV=303
DDNAME=MDK00001        VDEV=304      GROUP=1
DDNAME=MDK00002        VDEV=305      GROUP=2
DDNAME=BACKUP  DISK  FN=FILEPOOL  FT=BACKUP  FM=
BKDIRID=SERVERB:VMSYSU.DIRECTORYLEVEL01
BKDIREXT=.DIRECTORYLEVEL02.DIRECTORYLEVEL03
BKDIREXT=.DIRECTORYLEVEL04.DIRECTORYLEVEL05
BKDIREXT=.DIRECTORYLEVEL06.DIRECTORYLEVEL07
BKDIREXT=.DIRECTORYLEVEL08
```

For more information about the BKDIRID and BKDIREXT control statements, see [“FILESERV GENERATE”](#) on page 513.

16. If the command does not complete successfully, it can be rerun. Error messages are displayed. Correct the problem and process the command again.
17. If the control backup is directed to tape, it is assumed the tape will be ready when the backup starts. If it is not, message DMS113S will be issued and control backup will stop.



10. If the CRR logs are currently defined in the POOLDEF file, and the virtual device numbers are different, the POOLDEF file definitions will be updated to reflect the specified virtual device numbers (*vdev1* and *vdev2*).
11. If the CRR logs are not currently defined in the POOLDEF file, they will be added.
12. When defining a server to be an alternate CRR recovery server, use the FILESERV DEFCRRLOG command to get the CRR log definitions into the POOLDEF file of the alternate CRR recovery server.

**Attention:** Do not enter the FILESERV CRRLOG command because it formats the CRR logs and destroys any existing CRR log data. This means the newly defined alternate CRR recovery server cannot complete any logical units of work that were in resynchronization with the previous CRR recovery server. For more information about defining an alternate CRR recovery server, see [“Designate an Alternate CRR Recovery Server”](#) on page 318.

## FILESERV FIXCENT

► FILESERV — FIXCENT ◄

### Authorization

File Pool Server Machine

### Purpose

Use the FILESERV FIXCENT command to set the century for all your SFS objects in the file pool server machine. This includes the four dates maintained by the server date of:

- last update
- creation
- last reference (DOLR)
- last changed (DOLC)

FILESERV FIXCENT should be used in the cases of:

- Migrations in the Year 2000 or later

Objects are created or updated in the YEAR 2000 and later, on a VM release that is not Year 2000 ready. After migrating to a VM Year 2000 ready release, the objects will appear to have a 4-digit year of 19xx. Note that this command does not need to be run if you migrate your SFS file pool before the year 2000.

- Testing of Year 2000

During testing, objects are created or updated and appear to have a 4-digit year of 20xx but actually the 4-digit should be 19xx.

In these cases, the FILESERV FIXCENT command can be used to set the correct century for all the objects in the file pool. The century is set according to a sliding window. For more information see usage note [“12” on page 512](#).

Here is a list of objects types that may be updated:

- SFS files
- directories
- aliases (external, revoked, and erased)
- SFS files in migrated status (these are files that appear to be in the file pool, but which actually reside in the DFSMS/VM auxiliary storage repository).

### Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV FIXCENT will use the first *serverid* DMSPARMS file it finds in the CMS search order. For more information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,” on page 339](#).
3. All messages are written to the server machine console.
4. The FILESERV FIXCENT command locates the file pool minidisks by using a POOLDEF file. The POOLDEF file it uses is the first file named *filepoolid* POOLDEF in its search order.

The *filepoolid* POOLDEF file may reside on a CMS minidisk with other file pool definition files for other server machines. It can also reside in another file pool.

5. This command must be run in the server machine that owns the file pool minidisks. If the command is run in another machine, conflicts with the server processing may occur.
6. If successful, the century of each date for each object is corrected. One commit is done at the end of a successful run. Either all the dates in the filepool are processed, or none of them are processed.
7. You should back up the control data before running FILESERV FIXCENT if a current backup is not available, you will be reminded by the following message during FILESERV FIXCENT initialization:

```
DMS3009R  A current control data backup is recommended before fixing
          dates in the file pool catalog data. Enter 1 to continue or
          0 to cancel the FILESERV FIXCENT
```

8. If you have any previous control data backups, you may want to create a new backup after this command completes successfully, because the old ones may not reflect the correct century of the file pool objects. Therefore, after this command completes successfully, the server will allow you to process only a FILESERV BACKUP, another FILESERV FIXCENT, or a FILESERV START with the RESTORE parameter in effect.
9. If the command does not complete successfully, it can be rerun. If the command still does not complete successfully, please call your systems support representative.
10. When FILESERV FIXCENT is completed it will issue message:

```
DMS3729I  n objects scanned; m objects updated; processing completed
```

When processing a large file pool, for every 10,000 objects you will receive a status message:

```
DMS3729I  n objects scanned; m objects updated; processing continuing
```

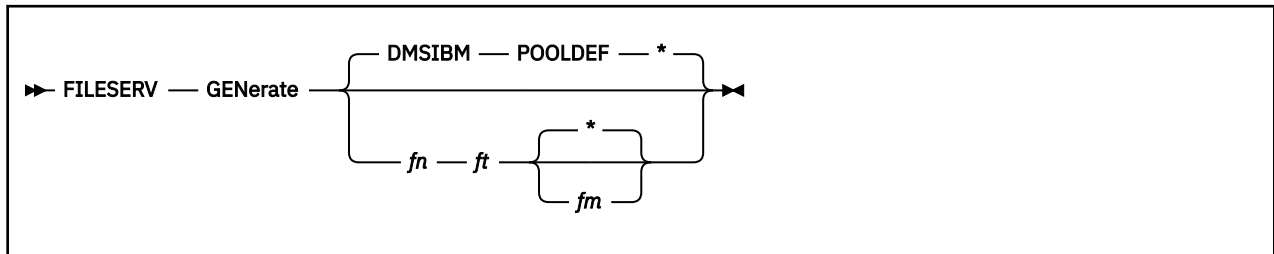
11. This command is not successful if any SFS logical units of work are found that have completed the first phase of the two-phase committing process. The logical units of work can be left in this condition by STOP IMMEDIATE or a server abend. This is also referred to as prepared-and-not-connected work. Generally, resynchronization resolves the prepared work when the server is executing in multi-user mode, making the prepared work consistent with the rest of the coordinated transaction. Before you enter FILESERV FIXCENT, restart the server, use QUERY PREPARED to verify the prepared work has been committed, then enter the STOP command.

If you cannot wait for resynchronization to be completed, for example, the communications line between the CRR recovery server and this file pool server is going to be down for a long time, you may manually resolve the prepared work by using the FORCE PREPARED command. See [“FORCE” on page 539](#) for more information.

12. The FILESERV FIXCENT command will set the century portion of the year according to a sliding window of:  $cy-50$ ,  $cy+49$ , where integer  $cy$  is the current year—that is, the year at the moment of the call. For example, if a 2-digit year of an object is 05, and the current year  $cy$  is 1997, the window range is 1947, 2046. The FILESERV FIXCENT command will set the century of the 2-digit year of 05 to be 20; thus, the 4-digit year of 2005.
13. For more information on a related command for minidisks, see the FIXCENT command in [z/VM: CMS Commands and Utilities Reference](#).



# FILESERV GENERATE



## Authorization

File Pool Server Machine

## Purpose

Use the FILESERV GENERATE command to define and initialize a new server file pool. This command completes the following major functions:

1. If the FORMAT startup parameter is in effect, FILESERV GENERATE enters CMS FORMAT and RESERVE commands for the minidisks assigned to the server's file pool (including the CRR log minidisks if the server is a CRR recovery server). If the NOFORMAT startup parameter is in effect, no CMS FORMAT and RESERVE commands are processed.
2. Initializes the server's file pool control, log, catalog space, and file space minidisks (including the CRR log minidisks if the server is a CRR recovery server).
3. Creates or replaces a file named *filepoolid* POOLDEF, which is referred to as the *POOLDEF* file. The POOLDEF file defines the server's file pool (including the CRR log minidisks if the server is a CRR recovery server). If it does not exist, FILESERV GENERATE processing creates the POOLDEF file using the first read/write file mode in the CMS search order. The POOLDEF file can reside on a minidisk or in another file pool.

To determine the *filepoolid* for the file pool, FILESERV GENERATE processing first tries to find a *serverid* DMSPARMS in the CMS search order. *serverid* is the user ID of the virtual machine in which the FILESERV GENERATE command is running. If it finds a *serverid* DMSPARMS file, it uses the value specified on the FILEPOOLID startup parameter as the file name of the POOLDEF file. If there is not a FILEPOOLID startup parameter, FILESERV GENERATE uses the *serverid* as the file name for the POOLDEF file. For more information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.

FILESERV GENERATE creates the POOLDEF file based on the contents of a control statement input file. You can specify a control statement input file you have created, or you can use an IBM-supplied file named DMSIBM POOLDEF.

For a step-by-step description of how to generate a file pool, see [Chapter 15, “Generating a File Pool and Server,”](#) on page 247.

## Operands

### *fn ft fm*

identifies the file name, file type, and file mode of the control statement input file. If *fn* and *ft* are specified but *fm* is not, FILESERV GENERATE uses the first file in the accessed search order having the specified file name and file type. The file must have a fixed record format of 80 bytes (RECFM F LRECL 80).

If you do not specify an input control statement file, FILESERV GENERATE uses the IBM-supplied DMSIBM POOLDEF file. The contents of the DMSIBM POOLDEF file are shown in [Figure 53](#) on page 514. DMSIBM POOLDEF resides on the MAINT machine's 193 minidisk.

```

MAXUSERS=1000
MAXDISKS=500
DDNAME=CONTROL          VDEV=301
DDNAME=LOG1             VDEV=302
DDNAME=LOG2             VDEV=303
DDNAME=BACKUP DISK     FN=FILEPOOL FT=BACKUP FM=*
DDNAME=MDK00001        VDEV=304   GROUP=1   BLOCKS=0
DDNAME=MDK00002        VDEV=305   GROUP=2   BLOCKS=0
DDNAME=CRR1            VDEV=306
DDNAME=CRR2            VDEV=307

```

Figure 53. Contents of the DMSIBM POOLDEF File

When *fn ft fm* are omitted, FILESERV GENERATE displays the DMSIBM POOLDEF file during its processing by invoking XEDIT for it. You can modify and save the file using any XEDIT commands. FILESERV GENERATE processing then continues using the modified file.

The format of each of the file pool generation control statements is described below.

### Control Statement File

To generate a file pool, you must provide input in the form of a file containing control statements. On the FILESERV GENERATE command you can specify a file you have created or you can allow FILESERV GENERATE to use the default DMSIBM POOLDEF file.

In any case, the file must contain fixed-length 80-byte records. Each record can contain only one complete control statement. Control statements cannot be continued on a second record. All information for a control statement must be contained in positions 1-72 of a single file record.

FILESERV GENERATE processing uses the control statement file to initialize the file pool components. This input file is used in creating the *filepoolid* POOLDEF file on the first read/write file mode in the search order. The resultant POOLDEF file is, in fact, merely a copy of the control statement input file.

The generation control statements establish maximum values for a file pool and assign minidisks to storage groups within the file pool.

Control statement file records that begin with an asterisk or contain all blanks in positions 1-72 will be ignored. Following are descriptions of the control statements.

```
MAXUSERS maxusers
```

The MAXUSERS control statement specifies the estimated maximum number of file pool users who will be individually authorized to create objects (such as directories, files, SFS aliases or Byte File System (BFS) links) in a file pool. FILESERV GENERATE (and FILESERV REGENERATE) processing use this value to compute the maximum logical size of the catalog space. This value computes how much space is needed on the control minidisk to represent logical catalog space in the file pool. For more information on MAXUSERS, see [Chapter 15, “Generating a File Pool and Server,” on page 247](#). Supporting more users than the value specified may require a regeneration of the file pool to expand the file pool control minidisk.

#### Note:

1. The MAXUSERS control statement must be specified.
2. A minimum value of 5 and a maximum value of 32767 is allowed for *maxusers*.
3. For most file pools, specifying a value of 1000 for *maxusers* should be adequate.
4. For CRR recovery server file pools, specify a value of 5 for *maxusers*.
5. Only one MAXUSERS control statement is allowed in the file.

```
MAXDISKS maxdisks
```

The MAXDISKS control statement defines the maximum number of minidisks and storage groups that can be used to contain catalog data and user data. The file pool control minidisk and log minidisks are not included in this limit, nor are any minidisks that contain the POOLDEF file, control data backup file, or the security audit trace file. To exceed the MAXDISKS value, you must regenerate the file pool.

**Note:**

1. **The MAXDISKS control statement must be specified.**
2. A minimum value of 2 and a maximum value of 32767 is allowed for *maxdisks*.
3. For most file pools, specifying a value of 500 for *maxdisks* should be adequate.
4. For CRR recovery server file pools, specify a value of 2 for *maxdisks*.
5. Only one MAXDISKS control statement is allowed in the file.
6. The MAXDISKS value also establishes the upper limit for the storage group identifier that can be specified in the FILESERV GENERATE, FILESERV MINIDISK, and FILEPOOL MINIDISK control statements. If, for instance, you specify 50 for *maxdisks*, the highest storage group identifier you can use is also 50.

```
DDNAME=CONTROL  VDEV=vdev
```

The DDNAME=CONTROL statement defines the virtual device number (*vdev*) of the server's file pool control minidisk. If the FORMAT startup parameter is in effect when you enter FILESERV GENERATE, the CMS FORMAT and RESERVE commands are entered for the control minidisk. FILESERV GENERATE processing uses a blocksize of 512 for the minidisk. If the startup parameter NOFORMAT is in effect, CMS FORMAT and RESERVE commands are not entered for the minidisk. (FILESERV GENERATE assumes the CMS FORMAT and RESERVE commands have already been entered for the minidisk.)

**The DDNAME=CONTROL control statement must be specified.**

```
DDNAME=LOG1  VDEV=vdev
```

The DDNAME=LOG1 control statement defines the virtual device number (*vdev*) of the server's first file pool log minidisk. If the FORMAT startup parameter is in effect when you enter FILESERV GENERATE, the CMS FORMAT and RESERVE commands are entered for the log minidisk. FILESERV GENERATE processing uses a blocksize of 4096 for the minidisk. If the startup parameter NOFORMAT is in effect, CMS FORMAT and RESERVE commands are not entered for the minidisk. (FILESERV GENERATE assumes the CMS FORMAT and RESERVE commands have already been entered for the minidisk.)

**The DDNAME=LOG1 control statement must be specified.**

```
DDNAME=LOG2  VDEV=vdev
```

The DDNAME=LOG2 control statement defines the virtual device number (*vdev*) of the server's second SFS file pool log minidisk. If the FORMAT startup parameter is in effect when you enter FILESERV GENERATE, CMS FORMAT and RESERVE commands are entered for the log minidisk. FILESERV GENERATE processing uses a blocksize of 4096 for the minidisk. If the startup parameter NOFORMAT is in effect, CMS FORMAT and RESERVE commands are not entered for the minidisk. (FILESERV GENERATE assumes CMS FORMAT and RESERVE commands have already been entered for the minidisk.)

**The DDNAME=LOG2 control statement must be specified.**

```

▶▶ DDNAME=BACKUP ——— TAPE — VDEV= — vdev —————▶▶
    |
    |—— DISK — FN=fn — FT=ft — FM=fm —————▶▶
  
```

The DDNAME=BACKUP control statement defines the file to be used for backups of the file pool control data. **The DDNAME=BACKUP control statement is optional.** You should specify it if you intend to use SFS facilities to back up the control data.

**TAPE VDEV=vdev**

identifies the virtual device number (*vdev*) for a standard label (SL) tape file.

**Note:** For a list of supported virtual device numbers for tapes, see the description of the FILEDEF command in *z/VM: CMS Commands and Utilities Reference*

**DISK FN=fn FT=ft FM=fm**

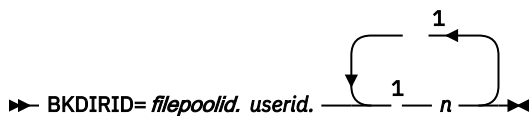
identifies the CMS file to be used as the backup file for the file pool control data. The file can reside on either a minidisk or in another file pool. If the backup file resides on a minidisk, it must be accessed in read/write (R/W) mode. If the backup file resides in another file pool, the directory may or may not be accessed. If the directory is not accessed, file mode must be blank (FM= ) and the POOLDEF must have a BKDIRID and, if needed, BKDIREXT control statements. Note that file mode \* is not valid for disk backup.

If you do not intend to backup the control data using SFS facilities, do not include the DDNAME=BACKUP statement in the file. When the DDNAME=BACKUP control statement is omitted, no corresponding record is created in the POOLDEF file. If you omit the DDNAME=BACKUP control statement, remember to add NOBACKUP to the *serverid* DMSPARMS file. BACKUP is the default.

Generally, if this is a dedicated repository file pool server or a server that is both SFS and CRR (**not recommended**), or a dedicated FIFO server, you would probably want to include the DDNAME=BACKUP control statement and also use the BACKUP and RESTORE startup parameters.

But, if this is a dedicated CRR recovery server or a dedicated FIFO server, you should delete the DDNAME=BACKUP control statement and use the NOBACKUP and NORESTORE startup parameters.

For more guidance on backup and restore of file pool control data, see [Chapter 7, “Recovery Procedures,”](#) on page 101.

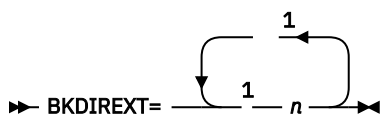


Notes:

<sup>1</sup> Specify *n* up to 8 times.

The BKDIRID control statement defines the fully qualified directory name of the SFS directory where the backup file is to be created. The file pool ID and user ID parameters are required. Because a fully qualified directory name can be up to 153 characters long, it may not be possible to store the entire dirname in the BKDIRID statement. In this case, it will be necessary to use one or more BKDIREXT statements as continuation lines. To derive the fully qualified directory name, server processing concatenates the information in the BKDIRID statement with the information in all BKDIREXT statements in the order in which they are found in the POOLDEF file.

**The BKDIRID control statement is optional.** If it is present, a DDNAME=BACKUP statement must also be present and its file mode value must be blank (FM= ). There can be at most one BKDIRID statement in the POOLDEF file and it cannot be preceded by BKDIREXT statements.



Notes:

<sup>1</sup> Specify *n* up to 8 times.

BKDIREXT control statements are used as continuation lines for the BKDIRID statement when the fully qualified directory name of the SFS directory for the backup file is too long and cannot fit in the BKDIRID statement alone. There can be more than one BKDIREXT control statements. To derive the fully qualified directory name, server processing concatenates the information in the BKDIRID statement with the information in all BKDIREXT statements in the order in which they are found in the POOLDEF file.

**BKDIREXT control statements are optional.** If they are present in the POOLDEF file, they must follow a BKDIRID statement.

```
➔ DDNAME=MDK nnnnn — VDEV= vdev — GROUP= grpnum — BLOCKS= blocks ➔
```

The DDNAME=MDK*nnnnn* control statement defines a minidisk to be allocated to a file pool storage group. **At least two of these DDNAME=MDK*nnnnn* control statements must be specified.** One is needed to allocate a minidisk to storage group 1, which will contain the file pool catalogs. A second is needed to allocate a minidisk to storage group 2, which will contain user data. You can specify additional minidisks if you wish.

For a dedicated file pool server, you would probably need to include additional DDNAME=MDK*nnnnn* control statements to allocate additional minidisks to contain user data.

For a dedicated CRR recovery server, you should only need the two DDNAME=MDK*nnnnn* control statements to allocate the minidisks needed for the file pool catalogs and user data.

When the startup parameter FORMAT is in effect, FILESERV GENERATE processing enters CMS FORMAT and RESERVE commands for the minidisks. A block size of 4096 is used. When the startup parameter NOFORMAT is in effect, the CMS FORMAT and RESERVE commands are not entered for the minidisks. (FILESERV GENERATE assumes the CMS FORMAT and RESERVE commands have already been entered for them.)

The following is a description of each control statement parameter:

#### **DDNAME=MDK*nnnnn***

FILESERV GENERATE processing creates a record in the POOLDEF file having the same MDK*nnnnn* you specify here. When you later enter other FILESERV commands for the file pool, the commands will build and process a FILEDEF command for the minidisk using MDK: DMSC6FGN

You can assign *nnnnn* a value ranging from 1 through 32767 and it must be assigned in ascending consecutive numeric sequence. For example, it is an error to use DDNAME=MDK00005 before DDNAME=MDK00004 is used. DDNAME=MDK00005 must be the next DDNAME used after DDNAME=MDK00004.

#### **VDEV=*vdev***

identifies the virtual device number (*vdev*) for the minidisk being added to the file pool.

#### **GROUP=*grpnum***

identifies the number of the storage group to which the minidisk is being added. The storage group may or may not already exist. A new storage group is defined simply by allocating a minidisk to a previously unused number.

You can assign *grpnum* a value ranging from 1 to 32767 but it cannot exceed the MAXDISKS value specified for the file pool. GROUP=1 must be used for the file pool catalog space. GROUP=2 must be used for the first storage group to contain user data.

Additional minidisks can be assigned to storage groups 3 through the value specified for MAXDISKS, all of which are used for user data. (The MAXDISKS value is the upper limit on storage group numbers.) The storage group numbers do not need to be assigned in ascending consecutive numeric sequence. For example, GROUP=5 can be assigned before GROUP=4.

#### **BLOCKS=*blocks***

identifies the number of 4KB blocks to allocate during GENERATE. The server handles this parameter automatically; it is not necessary for you to specify it if you are creating the POOLDEF file yourself.

During the FILESERV GENERATE process, the BLOCKS parameter is handled as follows:

## FILESERV GENERATE

- If the BLOCKS parameter is missing, or is set to zero, the server sets BLOCKS equal to the maximum number of blocks available on the minidisk and adds the BLOCKS parameter to the POOLDEF file, or modifies the existing BLOCKS value in the file, respectively. This is usually the case during the very first GENERATE for a file pool.
- If the value of the BLOCKS parameter is less than or equal to the maximum number of blocks available on the minidisk, the BLOCKS value in the POOLDEF file remains unchanged, and the server allocates to the storage group the number of blocks specified by the BLOCKS Parameter. The remainder of the blocks on the minidisk are not used. This is usually the case when a previously generated file pool is generated again, for example to restore data after a storage group DASD has been physically changed.
- If the BLOCKS parameter is less than zero or greater than the maximum number of blocks available on the minidisk, the GENERATE process will be unsuccessful.

If you modify the POOLDEF file, you should not change the value of the BLOCKS parameter, if it is present.

```
DDNAME=CRR1  VDEV=vdev
```

The DDNAME=CRR1 control statement defines the virtual device number (*vdev*) of the CRR recovery server's first CRR log minidisk. If the FORMAT startup parameter is in effect when you enter FILESERV GENERATE, the CMS FORMAT and RESERVE commands are entered for the CRR log minidisk. FILESERV GENERATE processing uses a blocksize of 4096 for the minidisk. If the startup parameter NOFORMAT is in effect, CMS FORMAT and RESERVE commands are not entered for the minidisk. (FILESERV GENERATE with NOFORMAT in effect assumes the CMS FORMAT and RESERVE commands have already been entered for the minidisk.)

**The DDNAME=CRR1 control statement must be specified if generating a CRR recovery server. If generating a dedicated SFS file pool server, do not include this control statement in the POOLDEF file.**

```
DDNAME=CRR2  VDEV=vdev
```

The DDNAME=CRR2 control statement defines the virtual device number (*vdev*) of the CRR recovery server's second CRR log minidisk. If the FORMAT startup parameter is in effect when you enter FILESERV GENERATE, the CMS FORMAT and RESERVE commands are entered for the CRR log minidisk. FILESERV GENERATE processing uses a blocksize of 4096 for the minidisk. If the startup parameter NOFORMAT is in effect, CMS FORMAT and RESERVE commands are not entered for the minidisk. (FILESERV GENERATE with NOFORMAT in effect assumes the CMS FORMAT and RESERVE commands have already been entered for the minidisk.)

**The DDNAME=CRR2 control statement must be specified if generating a CRR recovery server. If generating a dedicated SFS file pool server, do not include this control statement in the POOLDEF file.**

### Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV GENERATE will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, "File Pool Server Startup Parameters,"](#) on page 339.
3. All messages are written to the server machine console.
4. For CRR recovery servers, IBM does not recommend doing backups of control data (that is, use the NOBACKUP startup parameter).

- If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.

You should choose the *filepoolid* carefully. It is the name by which users will refer to the file pool. For a description of this startup parameter, see [FILEPOOLID](#).

- The FILESERV GENERATE command either creates or replaces a POOLDEF file. The POOLDEF file it uses is the first file named *filepoolid* POOLDEF in its search order.

The *filepoolid* POOLDEF file can reside on a CMS minidisk or it can reside in another file pool.

- If you do not specify a control statement input file, FILESERV GENERATE calls XEDIT during its processing to display the DMSIBM POOLDEF file. You may change the file and replace it using the FILE subcommand of XEDIT. If you plan to use this approach, be sure to record the virtual device numbers specified on the MDISK statements for the file pool minidisks and have them ready when you enter FILESERV GENERATE.
- File pool components are initialized, and the *filepoolid* POOLDEF file is created (or replaced) with the contents of the control statement input file.

Other FILESERV commands use the POOLDEF file to identify the file pool minidisks. Other FILESERV commands also update the POOLDEF file.

The following is an example of the contents of a POOLDEF file for a dedicated SFS file pool server immediately after a FILESERV GENERATE command completes:

```
MAXUSERS=1000
MAXDISKS=500
DDNAME=CONTROL      VDEV=301
DDNAME=LOG1         VDEV=302
DDNAME=LOG2         VDEV=303
DDNAME=BACKUP DISK FN=FILEPOOL FT=BACKUP FM=
BKDIRID=SERVERB :VMSYSU.BKUP

DDNAME=MDK00001     VDEV=304   GROUP=1   BLOCKS=292
DDNAME=MDK00002     VDEV=305   GROUP=2   BLOCKS=292
DDNAME=MDK00003     VDEV=306   GROUP=2   BLOCKS=292
DDNAME=MDK00004     VDEV=307   GROUP=2   BLOCKS=292
DDNAME=MDK00005     VDEV=308   GROUP=3   BLOCKS=292
DDNAME=MDK00006     VDEV=309   GROUP=3   BLOCKS=292
DDNAME=MDK00007     VDEV=30A   GROUP=3   BLOCKS=292
```

The following is an example of the contents of a POOLDEF file for a dedicated CRR recovery server immediately after a FILESERV GENERATE command completes:

```
MAXUSERS=5
MAXDISKS=2
DDNAME=CONTROL      VDEV=301
DDNAME=LOG1         VDEV=302
DDNAME=LOG2         VDEV=303
DDNAME=MDK00001     VDEV=304   GROUP=1   BLOCKS=292
DDNAME=MDK00002     VDEV=305   GROUP=2   BLOCKS=292
DDNAME=CRR1         VDEV=306
DDNAME=CRR2         VDEV=307
```

**Note:** IBM recommends you use separate servers dedicated to repository and CRR functions. A dedicated repository file pool server (VMSERVS) and a dedicated CRR recovery server (VMSEVR) can optionally be installed during the z/VM installation process. Or, you can generate your own repository file pool pool server and CRR recovery server in a post-installation procedure. For more information about generating your own servers, see [Chapter 15, “Generating a File Pool and Server,”](#) on page 247.

- The POOLDEF file is created or replaced and FILESERV GENERATE processing issues CMS FORMAT and RESERVE commands for the file pool block I/O minidisks (control minidisk, log minidisks, and all minidisks allocated to storage groups.) It issues the FORMAT and RESERVE commands only if the FORMAT startup parameter is in effect. The RESERVE command creates CMS-like block I/O files on the minidisks. These files are not the same as usual CMS files. **Never use CMS file manipulation commands (such as ERASE or EXECIO) against files residing on the file pool block I/O minidisks, as these commands may render the file pool useless.**

## FILESERV GENERATE

10. A security audit trace (AUDIT) file can later be defined for a server using the FILESERV DEFAUDIT command.
11. If successful, the *filepoolid* POOLDEF file is created on the first read/write file mode if one does not already exist. If a POOLDEF file already exists, it is replaced.
12. The following temporary files are created on the first read/write file mode in the CMS search order:
  - \$\$TEMP \$\$INPUT
  - \$\$TEMP \$POOLDEF
  - \$\$TEMP \$\$UPDATEIf processing is successful, the temporary files are erased.
13. If the command does not complete successfully, it can be rerun. If the command is unsuccessful and displays error messages, correct the problem and process the command again.
14. Before entering the FILESERV GENERATE command, you should ensure all storage group FBA minidisks are allocated in 8-block increments and are aligned on 4KB boundaries. Data spaces are not supported if there are any non-aligned FBA devices in any of the storage groups.
15. If you intend to use SFS facilities to back up the control data, process a FILESERV BACKUP command after FILESERV GENERATE completes successfully.
16. If you do not want to use SFS facilities to back up the control data, you must specify the NOBACKUP parameter in the *serverid* DMSPARMS file for the server machine.
17. If the CRR startup parameter is specified, the LUNAME *luname* DMSC6FGN GENERATE command is used. The LUNAME *luname* startup parameter establishes the fully qualified LU name of the CRR recovery server. For more information, see [LUNAME](#).



# FILESERV LIST

► FILESERV — LIST ◄

## Authorization

Repository File Pool Server Machine

## Purpose

Use the FILESERV LIST command to display the contents of the file pool catalogs. This command is intended to be used only for problem determination.

## Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV LIST will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.
3. All messages are written to the server machine console.
4. If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.
5. The FILESERV LIST command locates the file pool minidisks by using a POOLDEF file. The POOLDEF file it uses is the first file named *filepoolid* POOLDEF in its search order.

The *filepoolid* POOLDEF file may reside on a CMS minidisk with other file pool definition files for other server machines. It can also reside in another file pool.

6. This command must be run in the server machine that owns the file pool minidisks. If the command is run in another machine, conflicts with server processing may occur.
7. FILESERV LIST formats and writes catalog data to the file identified by the FILEDEF command for *ddname*=LIST. The *ddname*=LIST file contains fixed-length records with 68 positions and an ANSI printer control character in position 1. The FILEDEF command for *ddname*=LIST must be entered and in effect before the FILESERV LIST command is entered.
8. When the *ddname*=LIST file is assigned to a tape, multivolume tape files are supported if SL (standard labels) is specified.

**Note:** Only an output device can be specified on the FILEDEF LIST command.

9. If the *ddname*=LIST file is assigned to a minidisk, the minidisk must have sufficient free space to contain the current formatted contents of the file pool catalog space (storage group 1).
10. This command is not successful if any SFS logical units of work are found that have completed the first phase of the two-phase committing process. The logical units of work can be left in this condition by STOP IMMEDIATE or a server abend. This is also referred to as *prepared-and-not-connected* work. Normally, resynchronization resolves the prepared work when the server is executing in multi-user mode, making the prepared work consistent with the rest of the coordinated transaction. Before you enter FILESERV LIST, restart the server, use QUERY PREPARED to verify the prepared work has been committed, then enter the STOP command,

If you cannot wait for resynchronization to be completed, (for example, the communications line between the CRR recovery server and this file pool server is going to be down for a long time), you can manually resolve the prepared work by using the FORCE PREPARED command. For more

information, see [“FORCE” on page 539](#). See [“Forcing Prepared Work” on page 57](#) for more details on forcing prepared work.

## FILESERV LOG

► FILESERV — LOG — vdev1 — vdev2 ◄

### Authorization

File Pool Server Machine

### Purpose

Use the FILESERV LOG command to format and update the file pool log minidisks. It reformats the file pool logs when changing their size or location. Command processing updates the appropriate POOLDEF file and accesses all other file pool minidisks.

If you are changing the size of the file pool logs or are moving them to different minidisks, you need to adjust the z/VM system directory entry MDISK control statements for the file pool log minidisks before invoking this command.

**Note:** The FILESERV LOG command is similar to the “FILESERV CRRLOG” on page 502. The difference is the FILESERV CRRLOG command formats CRR log minidisks and the FILESERV LOG command formats the SFS log minidisks.

### Operands

#### vdev1

is the virtual device number of the first file pool log minidisk for the file pool. It must be specified.

#### vdev2

is the virtual device number of the second file pool log minidisk for the file pool. It must be specified.

**Note:** CMS FORMAT and RESERVE commands are entered for each file pool log minidisk when the startup parameter FORMAT is in effect. A blocksize of 4096 is used for the file pool log minidisks. If the startup parameter NOFORMAT is in effect, CMS FORMAT and RESERVE commands are not entered for the file pool log minidisks. (FILESERV LOG assumes the FORMAT and RESERVE commands have already been entered for the file pool log minidisks.)

### Usage Notes

1. This command must be issued from a file pool server machine. It is a dedicated maintenance mode command.
2. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV LOG will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.
3. All messages are written to the server machine console.
4. If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.
5. The FILESERV LOG command locates the file pool minidisks by using a POOLDEF file. The POOLDEF file it uses is the first file named *filepoolid* POOLDEF in its search order.

The *filepoolid* POOLDEF file may reside on a CMS minidisk with other file pool definition files for other file pool server machines. It can also reside in another file pool. The file mode for this file must be accessed read/write.

6. A temporary file named \$\$TEMP \$POOLDEF is created on the first read/write file mode in the search order. If FILESERV LOG is successful, the temporary file is erased.

7. If successful, the *filepoolid* POOLDEF file is updated.
8. Because FILESERV LOG can update the *filepoolid* POOLDEF file, you should run it in the file pool server machine that owns the file. If the command is run in another machine, conflicts with file pool server processing may occur.
9. Both *vdev1* and *vdev2* must be the same size and type. However, they must be on different physical devices.
10. The values specified for *vdev1* and *vdev2* will replace the virtual device numbers of the DDNAME=LOG1 and DDNAME=LOG2 entries in the *filepoolid* POOLDEF file.
11. For CRR recovery servers, IBM does not recommend doing backups of control data (that is, use the NOBACKUP startup parameter).
12. If new log minidisks are being defined and the current log minidisks exist and are still valid, you should backup the file pool control data before running FILESERV LOG. You can skip the back up if a backup file reflecting the current status of the file pool is available.
13. If the current log minidisks are not valid and are being reformatted by FILESERV LOG processing, there is no need to back up the file pool control data. In this situation, you should have a backup of each user storage group in case FILESERV LOG cannot be completed.
14. After FILESERV LOG completes successfully, previous control data backups are obsolete. Therefore, if the BACKUP startup parameter is in effect, you must back up the control data immediately after running FILESERV LOG.
15. If the command fails to complete successfully, it can be rerun. If the command fails and displays error messages, correct the problem and process the command again.
16. Because FILESERV LOG processing destroys all existing log information, it is incorrect to enter it unless the server was shut down with a STOP command or a STOP BACKUP command. If a FILESERV LOG is done without this “clean” shut down of the SFS server, it could result in inconsistent file pool data.

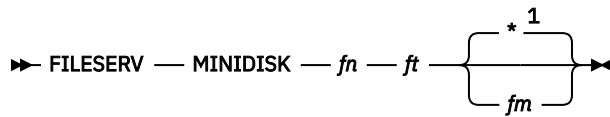
If the server was not shut down “cleanly”, the following message is displayed:

```
DMS3007R  Fileserv Log may cause data damage
DMS3007R  because outstanding log records.
DMS3007R  Enter '1' to continue or
DMS3007R  '0' to cancel Fileserv Log
```

You should respond 0 to cancel the operation. Otherwise, if it completes successfully, unpredictable results can occur.

17. The NOFORMAT option is not recommended for use in the DMSPARMS file when issuing this command on existing log minidisks. If you use the NOFORMAT option and enter CMS FORMAT and RESERVE commands on the existing log minidisks, and the FILESERV LOG command is rejected because it is not an appropriate time to enter a FILESERV LOG command, you may have to do a FILESERV GENERATE and restore all storage groups.
18. If you are replacing only one log minidisk, there is no need to use the FILESERV LOG command. Instead, follow the instructions on [“Replacing One File Pool Log Minidisk”](#) on page 131.

## FILESERV MINIDISK



Notes:

<sup>1</sup> The default control statement file is the first matching file in the CMS search order.

### Authorization

Repository File Pool Server Machine

### Purpose

Use the FILESERV MINIDISK command to add minidisks to storage groups in a file pool in dedicated maintenance mode. Command processing updates the appropriate POOLDEF file and accesses all file pool minidisks.

**Note:** If you want to add minidisks to storage groups in multiple user mode, use the administrator command “[FILEPOOL MINIDISK](#)” on page 470.

### Operands

#### *fn ft fm*

This parameter identifies the input control statement file that contains minidisk definition control statements. These control statements will be added to the *filepoolid* POOLDEF file. The format of a minidisk definition control statement is shown below. For a further description of a control statement file, see “[FILESERV GENERATE](#)” on page 513.

The file must have a fixed record format of 80 bytes (RECFM F LRECL 80).

#### Minidisk Definition Control Statement

This control statement identifies a new minidisk for a new or existing storage group. The format of a minidisk definition control statement is:

```
DDNAME=MDKnnnnn VDEV=vdev GROUP=grpnum BLOCKS=blocks
```

The following is a description of each control statement parameter:

#### **MDKnnnnn**

is the ddname the FILESERV command processing will use in the FILEDEF command issued for the minidisk.

The *nnnnn* can range from 3 through 32767 and must be assigned in ascending consecutive numeric sequence. For example, it is an error to use DDNAME=MDK00005 before DDNAME=MDK00004 is used. DDNAME=MDK00005 must be the next DDNAME used after DDNAME=MDK00004. The MAXDISKS value is the upper limit on what can be specified for *nnnnn*. MAXDISKS is a control statement in the control statement file required by the FILESERV GENERATE command. For more information, see [MAXDISKS](#).

If you aren't sure what the next available MDKnnnnn is, look in the POOLDEF file for the file pool.

#### **VDEV=vdev**

identifies the virtual device number for the minidisk being added to the file pool. CMS FORMAT and RESERVE commands are issued for the new minidisk when the startup parameter FORMAT is in effect.

A blocksize of 4096 is used for the minidisk. If the startup parameter NOFORMAT is in effect, CMS FORMAT and RESERVE commands are not issued for the new minidisk. (FILESERV MINIDISK assumes the FORMAT and RESERVE commands have already been entered for the new minidisk.)

### **GROUP=grpnum**

identifies the storage group (groupnumber) in the file pool to which the minidisk is being added. The storage group may or may not already exist. Storage group numbers do not need to be used in consecutive numeric sequence. The numbers cannot, however, exceed the value specified for MAXDISKS during file pool generation.

### **BLOCKS=blocks**

is an optional parameter handled automatically just as it is during FILESERV GENERATE. For more information on the BLOCKS parameter, see [“FILESERV GENERATE” on page 513](#).

## Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV MINIDISK will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,” on page 339](#).
3. All messages are written to the server machine console.
4. If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.
5. The FILESERV MINIDISK command updates the first file named *filepoolid* POOLDEF in its search order. The file mode for this file must be accessed read/write.  
  
The *filepoolid* POOLDEF file may reside on a CMS minidisk with other file pool definition files for other server machines. It can also reside in another file pool.
6. GROUP=1 is used exclusively for the file pool catalogs. During file pool generation, storage group 1 is allocated at least one minidisk.
7. GROUP=2 is used for user data. During file pool generation, storage group 2 is allocated at least one minidisk.
8. GROUP=3-32767 can be optionally used for user data. These groups may or may not have been allocated minidisks during file pool generation.
9. The values 3-32767 do not need to be assigned in ascending consecutive numeric sequence. For example, GROUP=5 can be assigned before GROUP=4.
10. If successful, this command will update the *filepoolid* POOLDEF CMS file with the new minidisk control statement(s).
11. The following temporary files are created on the first read/write file mode:
  - \$\$TEMP \$\$INPUT
  - \$\$TEMP \$POOLDEF
  - \$\$TEMP \$\$UPDATEIf FILESERV MINIDISK processing is successful, the files are erased.
12. Before entering the FILESERV MINIDISK command, you should ensure a current backup of the control data exists. If one doesn't exist, you should back up the control data.
13. Before entering the FILESERV MINIDISK command, you should ensure all FBA storage group minidisks are allocated in 8-block increments and are aligned on 4KB boundaries. FILESERV MINIDISK command processing will cause an error until the minidisks are realigned on 4KB boundaries.
14. After this command completes successfully, previous control data backup files are obsolete. Therefore, after this command completes successfully, you must back up the control data if the

BACKUP startup parameter is in effect. Use the FILESERV BACKUP command to back up the control data.

15. If FILESERV MINIDISK is unsuccessful, you should not immediately rerun it. Instead, you must take some recovery action. The action you take depends on whether message DMS3922I was displayed before the error occurred. Message DMS3922I is:

```
DMS3922I n minidisk(s) were added to the file pool
```

Take recovery actions as follows:

- If the FILESERV MINIDISK command caused an error before message DMS3922I is displayed, rerun the command:
    - If the command causes an error because of an attempt to add duplicate minidisks, manually update the DDNAME=MDKnnnnn control statements in the *fn ft fm* file that was specified on the FILESERV MINIDISK command, making sure no *vdev* address is a duplicate.
    - If the rerun of the command causes an error for another reason, take corrective action and rerun the command again.
    - If the rerun of the command is successful, no further action is required.
  - If the FILESERV MINIDISK command caused an error after the message DMS3922I was displayed, do not rerun the command. Manually add the new DDNAME=MDKnnnnn control statements to the *filepoolid* POOLDEF file.
16. When FILESERV MINIDISK updates the POOLDEF file, the server will add a BLOCKS parameter to the new DDNAME=MDKnnnnn control statement. For a description of DDNAME=MDKnnnnn control statement, see [“FILESERV GENERATE” on page 513](#).

## FILESERV MOVEUSER

```
► FILESERV — MOVEUSER — userid — group_number ◄
```

### Authorization

Repository File Pool Server Machine

### Purpose

Use the FILESERV MOVEUSER command to move all the file pool data for a user to a different storage group within the same file pool. Before using this command, see [“Moving an SFS User in Dedicated Maintenance Mode”](#) on page 96 for background information and details on moving users to another storage group.

The FILESERV MOVEUSER command may be used to move Byte File System (BFS) file spaces. The FILEPOOL UNLOAD and FILEPOOL RELOAD commands provide this function for those Byte File System file spaces.

**Note:** FILEPOOL UNLOAD and FILEPOOL RELOAD also work for SFS file spaces and are the preferred technique.

### Operands

#### *userid*

identifies the owner of the file space data to be moved. A nickname cannot be specified for *userid*.

#### *group\_number*

identifies the storage group where the user's file space data will be placed. The storage group must already exist (that is, must have at least one minidisk assigned to it). The *group\_number* can range from 2 to 32767.

### Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV MOVEUSER will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.
3. All messages are written to the server machine console.
4. If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.
5. The FILESERV MOVEUSER command locates the file pool minidisks by using a POOLDEF file. The POOLDEF file it uses is the first file named *filepoolid* POOLDEF in its search order.  
The *filepoolid* POOLDEF file may reside on a CMS minidisk with other file pool definition files for other server machines. It can also reside in another file pool.
6. You can change the ownership of an SFS file space using the FILEPOOL RENAME command.
7. User data cannot be moved to the catalog storage group (storage group 1).
8. If successful, all user file data is removed from the original storage group and moved to the new storage group.
9. Existing authorizations are not affected.



10. Existing explicit locks are not affected.
11. You should back up the control data before running FILESERV MOVEUSER if a current backup is not available.
12. A temporary file named \$\$TEMP \$\$INPUT is created on the first read/write file mode in the CMS search order. If FILESERV MOVEUSER processing is successful, the temporary file is erased.
13. After this command completes successfully, any previous control data backups are obsolete because they do not reflect the current location of the user's data. Therefore, after this command completes successfully, the server will allow you to process only a FILESERV BACKUP, another FILESERV MOVEUSER, or a FILESERV START with the RESTORE parameter in effect. Do not attempt to enter multiple FILESERV MOVEUSER commands until you have read the section [“Moving an SFS User in Dedicated Maintenance Mode”](#) on page 96.

After you are done moving users, you should immediately back up the control data. You should also back up all the *from* and *to* storage groups because any existing backups for them are obsolete.

14. If the command does not complete successfully, it can be rerun. If the rerun is unsuccessful because the user ID is already in the specified storage group (*group\_number*), no further reruns are required.
15. This command will be unsuccessful if any prepared work is found on the file pool log (because of a STOP IMMEDIATE or server abend). Prepared work involves those SFS logical units of work that have completed the first phase of the two-phase commit process. This is also referred to as *prepared and not connected* work. Normally, resynchronization resolves the prepared work when the server is executing in multi-user mode, making the prepared work consistent with the rest of the coordinated transaction. You should restart the server, verify the prepared work was completed using the QUERY PREPARED command, and then enter the STOP command prior to issuing FILESERV MOVEUSER.

If you cannot wait for resynchronization to complete (for example, the communications line between the CRR recovery server and this file pool server is going to be down for a long time), you can manually resolve the prepared work by using the FORCE PREPARED command. For more information, see [“FORCE”](#) on page 539. See [“Forcing Prepared Work”](#) on page 57 for more details on forcing prepared work.

16. If the user's file space is in a file pool being managed by DFSMS/VM, it may contain files that appear to be in the file pool, but which actually reside in the DFSMS/VM auxiliary storage repository. Such files are in *migrated* status. FILESERV MOVEUSER cannot move a file space containing files in migrated status; if they are present, it will cause an error with the following message:

```
DMS3723E Migrated files detected. FILESERV
MOVEUSER processing is terminated
```

To use FILESERV MOVEUSER successfully, the files in migrated status must be either erased or recalled. If you choose to recall these files, first ensure there is sufficient space in the current storage group to hold the files to be recalled. Then recall the files using DFSMS/VM commands and facilities. This can be done in the following manner to explicitly recall files:

- a. First, determine how many blocks these files are taking up by using the DFSMS REPORT command. (A sample report follows for VMTEST0.)
  - Subtract the number on the line "Physical 4K blocks in use by auxiliary storage ..." from the number on line "Physical 4K blocks ...". (In the following example report this would be 15 - 11 = 4.)
  - Next, subtract the above number from the "Logical 4K blocks currently in use ..." (Again, in this example it would be 25 - 4 = 21. Therefore 21 blocks are needed in order to recall all these files.)

```
DFSMS Function Level 221, Service Level 200    dd/mm/yy hh:mm:ss
DFSMS REPORT SPACEMANAGEMENT FILESPACE command, request identifier 117
```

```
DFSMS REPORT SPACEMANAGEMENT FILESPACE processing started
for file pool VMSYSU and userid VMTEST0
```

```
Files migrated for this file space:                6
Logical 4K blocks currently in use by this file space: 25
```

```
Physical 4K blocks in use by this file space:      15
Physical 4K blocks in use by auxiliary storage for this file space: 11
```

The following files are migrated:

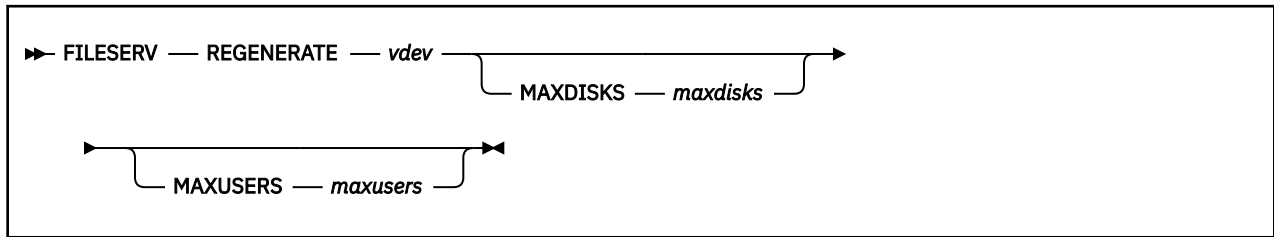
Filename	Filetype	Number of 4K Blocks	Management Class	Date Last Referenced	Date Migrated
Directory VMSYSU:VMTEST0.					
DGTTABL	MACLIB	10	--NULL--	dd/mm/yy	dd/mm/yy
LOAD	MAP	1	--NULL--	dd/mm/yy	dd/mm/yy
PROFILE	EXEC	1	--NULL--	dd/mm/yy	dd/mm/yy
SMPCNFGS	CONFIG	4	--NULL--	dd/mm/yy	dd/mm/yy
TABLES	MACLIB	4	--NONE--	dd/mm/yy	dd/mm/yy
TRY	THIS	1	--NULL--	dd/mm/yy	dd/mm/yy

DFSMS REPORT SPACEMANAGEMENT FILESPACE completed with no errors

- b. Next, ensure you have enough space (free blocks) in the users current storage group to hold the files to be recalled. This can be done by issuing the QUERY FILEPOOL STATUS command and looking at the "Currently Defined Minidisk Information" values for the minidisks in that user's storage group.
- c. If there is not enough space you will need to add space to the storage group. See [Chapter 9, "Managing Storage,"](#) on page 195 for more information.
- d. Use the ISMF file application, a DFSMS/VM facility, to list the files in migrated status you want to recall. (You must have explicit access to each directory in which the files reside. You must also have DFSMS/VM administration authority or read access to all migrated files and directories they reside in to do this step.) Or instead of using ISMF, you can code an EXEC using the output from the DFSMS REPORT command. The report lists the migrated files and the amount of space they require.
- e. Type RECALL on the command line to recall all the files on the list. (If you prefer, you can enter RECALL against individual files.)

For information about the DFSMS/VM commands, see [z/VM: DFSMS/VM Storage Administration](#).

# FILESERV REGENERATE



## Authorization

Repository File Pool Server Machine

## Purpose

Use the FILESERV REGENERATE command to expand the file pool control minidisk. It moves the file pool control minidisk to a larger minidisk without affecting the user storage group data. Command processing also updates the appropriate POOLDEF file and accesses all other file pool minidisks.

FILESERV REGENERATE can also be used to increase the maximum number of users who have file space (MAXUSERS) or the maximum number of minidisks (MAXDISKS) in the file pool. (MAXDISKS and MAXUSERS are established when the file pool is generated by the FILESERV GENERATE command.) Before using this command, see [Chapter 11, “Regenerating a Repository File Pool,”](#) on page 217.

## Operands

### *vdev*

identifies the virtual device number of the new, larger file pool control minidisk. This parameter is required.

### Note:

1. If neither MAXDISKS or MAXUSERS is specified, FILESERV REGENERATE only increases the size of the file pool control minidisk. When the control minidisk is increased, more physical space can be addressed as part of the file pool. That is, the potential number of 4KB blocks in the file pool is increased.
2. You must enter the following CMS FORMAT and CMS RESERVE commands for the new file pool control minidisk before running the FILESERV REGENERATE command:

```
format vdev fm (blksize 512
reserve filename filetype fm
```

Although anything can be specified for the file type in the RESERVE command, it is recommended you specify NCONTROL for *filetype*. The *filename* can also be of your own choosing, but it is recommended you use the file pool ID.

### **MAXDISKS** *maxdisks*

specifies the maximum number of minidisks (and storage groups) file pool can support. For *maxdisks*, specify the new maximum number of minidisks. The *maxdisks* must be greater than or equal to the current MAXDISKS value, but cannot be greater than 32767. (The current MAXDISKS value can be found in the POOLDEF file and in the QUERY FILEPOOL STATUS command output.)

MAXDISKS refers to the number of minidisks in all storage groups, including storage group 1. Do not count these minidisks as part of MAXDISKS:

- Control minidisk
- File pool log minidisks
- CRR log minidisks (if this is a CRR recovery server)

- Minidisk used for a security audit file
- Minidisk used for the POOLDEF file

If this operand is omitted, the current value of MAXDISKS is not changed.

### **MAXUSERS *maxusers***

specifies the estimated maximum number of file pool users who will be authorized to create objects (such as directories, files, SFS aliases, or Byte File System (BFS) links) in the file pool. For more information, see [“Step 1: Estimate the Maximum Number of Users \(MAXUSERS\)”](#) on page 249.

For *maxusers*, specify the increased number of users. The *maxusers* must be greater than or equal to the current MAXUSERS value for the file pool. (The current MAXUSERS value can be found in the POOLDEF file.) The maximum number of users that can be supported is 32767.

When the MAXUSERS parameter is specified and is greater than the current MAXUSERS value, FILESERV REGENERATE processing reorganizes the file pool catalogs. In this case, you must enter a FILEDEF for ddname TEMP before entering the FILESERV REGENERATE command. The file identified by ddname TEMP is not erased at the successful completion of FILESERV REGENERATE. Refer to [“FILESERV REORG”](#) on page 535 for a further description of the *ddname* TEMP file.

If MAXUSERS is specified, but is the same as the current MAXUSERS value, you do not need to enter a FILEDEF command for ddname TEMP. If MAXUSERS is omitted, no FILEDEF is required.

When both MAXDISKS and MAXUSERS are specified, they must be specified in the order shown in the command format. That is, MAXUSERS must follow MAXDISKS.

If MAXUSERS is omitted, the current value of MAXUSERS is not changed.

## Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV REGENERATE will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.
3. All messages are written to the server machine console.
4. For a list of supported virtual device numbers for tapes, see the description of the FILEDEF command in [z/VM: CMS Commands and Utilities Reference](#). The *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.
5. The FILESERV REGENERATE command locates the file pool minidisks by using a POOLDEF file. The POOLDEF file it uses is the first file named *filepoolid* POOLDEF in its search order.  
  
The *filepoolid* POOLDEF file may reside on a CMS minidisk with other file pool definition files for other server machines. It can also reside in another file pool.
6. This command updates the first *filepoolid* POOLDEF file found in the CMS search order. The file mode for this file must be accessed read/write.
7. The following temporary files are created on the first read/write file mode in the CMS search order:
  - \$\$TEMP \$\$INPUT
  - \$\$TEMP \$POOLDEF
8. If FILESERV REGENERATE is successful, the temporary files are erased.
9. Before running the command, the z/VM system directory entry for the server machine must be updated to define the virtual address for the new file pool control minidisk. The new file pool control minidisk must have a larger space allocation than the current file pool control minidisk. (Do not delete the MDISK entry for the old control minidisk until after the FILESERV REGENERATE command completes successfully.)

10. The MAXUSERS and MAXDISKS parameters have the same meaning as the FILESERV GENERATE control statements with the same name.
11. After this command completes successfully, the minidisk previously used as the file pool control minidisk is no longer required. You can remove the MDISK control statement from the server machine's z/VM system directory entry and use the minidisk for something else. A FILESERV REGENERATE message identifies the virtual address of the old file pool control minidisk.
12. Informational messages generated by server processing indicate processing is proceeding normally.
13. If the FILESERV REGENERATE command does not complete successfully, you must rerun it before performing any other processing using the file pool. Unless messages indicate the error was caused by an invalid parameter or a damaged control minidisk, rerun the command without changing any parameter values or reformatting the new control minidisk.
14. When the MAXUSERS parameter is specified and is greater than the current MAXUSERS value, FILESERV REGENERATE processing reorganizes the file pool catalogs. It implicitly performs all processing described for the FILESERV REORG command (see [“FILESERV REORG”](#) on page 535).  
If the command is unsuccessful after message DMS3701I is written to the server console, FILESERV REGENERATE processing must complete before the file pool can be used for other processing. In this case, ensure the file identified by ddname=TEMP is the same file created by the processing that was unsuccessful. The file identified by the ddname=TEMP must be read as input to rebuild the file pool's storage group 1 when the FILESERV REGENERATE command is rerun.
15. You should back up the control data before running FILESERV REGENERATE if a current backup is not available.
16. After this command completes successfully, all previous backups of the file pool control data are obsolete. Therefore, if the BACKUP startup parameter is in effect, you must back up the control data after this command completes successfully. Use the FILESERV BACKUP command to back up the control data.
17. This command will cause an error if any prepared work is found on the file pool log (because of a STOP IMMEDIATE or server abend). Prepared work involves those SFS logical units of work that have completed the first phase of the two-phase commit process. This is also referred to as *prepared and not connected* work. Normally, resynchronization resolves the prepared work when the server is executing in multi-user mode, making the prepared work consistent with the rest of the coordinated transaction. You should restart the server, verify the prepared work was completed using the QUERY PREPARED command then enter the STOP command prior to issuing FILESERV REGENERATE.  
If you cannot wait for resynchronization to complete (for example, the communications line between the CRR recovery server and this SFS file pool server is going to be down for a long time), you can manually resolve the prepared work by using the FORCE PREPARED command. For more information, see [“FORCE”](#) on page 539. See [“Forcing Prepared Work”](#) on page 57 for more details on forcing prepared work.
18. The FILESERV REGENERATE command is not applicable to CRR recovery servers because you should never have to increase the CRR recovery server's:
  - MAXDISKS beyond the recommended value of 2 for CRR recovery servers
  - MAXUSERS beyond the recommended value of 5 for CRR recovery servers
  - Control minidisk size, because CRR recovery servers never add records to the catalogs and never write any file blocks
 If you ever need to generate a CRR recovery server again, see [Chapter 15, “Generating a File Pool and Server,”](#) on page 247 for more information about restoring a file pool by generating it again.
19. Users with mounted BFS file systems from this file pool must re-IPL CMS or logoff and then re-mount the file system. Otherwise, inconsistencies may occur.
20. Informational message DMS3036I may be displayed during the Reload phase of the Regenerate. The message states:

```
hh:mm:ss Number of catalog records reloaded: nn
```

## **FILESERV REGENERATE**

This message lets you know the reload step is currently processing. It is displayed for every 1,500,000 catalog records that are reloaded.

# FILESERV REORG

► FILESERV — REORG ◄

## Authorization

Repository File Pool Server Machine

## Purpose

Use the FILESERV REORG command to delete unused file pool catalog entries for each SFS user and Byte File System (BFS) and to reorganize the file pool catalogs to ensure optimum use of catalog data and index space. (Space within the catalog storage group is used for indexes on the catalog data.) Before using this command, you should read the information on reorganizing file pool catalogs in [Chapter 10, “Reorganizing the File Pool Repository Catalogs,”](#) on page 213.

## Usage Notes

1. This command must be issued from a server machine. It is a dedicated maintenance mode command.
2. All messages are written to the server machine console.
3. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV REORG will use the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.
4. If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running.
5. The FILESERV REORG command locates the file pool minidisks by using a POOLDEF file. The POOLDEF file it uses is the first file named *filepoolid* POOLDEF in its search order.  
The *filepoolid* POOLDEF file may reside on a CMS minidisk or it can reside in another file pool.
6. You should back up the control data before running FILESERV REORG if a current backup is not available. If you use a non-SFS facility for backing up the file pool, you should make sure you have a recent backup before issuing FILESERV REORG.
7. Before entering FILESERV REORG, enter a FILEDEF command that associates a disk or tape file with *ddname* TEMP. FILESERV REORG processing writes catalog data to the file identified by *ddname* TEMP and then reads it.

For example, to define a disk output file, you might enter:

```
filedef temp disk filename filetype filemode
```

An example of a FILEDEF command that defines a tape output file is:

```
filedef temp tap1 sl
```

Standard labels (SL) allow multivolume tape file support.

**Note:** If a tape is used, do not specify the LABOFF operand on the FILEDEF command. Do not let the FILEDEF command default to LABOFF either.

If the TEMP file is assigned to a minidisk, the minidisk must have sufficient free space to contain the current contents of the file pool catalog space (storage group 1).

The file identified by *ddname* TEMP is not erased at the successful completion of FILESERV REORG.

8. For a list of supported virtual device numbers for tapes, see the description of the FILEDEF command in *z/VM: CMS Commands and Utilities Reference*.
9. If the FILESERV REORG command is unsuccessful, it can be rerun. Recovery actions may be necessary, however, depending on when the error occurred.

If the command was unsuccessful after the DMS3701I message is written to the server console, FILESERV REORG processing must complete before the file pool can be used for any other processing. In this case, you must rerun the FILESERV REORG command and insure the file identified by *ddname* TEMP is the same file created by the FILESERV REORG processing that was unsuccessful. The file identified by *ddname* TEMP must be read as input to rebuild storage group 1.

10. If FILESERV REORG processing is unsuccessful and cannot be rerun, you can recover by restoring the control data by issuing the FILESERV START command with RESTORE specified in the DMSPARMS file.
11. After this command completes successfully, all previous backups of the control data are obsolete. Therefore, if the BACKUP startup parameter is in effect, you must back up the control data after this command completes successfully. Use the FILESERV BACKUP command to back up the control data.
12. This command will cause an error if any prepared work is found on the file pool log (because of a STOP IMMEDIATE or server abend). Prepared work involves those SFS logical units of work that have completed the first phase of the two-phase commit process. This is also referred to as *prepared and not connected* work. Normally, resynchronization resolves the prepared work when the server is executing in multi-user mode, making the prepared work consistent with the rest of the coordinated transaction. You should restart the server, verify the prepared work was completed using the QUERY PREPARED command then issue the STOP command prior to issuing FILESERV REORG.

If you cannot wait for resynchronization to complete (for example, the communications line between the CRR recovery server and this SFS file pool server is going to be down for a long time), you can manually resolve the prepared work by using the FORCE PREPARED command. For more information, see [“FORCE” on page 539](#). See [“Forcing Prepared Work” on page 57](#) for more details on forcing prepared work.

13. Users with mounted BFS file systems from this file pool must re-IPL CMS or logoff and then re-mount the file system. Otherwise, inconsistencies may occur.



# FILESERV START

► FILESERV — START ◄

## Authorization

File Pool Server Machine

## Purpose

Use the FILESERV START command to start SFS server processing in *multiple user mode*. In *multiple user mode*, SFS server processing can support access to a repository file pool, act as a CRR recovery server, or act as a FIFO server.

FILESERV START runs successfully only if it is issued from a virtual machine that has been properly defined as a server machine. (For more information on defining a server machine, see [“Step 4: Define a Server Machine”](#) on page 256.)

## Usage Notes

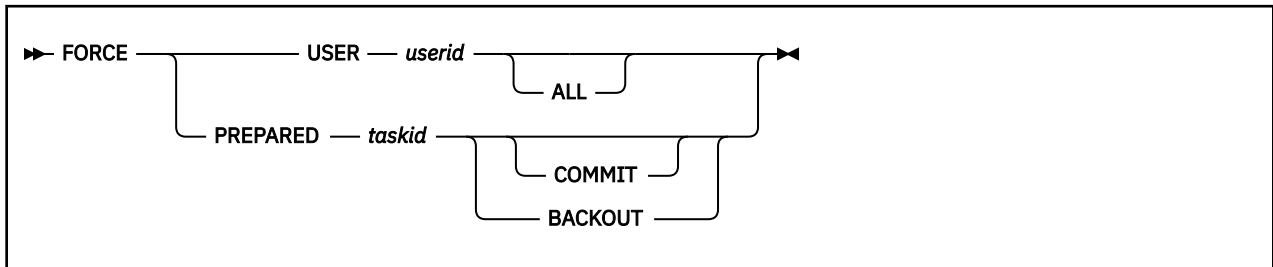
1. The FILESERV START command must be run in a server machine.
2. You can supply startup parameters in a file named *serverid* DMSPARMS, where *serverid* is the user ID of the server machine. FILESERV START uses the first *serverid* DMSPARMS file it finds in the CMS search order. For information on the startup parameters, see [Chapter 20, “File Pool Server Startup Parameters,”](#) on page 339.
3. All messages are written to the server machine console.
4. After server processing is initialized, the server can process file pool access requests from other virtual machines.
5. If you do not specify a FILEPOOLID startup parameter, the *filepoolid* defaults to the user ID of the server machine (*serverid*) in which the FILESERV command is running. The *filepoolid* is the name by which users refer to the file pool in their commands. It is also used as the transaction program name (TPN).
6. The FILESERV START command locates the file pool minidisks by using a POOLDEF file. The POOLDEF file it uses is the first file named *filepoolid* POOLDEF in its search order. FILESERV START displays a message that identifies the POOLDEF file it is using.  
  
The *filepoolid* POOLDEF file may reside on a CMS minidisk with other file pool definition files for other server machines. It can also reside in another file pool.
7. Only one server machine should access a particular file pool at any given time. No other processing that accesses the file pool should be in process when the FILESERV START command is issued.
8. During multiple user mode processing, server operator commands may be issued. The operator commands include a command to stop server processing. The operator commands are described in [“Operator Commands”](#) on page 359.
9. If the startup parameter RESTORE is in effect, the file pool control data will be restored before usual multiple user mode processing is started. For more information about restoring the control data, see [“Restoring Control Data”](#) on page 110. For more information about the RESTORE startup parameter, see [“Startup Parameter Descriptions”](#) on page 342.
10. During a warm start, if prepared work is found logged on the SFS logs, the SFS file pool server will assign an agent per instance of prepared work and reacquire locks associated with the prepared work. The transaction will then be recreated to the point of completion of the first phase of the two-phase commit process. If this is a CRR recovery server, resynchronization will start to process any in-doubt work that was still logged on the CRR logs. Resynchronization attempts to complete the

## FILESERV START

coordinated transactions. For further information on prepared work and CRR resynchronization see [“Resynchronization Function”](#) on page 305.

11. If the CRR startup parameter is in effect when this command is issued, the server will operate as a CRR recovery server. If a recovery server already exists, the command will cause an error.
12. Before entering the FILESERV START command, you should ensure all storage group FBA minidisks are allocated in 8-block increments and are aligned on 4KB boundaries. Data spaces are not supported if there are any non-aligned FBA devices in any of the storage groups.
13. If the FIFO startup parameter is in effect when this command is issued, the server being started will assume the server identified by the FIFO startup parameter will act as its named pipe or FIFO processing server. The identified FIFO server does not need to be active when the server starts.

## FORCE



### Authorization

Repository File Pool Operator

### Purpose

Use the FORCE USER operator command to roll back uncommitted work and sever the user's connections to the file pool server. Use the FORCE PREPARED operator command to commit or roll back SFS prepared work.

**Note:** The operator must be very careful when deciding whether to commit or back out the unavailable resource. For FORCE PREPARED, the SFS operator should consult the CRR operator to determine the proper heuristic decision. If the operator's heuristic decision is the opposite of the intent of synchronization point processing, the heuristic decision results in *heuristic damage*.

### Operands

#### USER

rolls back the identified uncommitted work.

#### *userid*

is the virtual machine identifier whose uncommitted work is to be rolled back and severed. The server rolls back the user ID's prepared work before severing the user ID from the file pool. Do not specify a nickname for *userid*. Nicknames are not recognized on server operator commands.

#### ALL

causes all connections for the user ID to be severed, even those from different virtual machines (see the "Usage Notes").

#### PREPARED

specifies to commit or roll back the identified prepared work. The QUERY PREPARED command *must* be entered before issuing FORCE PREPARED.

#### *taskid*

indicates the prepared work to be forced. It is the task ID previously displayed by the QUERY PREPARED command.

#### COMMIT

causes the user's prepared work to be committed.

#### BACKOUT

causes the user's prepared work to be rolled back.

### Usage Notes

1. The FORCE USER command only applies to SFS and Byte File System (BFS) APPC/VM connections. That is, connections associated with the input *userid* relating to CRR connections (for example, CRR Log Manager) are not affected by the FORCE USER command.
2. All messages are written to the file pool server operator console.

## FORCE

3. For FORCE PREPARED, two messages are issued. First, DMS3230I informs you the force has been scheduled. Then, DMS3377I is issued to let you know the heuristic decision has been recorded. (This recording is required in the event of resynchronization to indicate a heuristic decision was made). If the second message is not entered (the server goes down, for example), notify the operator of the corresponding CRR recovery server.
4. FORCE USER rolls back all uncommitted work and severs one or more APPC/VM paths between the virtual machine and the server. All the user's open files and directories in this file pool are closed. All the user's accessed SFS directories in this file pool are released.

If the user ID is connected to the server on APPC/VM paths from different virtual machines (perhaps the user has submitted a job to a batch machine that runs the job using the submitter's user ID):

- If the ALL option is specified on the command, all the user's uncommitted work is rolled back and all APPC/VM paths for the user ID are severed.
  - If the ALL option is not specified, the user's uncommitted work will not be rolled back or severed. The server will also display a message stating the user ID is connected from multiple virtual machines.
5. The FORCE USER command will not roll back prepared work. To roll back or commit prepared work, use the FORCE PREPARED command.
  6. If CRR security audit tracing is enabled, the FORCE PREPARED command generates an audit trace record.
  7. The ERASE LUNAME command erases the history of work forced by the FORCE PREPARED command.
  8. If you enter the FORCE PREPARED command at the file pool server (heuristic decision), you should ensure the CRR RESYNC command is issued at the appropriate CRR recovery server to record the heuristic decision made for the file pool. This prevents resynchronization processing from trying to complete the transaction with the file pool.

# GRANT ADMIN

---

```
▶ GRANT — ADMIN — userid ◀
```

## Authorization

File Pool Operator

## Purpose

Use the GRANT ADMIN operator command to give a user file pool administration authority. The authorization remains in effect until it is explicitly removed or the server processing ends.

File pool administration authority from the GRANT ADMIN command is temporary, and lasts only until one of the following occurs:

- Server processing is stopped or fails
- A DELETE ADMINISTRATOR command is entered for the user ID from an administration machine
- A REVOKE ADMIN operator command is entered at the server machine console.

For permanent file pool administration authority, use the ADMIN statement in the DMSPARMS file.

A user with file pool administration authority can enter commands and parameters on commands that general file pool users cannot. These commands are described in [“Administration Commands” on page 357](#). In addition, a user with file pool administration authority can do anything to users' objects that the users themselves can do. For example, a user with administration authority can read from or write to any file in the file pool, even though he or she has not been granted explicit authority to do so. For more information on administration authority, see [“File Pool Administration Machines” on page 25](#).

## Operands

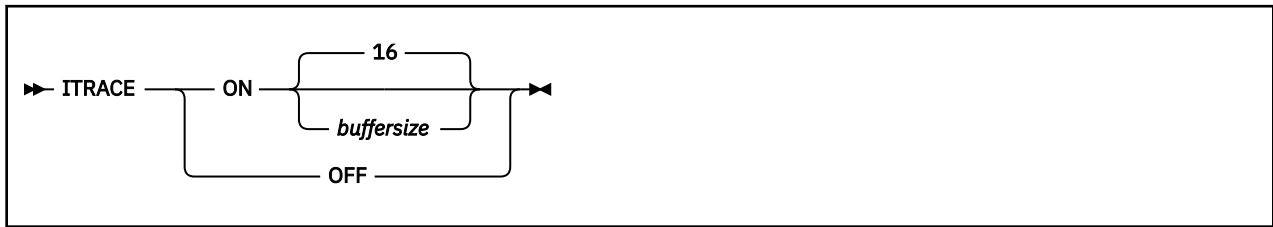
### *userid*

is the ID of the user to be given file pool administration authority. Do not specify a nickname for *userid*. Nicknames are not recognized on server operator commands.

## Usage Notes

1. All messages are written to the server operator console.
2. The administration authority takes effect when the user ID initiates the next logical unit of work in the file pool.
3. The ADMIN startup parameter identifies user IDs that are given file pool administration authority when the server is started. If the ADMIN startup parameter is not specified, use the GRANT ADMIN command to grant someone file pool administrator authority. For more information on the ADMIN startup parameter, see [Chapter 20, “File Pool Server Startup Parameters,” on page 339](#).

## ITRACE



### Authorization

File Pool Operator

### Purpose

Use the ITRACE operator command to start or stop server internal trace processing of APPC/VM communication related activities.

### Operands

#### ON *buffersize*

starts server internal trace processing and optionally identifies the size (*buffersize*) of the server machine trace buffer. Trace data is written to a buffer in storage.

**Note:** This data will be in a format compatible with the Dump Viewing Facility.

When the buffer is full, the server writes over the records at the beginning of the buffer, and continues writing records as needed until the buffer is again full. The server continues to write over the buffer as it generates more records. (An area of virtual storage used in this manner is typically called a *ring buffer*.)

If a buffer size value is specified, it must be greater than 0. An error will occur if a nonnumeric value or a numeric value less than 1 is specified. The buffer size value represents the size of the buffer in increments of 1024 (1KB) bytes which is then rounded to the next highest multiple of 4096 (4KB) bytes. For example, a buffer size value of 1, 2, 3, or 4 will result in a 4096-byte (4KB) buffer while a buffer size value of 5, 6, 7, or 8 will result in a 8192-byte (8KB) buffer.

The maximum buffer size allowed is 2097148KB.

If omitted, 16 is used for *buffersize*. This results in a default buffer size of 16384 bytes (16KB). If a buffer of the specified size cannot be acquired by server processing, a message will be issued and processing will continue without internal trace processing being performed.

#### OFF

stops server internal trace processing.

### Usage Notes

1. You should use the ITRACE facility only when requested by the designated support group for your installation.
2. All messages are written to the server operator console.
3. ITRACE processing traces only activities related to APPC/VM communications. For more information about APPC/VM communications, see *z/VM: CP Programming Services*.
4. ITRACE related errors will be identified by operator console error messages but will not stop server processing.

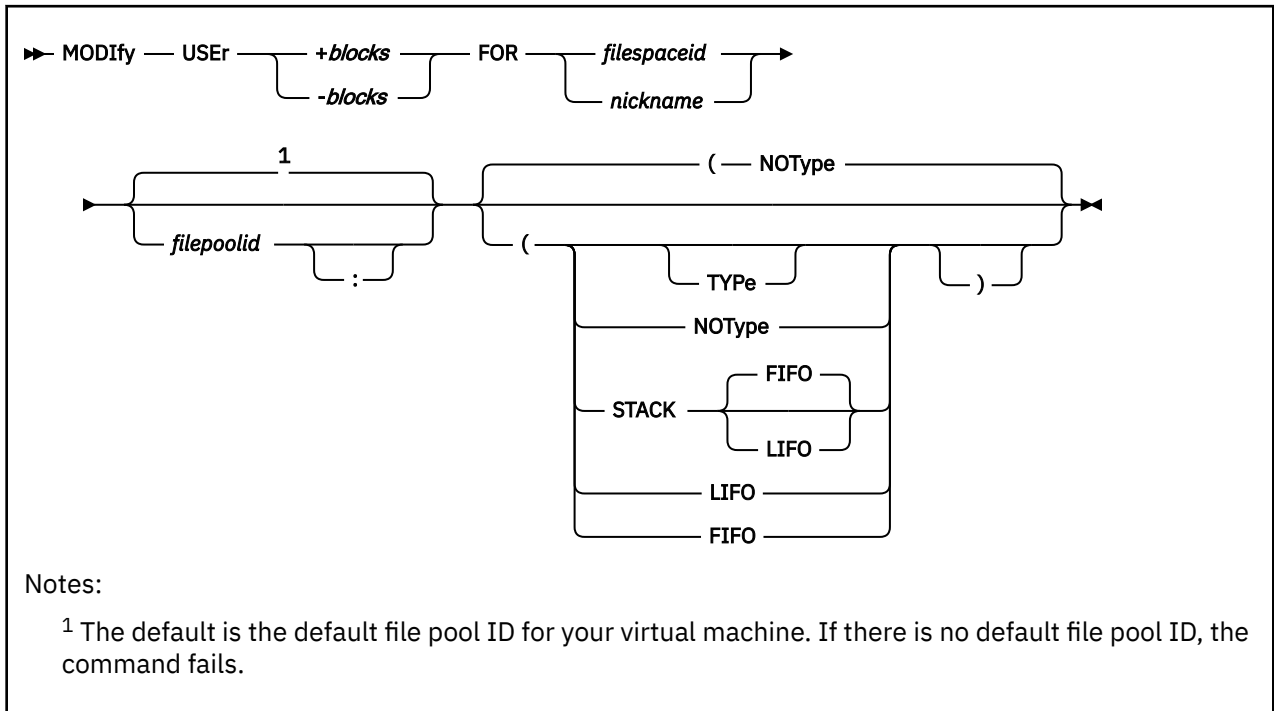
5. To view the internal trace data, a dump of the server machine must be made while internal tracing is active. To make the dump, enter this CP command at the server machine console while it is running in multiple user mode:

```
#cp vmdump 0-end format sfs
```

You can view the internal trace data by using the Dump Viewing Facility. For more information, see [z/VM: Dump Viewing Facility](#).

6. After ITRACE processing is active, you can increase or decrease the buffer size used by entering ITRACE OFF followed by ITRACE ON with the new buffer size.
7. You can also start internal trace processing by using the ITRACE startup parameter (see [ITRACE](#)).

## MODIFY USER



### Authorization

Repository File Pool Administrator

### Purpose

Use the MODIFY USER command to change the amount of file block space allocated to a file space in a specific file pool. (Space is initially allocated to a user by specifying the BLOCKS option on the ENROLL USER command.) You must have file pool administration authority to use this command.

### Operands

**+blocks**

**-blocks**

indicates the number of blocks to be added to or subtracted from the file space allocation. Remember for SFS file spaces, the space allocation is really the maximum number of committed blocks that may be used in the file space. For BFS file spaces, the space allocation is the maximum number of blocks that may be used.

The space allocation is a limit. The value of *blocks* can range from 1 through 2,147,483,647.

**FOR filepaceid**

identifies the file space for which the storage allocation is to be changed.

**FOR nickname**

identifies a nickname for user ID, a Byte File System (BFS), or a list of user IDs whose storage allocation is to be changed. (Use the NAMES command to define nicknames.) You cannot use a nickname representing a list if any of the file spaces in the list are BFS.

**filepoolid**

**filepoolid:**

is the ID of the file pool in which the user's allocation is to be modified. If not specified, the default file pool ID for your virtual machine will be used.



## Options

### NOType

suppresses the display of file space IDs whose space allocation has been modified. NOType is the default.

### TYPE

displays at the terminal the file space IDs whose space allocation has been modified.

### STACK FIFO

### STACK LIFO

places the file space IDs whose space allocation has been modified in the console stack rather than displaying it at the terminal. FIFO is the default.

### LIFO

specifies the file space IDs whose space allocation has been modified should be stacked in a last in, first out order. This option is equivalent to STACK LIFO.

### FIFO

specifies the file space IDs whose space allocation has been modified should be stacked in a first in, first out order. This option is equivalent to STACK FIFO.

## Usage Notes

1. The deletion of space allocations acts upon unused space. A file pool server calculates unused space by subtracting the space currently being used from the total space allocation to the user. If the amount being deleted is greater than the unused portion, the command will cause an error.
2. You can change the ownership of a file space using the FILEPOOL RENAME command.
3. You can use the QUERY LIMITS command to determine the user's current 4K block limit and the number of 4K blocks that are committed. For a description, see [“QUERY LIMITS” on page 655](#).
4. If a *nickname* representing a list of user IDs was specified, and the modification is unsuccessful for one user in the list, it is unsuccessful for all users in the list.
5. If the MODIFY USER command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
6. If you specify a nickname and the NODE tag in the NAMES file indicates the user is on another processor, the LOCALID tag must also be specified.

## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in *z/VM: CMS Commands and Utilities Reference*.

This command internally issues IDENTIFY, FINIS, NUCXDROP, and NUCXLOAD commands. Therefore, messages from those commands can also be displayed. (For more information about those commands, see *z/VM: CMS Commands and Utilities Reference*.)

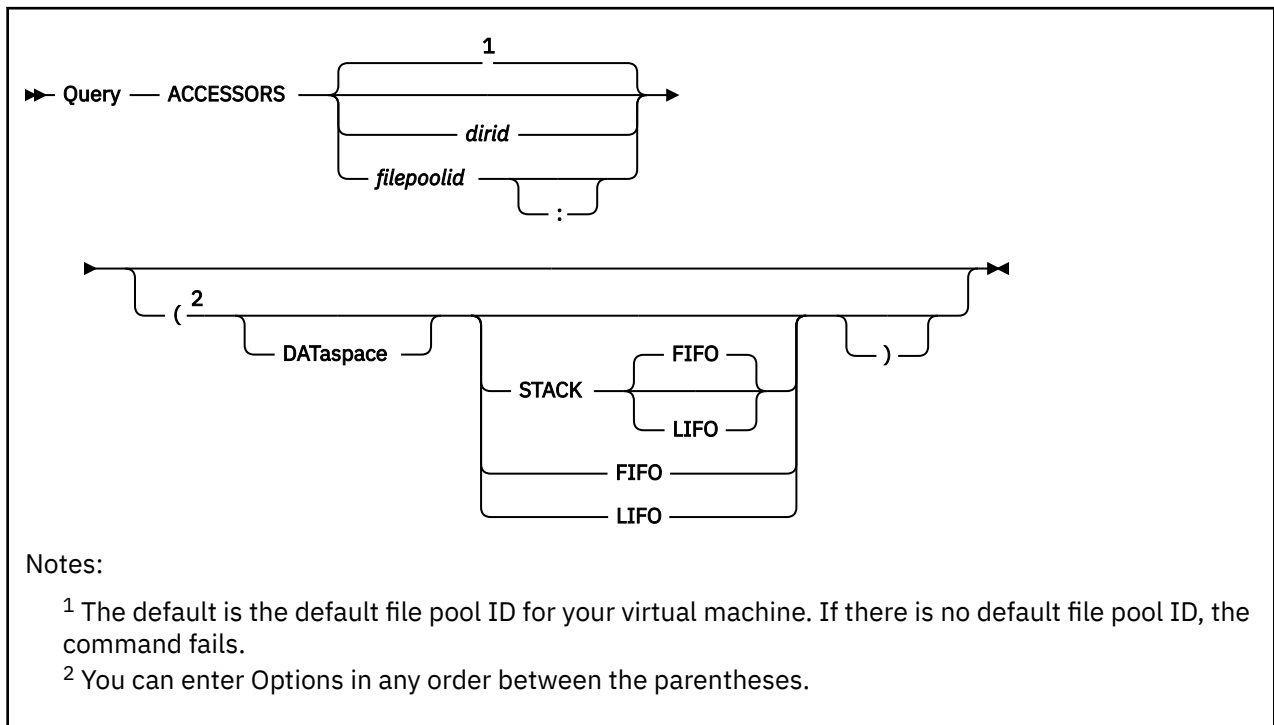
Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS066E *option1* and *option2* are conflicting options [RC=24]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS149E Userid *filepaceid* not valid [RC=32]
- DMS389E Incorrect operand: *operand* [RC=24]
- DMS637E Missing nodeid for the AT operand [RC=24]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=32]
- DMS653E Error executing *command rc=nn* [RC=40]

## MODIFY USER

- DMS1167E Userid *filespaceid* is not enrolled [RC=40]
- DMS1171E You are attempting to delete too much storage for *filespaceid*[RC=40]
- DMS1201E STACK option cannot follow FIFO or LIFO [RC=24]
- DMS1209E Nickname *nickname* resolved to more than one name; only one BFS file system can be changed at a time [RC=40]
- DMS1294E The requested block increment exceeds the maximum allowed for userid *filespaceid* [RC=40]

## QUERY ACCESSORS (SFS only)



### Authorization

Repository File Pool Administrator

### Purpose

The QUERY ACCESSORS command displays information about current accessors of directory control directories. (Only SFS directories that have the directory control attribute are directory control directories).

There are two versions of the QUERY ACCESSORS command: one is for general users, while the other is for users with file pool administration authority. For users with file pool administration authority, this lets you determine which directories currently reside in data spaces. See [z/VM: CMS Commands and Utilities Reference](#) for the QUERY ACCESSORS command for general users.

### Operands

#### *dirid*

identifies a directory control directory for which you want to display information. If you omit *dirid*, information is displayed for all directory control directories in the specified file pool or default file pool.

#### *filepoolid*

#### *filepoolid:*

establishes the file pool for which the query is intended. If both *filepoolid* and *dirid* are omitted, the default file pool for your virtual machine displays all directory control directories in that file pool.

### Options

#### **DATAspace**

means the results are to include an indication of whether the directories reside in data spaces. File pool administration authority is required to use this option.

**STACK FIFO**

**STACK LIFO**

causes the results of the QUERY command to be placed in the program stack instead of being displayed at the terminal. The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO. Note the heading is not put on the stack.

**FIFO**

(first-in first-out) is the default option for STACK. FIFO causes the results of the QUERY command to be placed in the program stack instead of being displayed at the terminal. The information is stacked FIFO. The options STACK, STACK FIFO, and FIFO are all equivalent. Note the heading is not put on the stack.

**LIFO**

(last-in first-out) causes the results of the QUERY command to be placed in the program stack rather than being displayed at the terminal. The information is stacked LIFO. This option is equivalent to STACK LIFO. Note the heading is not put on the stack.

**Usage Notes**

1. For this command, you are an accessor if you:
  - Entered a CMS ACCESS or VMLINK command for a directory control directory and have not yet released it.
  - Used the CSL Open Directory (DMSOPDIR) routine with the FILE keyword for a directory control directory and have not yet issued the Close Directory (DMSCLDIR) for it.
2. If you have administration authority, you can enter QUERY ACCESSORS for any directory control directory in the file pool. General users can enter QUERY ACCESSORS for their own directory control directories or for directory control directories for which they have directory write authority.
3. If you have administration authority, it is not possible to display information about only your own directory control directories with a single command. Instead, information for all directory control directories in the file pool is displayed when you omit *dirid*. To display information about one of your own directories, specify the *dirid* parameter.
4. If the QUERY ACCESSORS command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
5. The QUERY ACCESSORS command will cause an error if you specify a file control directory.
6. Error messages and the heading are suppressed if QUERY ACCESSORS is invoked:
  - As a function from a program
  - From a CMS exec file that has the &CONTROL NOMSG option in effect
  - From an EXEC 2 exec where CMDCALL is not in effect
  - From a REXX exec with ADDRESS COMMAND in effect

**Responses**

QUERY ACCESSORS displays one line for each accessor of each directory specified. If information for more than one directory is returned, the information is displayed in alphanumeric order by directory name.

The following example includes a column labeled 'DS'. It also contains a column labeled 'Level'. These columns are included only when the DATASPACE option is specified.

Accessors	Mode	DS	Level	Directory Name
BENNETRB	R/O	-	2	VM8SERV:BARNES.TEST
BARNES	R/W	-	0	VM8SERV:BENNETRB.MYTEST
WORMKJ	R/O	X	1	VM8SERV:BENNETRB.MYTEST
TERRYCR	R/O	X	2	VM8SERV:BENNETRB.MYTEST
CROCKETT	R/O	X	1	VM8SERV:BENNETRB.NEWTEST

**Note:** The header is generated only if output is displayed at the terminal.

If you omit the DATASPACE option, the display has this form:

Accessors	Mode	Directory Name
BENNETRB	R/O	VM8SERV:BARNES.TEST
BARNES	R/W	VM8SERV:BENNETRB.MYTEST
WORMKJ	R/O	VM8SERV:BENNETRB.MYTEST
TERRYCR	R/O	VM8SERV:BENNETRB.MYTEST
CROCKETT	R/O	VM8SERV:BENNETRB.NEWTEST

where:

#### Accessors

is the ID of the user who has the directory accessed.

#### Mode

is the status of the access of the SFS directory: R/O (read-only) or R/W (read/write).

#### DS

is the data space (DS) indicator. An 'X' in this column means the directory currently resides in a data space.

#### Level

indicates the level of the directory accessed by a reader. Each time a reader accesses a directory control directory and there have been changes (including additions and deletions), made to files in that directory (since the previous reader accessed it), the accessor gets a new access level assigned. Note that a directory/level pair consumes a unique data space. When mode is R/W, the Level value is always 0. For more information about how directory control directories work in data spaces, see [z/VM: CMS User's Guide](#).

#### Directory Name

is the name of the directory being accessed.

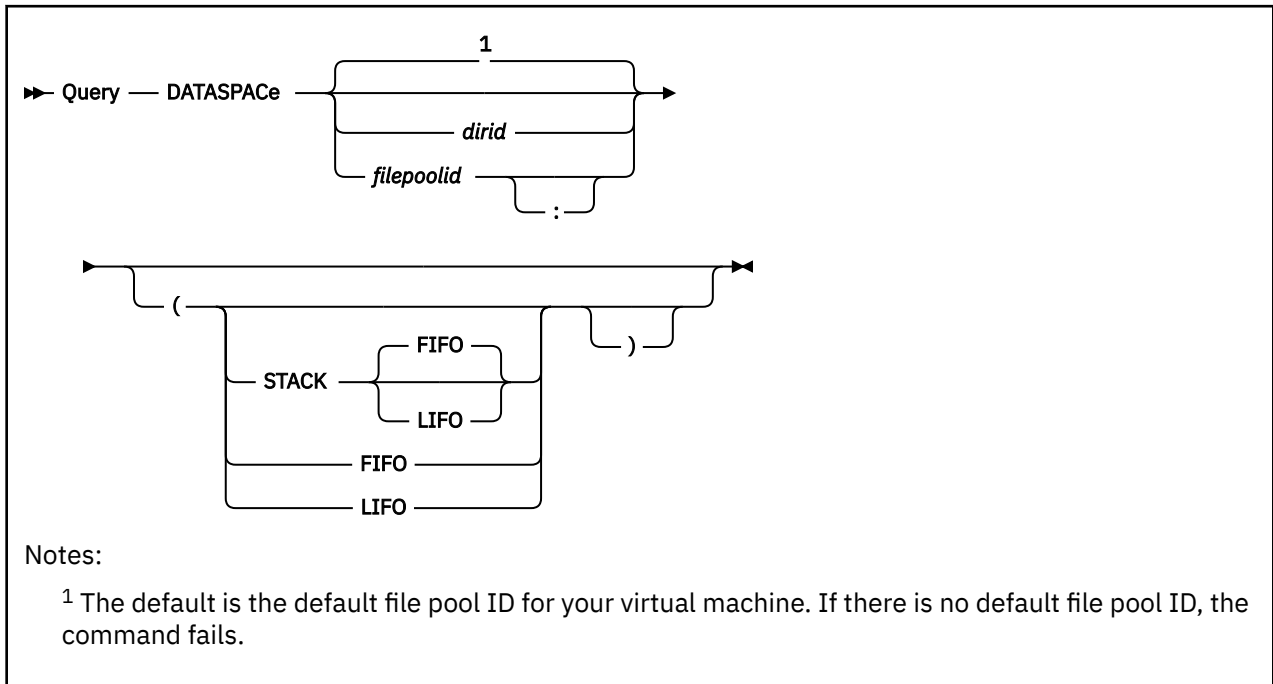
## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS066E *option1* and *option2* are conflicting options [RC=24]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS1184E Directory *dirname* not found or you are not authorized for it [RC=28]
- DMS1223E There is no default file pool currently defined [RC=40]
- DMS2023E File pool *filepoolid* does not support the requested QUERY command [RC=88]
- DMS2037E Directory *dirname* is not a directory control directory [RC=40]
- DMS2041E You are not authorized to use the DATASPACE option [RC=76]
- DMS2042I No users are accessing directory control directories [RC=0]
- DMS2042I No users are accessing directory *dirname* [RC=0]
- DMS2044E No directory control directories exist or you are not authorized for any [RC=28]

## QUERY DATASPACE (SFS only)



### Authorization

Repository File Pool Administrator

### Purpose

Use the QUERY DATASPACE command to display information about the eligibility of directory control directories for use of data spaces. Eligibility is established or removed by using the DATASPACE command. File pool administration authority is required to process this command.

### Operands

#### *dirid*

identifies an SFS directory for which you wish to display eligibility. Only directory control directories may be specified.

If you specify *dirid*, the display indicates whether that directory has been assigned for data space eligibility. If you omit *dirid*, the display lists all directories in the file pool that have been assigned for data space eligibility.

#### *filepoolid*

#### *filepoolid:*

is the name of the file pool for which the query is intended. If *filepoolid* and *dirid* are both omitted, the default file pool lists all directories in that file pool that have been assigned for dataspace eligibility.

### Options

#### STACK FIFO

#### STACK LIFO

causes the results of the QUERY command to be placed in the program stack instead of being displayed at the terminal. The information is stacked either FIFO (first in first out) or LIFO (last in first out). The default order is FIFO. Note that the heading is not put on the stack.

**FIFO**

(first-in first-out) is the default option for STACK. FIFO causes the results of the QUERY command to be placed in the program stack instead of being displayed at the terminal. The information is stacked FIFO. The options STACK, STACK FIFO, and FIFO are all equivalent. Note that the heading is not put on the stack.

**LIFO**

(last-in first-out) causes the results of the QUERY command to be placed in the program stack rather than being displayed at the terminal. The information is stacked LIFO. This option is equivalent to STACK LIFO. Note that the heading is not put on the stack.

**Usage Notes**

1. The resultant display shows which directories are assigned for data space eligibility. Even though a directory is eligible for data space use, there is no guarantee the directory will, in fact, reside in a data space. For more information about SFS use of data spaces, see [Chapter 14, “Using Data Spaces \(SFS Repository Servers Only\),”](#) on page 241.
2. To determine which directories currently reside in data spaces, use the QUERY ACCESSORS command with the DATASPACE option. See [“QUERY ACCESSORS \(SFS only\)”](#) on page 547.
3. If the QUERY DATASPACE command is called from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
4. Error messages and the heading are suppressed if QUERY DATASPACE is invoked:
  - As a function from a program
  - From a CMS exec file that has the &CONTROL NOMSG option in effect
  - From an EXEC 2 exec where CMDCALL is not in effect
  - From a REXX exec with ADDRESS COMMAND in effect

**Responses**

Here are some examples:

For:

```
QUERY DATASPACE  dirid

Assigned Directory Name
YES      VM8SERV:BRAD.TEST
```

For

```
QUERY DATASPACE

Number of Directories Eligible to use Dataspaces: nnnn
Directory Name
FP3:CHERIE.TEST
FP3:BRAD.TEST
FP3:BUTCH.TEST
:
```

**Messages and Return Codes**

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

Messages:

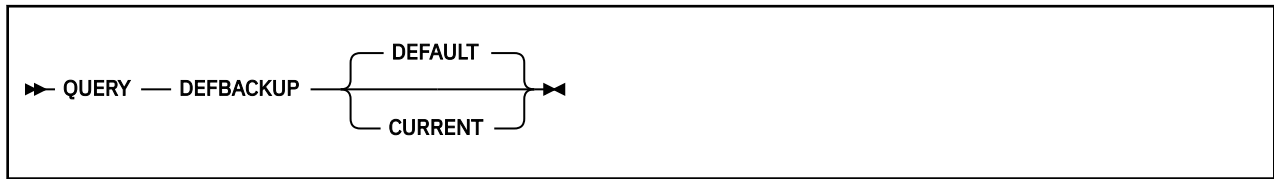
- DMS065E *option* option specified twice [RC=24]

## QUERY DATASPACE

- DMS066E *option1* and *option2* are conflicting options [RC=24]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS1139E You are not authorized to enter this command [RC=76]
- DMS1184E Directory *dirname* not found or you are not authorized for it [RC=28]
- DMS1223E There is no default file pool currently defined [RC=40]
- DMS2023E File pool *filepoolid* does not support the requested QUERY command [RC=88]
- DMS2037E Directory *dirname* is not a directory control directory [RC=40]
- DMS2043I No directories are eligible for a dataspace [RC=0 ]
- DMS2044E No directory control directories exist or you are not authorized for any [RC=28]



## QUERY DEFBACKUP



### Authorization

Repository File Pool Operator

### Purpose

Use the QUERY DEFBACKUP operator command to determine the default and current assignments of the file pool control data backup file, or to determine where the last control data backup file was created.

### Operands

#### DEFAULT

causes the default control data backup file destination currently in effect to be displayed. If no operands are specified on the BACKUP or STOP commands, control data backup files will be directed to the identified destination.

If no operands are specified on the QUERY DEFBACKUP command, DEFAULT is assumed.

#### CURRENT

shows where the current valid (for example, last successful) control data backup file was created.

If backup processing is taking place at the time the command is issued, the destination of the control data backup file currently being created will be displayed.

### Usage Notes

1. All messages are written to the server machine operator console.
2. For CRR recovery servers, IBM does not recommend doing backups of control data (that is, use the NOBACKUP startup parameter).
3. This command may be used by an operator if the definition of the control data backup file in the POOLDEF file is forgotten, if the results of a previous DEFBACKUP command are lost or unknown, or to show where the current control data backup file is being created.
4. The command can also be used to display the location where the last valid control data backup file was created.

### Responses

In response to this command the server displays message(s) describing the control data backup file definition. For example, if you enter:

```
query defbackup
```

the server may display:

```
DMS3613I Default control data backup file will
be directed to tape device 181
```

or, if a valid control data backup file destination does not exist, the server may display:

## QUERY DEFBACKUP

```
DMS3600W Control data backup file not defined
```

If you enter:

```
query defbackup current
```

and backup processing is currently taking place, the server may display:

```
DMS3615I Current control data backup file is
          directed to file pool ADMIN
DMS3615I File name: BACKUP File type: CNTLDATA
DMS3615I Directory id: ADMIN:VMSYSU.BKUP
```

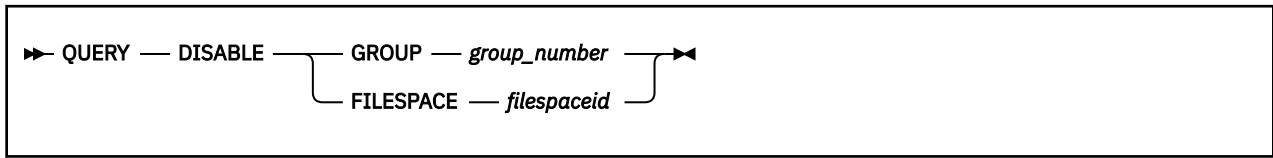
If you enter:

```
query defbackup current
```

and backup processing is not currently taking place, the server may display something similar to the following:

```
DMS3614I Last successful control data backup
          file was directed to minidisk.
DMS3614I File name: BACKUP File type:
          CNTLDATA File mode: B
```

## QUERY DISABLE



### Authorization

Repository File Pool Operator

### Purpose

Use the QUERY DISABLE operator command to determine if a storage group or file space has been previously disabled, as well as the user ID of the disabler.

### Operands

#### **GROUP** *group\_number*

identifies the number of a storage group for which you want information. For *group\_number*, specify a number from 2 to the MAXDISKS value for the file pool.

#### **FILESPACE** *filespaceid*

identifies the file space for which you want disable information. Do not specify a nickname for *filespaceid*. Nicknames are not recognized on server operator commands.

### Usage Notes

- This command might be used to determine disable information for a specified file space or storage group when the results are lost or unknown of one of the following:
  - DISABLE operator command
  - FILEPOOL DISABLE command
  - Disable File Space (DMSDISFS) CSL routine
  - Disable Storage Group (DMSDISSG) CSL routine
  - FILEPOOL RELOAD
  - FILEPOOL UNLOAD
  - FILEPOOL BACKUP and FILEPOOL RESTORE commands
  - FILEPOOL RENAME command
- To get the equivalent information from an administrator's or general user's user ID, use the QUERY FILEPOOL DISABLE command. (See [“QUERY FILEPOOL DISABLE”](#) on page 577 for more information.)

### Responses

In response to this command, the server displays message(s) describing the DISABLE status of the object being queried. Note that if the DISABLE operator command was used to disable the object and an *owner* was specified on the command, the user ID shown in the QUERY DISABLE response is the *owner*, not the user ID that issued the DISABLE.

Message content consists of:

```

msgid objecttype objectid DISABLED BY
userid mode link-status
  
```

For a QUERY DISABLE GROUP 4, the output might look like:

## QUERY DISABLE

```
DMS3707I Storage Group 4 disabled by JONES, MODE=EXCLUSIVE  
Devices are detached
```

Or, if the storage group is not disabled, you will see:

```
DMS3709I Storage Group 4 not disabled
```

For a QUERY DISABLE FILESPACE JONES, the output might look like:

```
DMS3708I Filespace JONES disabled by SMITH, MODE=SHARE
```

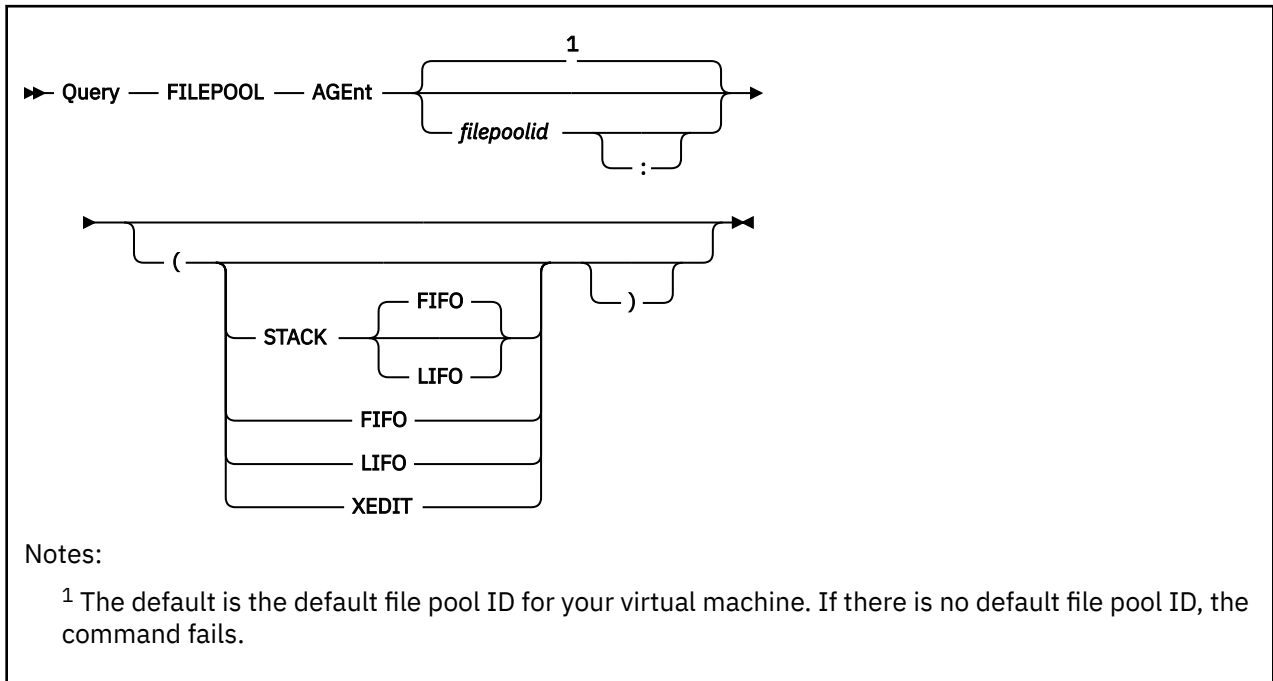
Or, if the file space is not disabled, you might see:

```
DMS3710I Filespace JONES not disabled
```

If the file space is disabled by a function such as the FILEPOOL RENAME command, the output might look like:

```
DMS3708I File space and storage group disabled by RENAME.
```

## QUERY FILEPOOL AGENT



### Authorization

File Pool Administrator or Connected User

### Purpose

Use the QUERY FILEPOOL AGENT command to display information about the users or internal processes for which the file pool server or CRR recovery server is currently doing work. This includes the total number of agents, the highest value, current number of agents, and information about each agent.

The information returned can assist you with the task of performance analysis and timing for file pool servers.

Note that this command should not be used by permanent applications to acquire data. The content of the output from this command may change in the future.

### Operands

*filepoolid*

**filepoolid:**

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

### Options

**STACK FIFO**

**STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.

**FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

**LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

**XEDIT**

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

**Usage Notes**

1. The XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set on. CMS FULLSCREEN also lets you scroll through the information.
2. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL AGENT command with the XEDIT option on the command line. When a new file is edited, there is not a chance of accidentally overlaying information already in the file.
3. If the QUERY FILEPOOL AGENT command is issued from an exec or an assembler program for a file pool that is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command fails.
4. If the current release of z/VM is processing prior release file pool information, you will only see the information applicable for that prior release.

**Responses**

The QUERY FILEPOOL AGENT command displays information about the users or internal processes for which the server is currently doing work.

An *agent* is the server's internal representation of a user (or system task). There are usually far fewer agents than logged-on users. The server calculates how many agents it needs based on what is specified in the USERS startup parameter.

An example of the output is:

```

VM9SERV File Pool Agents

Start-up Date mm/dd/yy          Query Date mm/dd/yy
Start-up Time hh:mm:ss         Query Time hh:mm:ss
=====
AGENT INFORMATION
    16 Total Number of Agents
     2 Active Agents Highest Value
     2 Current Number of Agents

Userid  Type    Status  Agent Number  Wait  Uncommitted Blks
CHECKPT Chkpt   Wait    2             I/O    0
ANASTOSM User    Read    4             None   0
=====

```

For the *Type*, *Status*, and *Wait* fields, one indication from the following list under the respective heading is displayed for each agent.

```

Userid  Type    Status  Agent Number  Wait  Uncommitted Blks
uuuuuuuu Chkpt   Read    xxxxx        None  xxxxxxxx
User    Write   Communication
Syncpt-Logr Wait    Checkpoint
Resync-Task Inact   Lock
Pc->Resync Running Catalog_Buffer
Rm->Resync Prepared Control_Buffer
Resync->Rm I/O
BFS_User Connection
Timer
Subtask
Multi-Event
Suspended

```

```
AVSwait
ESM_Wait
DFSMS/VM_Wait
xx
Storage_Wait
Token_Callback
```

For a description of these fields, see [AGENT INFORMATION](#) of the command [“QUERY FILEPOOL REPORT”](#) on page 592.

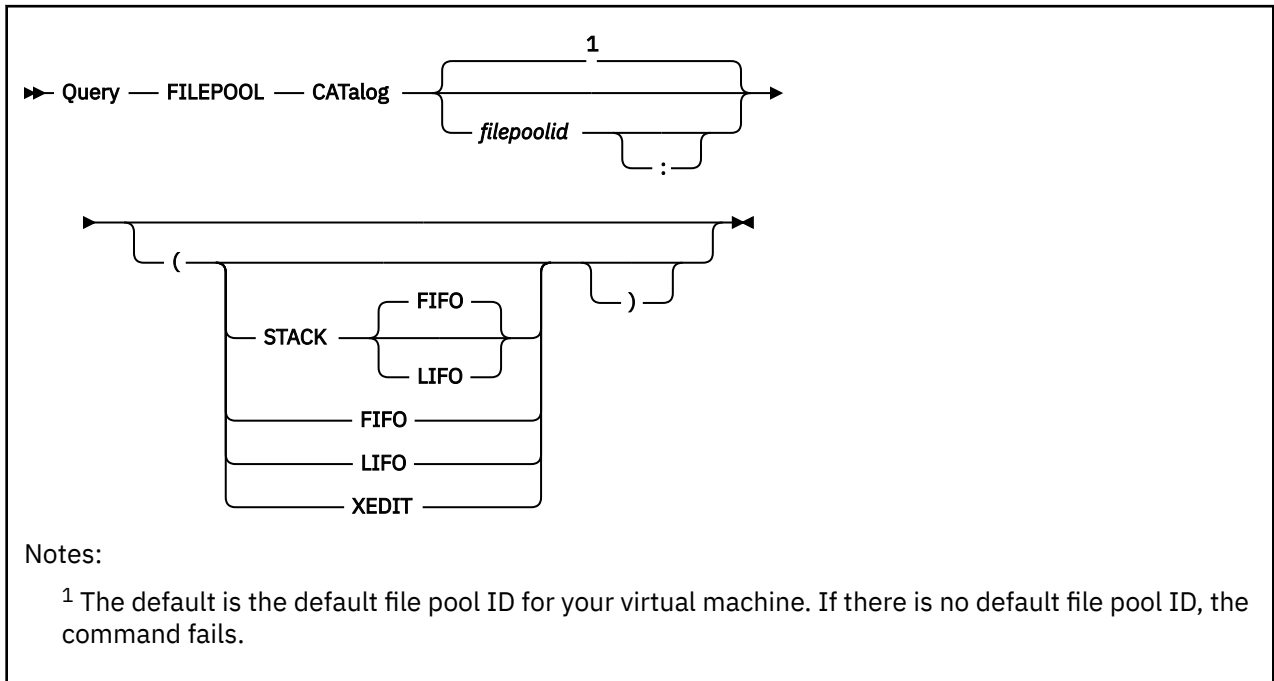
## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24 ]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS688E XEDIT option only valid from XEDIT environment [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24 ]
- DMS1223E There is no default file pool currently defined [RC=40 ]

## QUERY FILEPOOL CATALOG



### Authorization

File Pool Administrator

### Purpose

Use the QUERY FILEPOOL CATALOG command to display information about the file pool catalog space.

The information returned can assist you with the task of performance analysis and timing for file pool servers. The information returned is generally not useful for CRR recovery servers or dedicated FIFO servers.

File pool administration authority is required to process this command.

This command should not be used by permanent applications to acquire data. The content of the output from this command may change in the future.

### Operands

*filepoolid*

*filepoolid:*

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

### Options

**STACK FIFO**

**STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.

**FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

**LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.



**XEDIT**

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

**Usage Notes**

1. The XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set to ON. CMS FULLSCREEN also lets you scroll through the information.
2. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL CATALOG command with the XEDIT option on the command line. When a new file is edited, there is not a chance of accidentally overlaying information already in the file.
3. If the QUERY FILEPOOL CATALOG command is issued from an exec or an assembler program for a file pool that is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
4. If the current release of z/VM is processing prior release file pool information, you will only see the information applicable for that prior release.
5. Avoid excessive use of this command as considerable processing may be required to obtain the requested information.

**Responses**

The QUERY FILEPOOL CATALOG command displays information about the file pool catalog space.

An example of the output is:

```

VM9SERV File Pool Catalogs

Start-up Date dd/mm/yy          Query Date dd/mm/yy
Start-up Time hh:mm:ss         Query Time hh:mm:ss
=====
CATALOG STORAGE GROUP INFORMATION

Storage   Minidisk  4K Blocks   4K Blocks   Virtual
Group No. Number   In-Use     Free        Address
  1         1      124 - 42%   168         0303
  1         2      101 - 35%   191         0304
  1         3       71 - 24%   221         0305
Storage Group 1 Totals 296 - 34%   580
=====
CATALOG SPACE INFORMATION

503 Data Blocks
174 Occupied Data Blocks
35% Percent Occupied Data Blocks
505 Index Blocks
118 Occupied Index Blocks
23% Percent Used Index Blocks
=====

```

See [STORAGE GROUP MINIDISK INFORMATION](#) and [CATALOG SPACE INFORMATION](#) of the QUERY FILEPOOL REPORT command for a description of these fields.

**Messages and Return Codes**

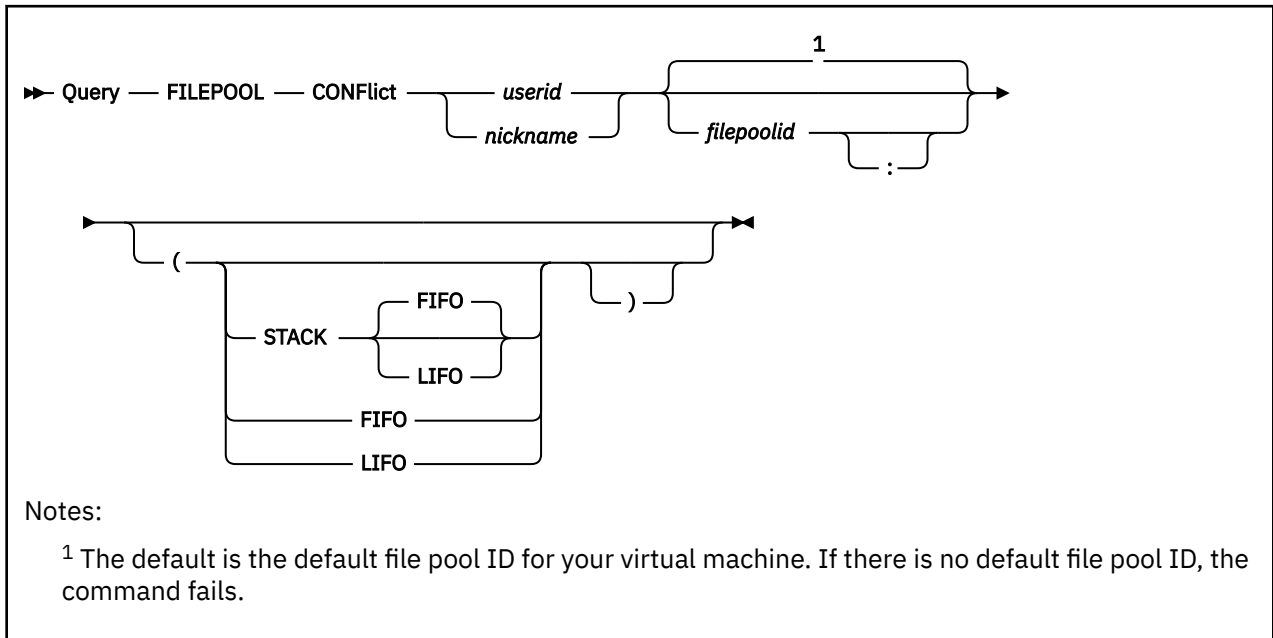
In addition to the messages listed below, this command displays other system messages. These system messages are listed in *z/VM: CMS Commands and Utilities Reference*.

Messages:

## QUERY FILEPOOL CATALOG

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS688E XEDIT option only valid from XEDIT environment [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24]
- DMS1223E There is no default file pool currently defined [RC=40]

# QUERY FILEPOOL CONFLICT



## Authorization

Repository File Pool Administrator or Connected User

## Purpose

Use the QUERY FILEPOOL CONFLICT command to display information about lock conflicts in the specified file pool.

The information returned can assist you with the task of diagnosing why file pool users are experiencing delays in file pool server responses.

## Operands

### *userid*

identifies the user whose lock conflicts are to be displayed.

### *nickname*

identifies a nickname that represents a user ID or a list of user IDs. (Use the NAMES command to define nicknames.)

### *filepoolid*

#### *filepoolid:*

identifies the file pool. If not specified, the default file pool ID for your virtual machine is used.

## Options

### **STACK FIFO**

### **STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. FIFO is the default.

### **FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

### **LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

## Usage Notes

1. You cannot use this command to find your own lock conflicts. If you are able to process the command, you have no conflicts. You must enter this command from another user ID if you want to see the lock conflicts for your own user ID. For more information about detecting lock conflicts, see [z/VM: CMS User's Guide](#).
2. If a *nickname* that represents a list of user IDs is specified, each user ID in that list is processed separately. There is no synchronization of the information across user IDs.
3. Headers are included in the response only if the response is displayed at the terminal (that is, it is not stacked).
4. If you specify a nickname and the NODE tag in the NAMES file indicates the user is on another processor, the LOCALID tag must also be specified.
5. You cannot use this command to find lock conflicts caused by byte range locking of Byte File System (BFS) files.

## Responses

The format of the output is:

Requestor	Holder	Wait	Lock	Lock Type	Identifier
uuuuuuuu	uuuuuuuu	Communication	File	Share	nnnnnnnn
		Checkpoint	Catalog_Row	Excl	
		Lock	Catalog_Block	IShare	
		Catalog_Buffer	Catalog_Index	IExcl	
		Control_Buffer	Storage_Group	Ck_Shr	
		I/O	Recovery	Ck_Up	
		None	File_Space	Ck_Ex	
		Subtask	Directory	Sh_Ix	
		Multi_Event	DIRC_Resource		
		Timer	File_Recall		
		Suspended	Recall_Exit		
		AVSwait	-		
		ESM_Wait	BFS_Cre/De1		
		DFSMS/VM_Wait	BFS_Obj_Write		
		-	BFS_Name_Unal		
		Storage_Wait	BFS_Glob_Stor		
		Token_Callback	BFS_BR_Lock		

For the *Wait*, *Lock*, and *Lock Type* fields, one indication from the list under the respective heading is displayed for each requestor.

General CMS users are allowed to enter this command. When the command is entered from an administration machine, however, an additional column of information is displayed. (The *Identifier* column is the extra column.) The fields for each column header have the following meanings:

### Requestor

this column shows the user ID (uuuuuuuu) that is waiting for a lock being held by the *Holder*

### Holder

this column shows the user ID (uuuuuuuu) that is preventing the *Requestor* from getting a lock. The *Holder* may be waiting for some other user to free a lock.

### Wait

indicates the wait state of the *Holder*. *Wait* can be:

#### Communication

is displayed when the file pool server is waiting for another request from the holder of the lock. (That is, the server is waiting for the user to ask it to do something.)

#### Checkpoint

is displayed if the holder of the lock waiting for an internal checkpoint to complete.

#### Lock

is displayed when the holder of the lock is waiting for another implicit lock that someone else holds. (The file pool server never waits for explicit locks, which are created, for example, by the

CREATE LOCK command or by the DMSCRLOC CSL routine.) For more information on explicit and implicit locks, see [Lock Type](#).

**Catalog\_Buffer**

is displayed when the holder of the lock is waiting for a buffer to hold catalog data to become free.

**Control\_Buffer**

is displayed when the holder of the lock is waiting for a buffer to hold control minidisk data to become free.

**I/O**

is displayed when the holder of the lock is waiting for file pool I/O to complete.

**None**

the holder is not waiting.

**Subtask**

is displayed when the agent sends a request to the repository file pool server, which must be suspended pending the completion of an SFS function (for example, the initial exchange of log names with the user's CRR recovery server). The SFS function is handled as a subtask, which is not directly related to the user's request that resulted in the subtask being initiated. While the subtask is executing, the original user's request is suspended in this wait state.

**Multi\_Event**

is displayed when the CRR recovery server communicates with multiple participating resource managers (such as the repository file pool server) and waits for a response from a participating resource manager.

**Timer**

is displayed when the agent is waiting for a time period to elapse (as specified by the RESYNCINTERVAL startup parameter or defaults to 10 minutes) before the periodic retry of resynchronization in an attempt to complete the update of a protected resource.

**Suspended**

is displayed when the agent is temporarily put into a waiting state by the CRR SUSPEND operator command. This wait state prevents the periodic retry of resynchronization.

**AVSwait**

is displayed when the agent is waiting for a response from AVS. The AVS virtual machine is the agent's communication partner.

**ESM\_Wait**

is displayed when the agent is waiting for an External Security Manager (ESM) response.

**DFSMS/VM\_Wait**

is displayed when the agent is waiting for a DFSMS/VM response.

—

is displayed when the file pool server is at a higher release level than the CMS in your machine, and the wait is a type that is unknown to CMS.

**Storage\_Wait**

is displayed when the agent is waiting for storage to be available on the server.

This wait state applies to all file pools servers.

**Token\_Callback**

is displayed when the file pool server is waiting for the Byte File System (BFS) client machine to respond to a request to invalidate cached file pool information.

**Lock**

is the type of file pool resource for which the *Requestor* has requested a lock. These locks are caused by requests for files and directories. The server, in processing a request for a file, determines which of these locks are needed and then tries to acquire them for the user making the request. All of this is transparent to the user. *Lock* can be:

**File**

is displayed when the requestor is causing the server to try to lock a file.

**Catalog\_Row**

is displayed when the requestor is causing the server to try to lock a catalog row. Information stored in the file pool catalogs is organized into rows (or records) of data. A catalog row might, for example, contain all the information about a file's attributes (record length, number of records, and so on).

**Catalog\_Block**

is displayed when the requestor is causing the server to try to lock a 4K block of catalog information. A catalog block might contain multiple catalog rows.

**Catalog\_Index**

is displayed when the requestor is causing the server to try to lock a catalog index.

**Storage\_Group**

is displayed when the requestor is causing the server to try to lock the entire storage group.

**Recovery**

is displayed when the requestor is causing the server to try to roll back a logical unit or work. In this case, the resources needed to do the rollback are unavailable. (Serialization has occurred during a rollback.)

**File\_Space**

is displayed when the requestor is causing the server to try to lock a file space.

**Directory**

is displayed when the requestor is causing the server to try to lock a directory.

**DIRC\_Resource**

is displayed when the requestor is causing the server to try to lock a DIRCONTROL Resource.

**File\_Recall**

is displayed when the requestor is causing the server to try to lock a File Recall resource.

**Recall\_Exit**

is displayed when the requestor is causing the server to try to lock a Recall Exit resource.

—

is displayed when the SFS file pool server is at a higher release level than the CMS in your machine, and the wait is a type that is unknown to CMS.

**BFS\_Cre/Del**

is displayed when a request for a lock on the object (file, directory, link, or symbolic link) to be created or deleted was denied because the implicit lock was already held on the object in a Byte File System (BFS).

**BFS\_Obj\_Write**

is displayed when the Token Manager requested a WRITE VNODE lock but had to wait for the lock because the implicit lock was already held in a Byte File System (BFS).

**BFS\_Name\_Unal**

is displayed when a request for a lock on an unallocated NAMECAT row was denied because the implicit lock was already held on the row in a Byte File System (BFS).

**BFS\_Glob\_Stor**

is displayed when a request for a lock on the object was denied (or waited for) because an implicit lock was already held on the object in a Byte File System (BFS). This is also known as a Global Storage logical lock conflict.

**BFS\_BR\_Lock**

is displayed when a request for a lock on a file for serializing byte range lock/unlock and file closes was denied (or waited for) because an implicit lock had already been created on that file in a Byte File System (BFS). The request for the lock was from an implicit request.

**Lock Type**

is the type of lock the requestor needs. Some of these are internal lock types the server needs to complete a request. The internal lock types are intended for use by service personnel. *Lock Type* can be:

**Share**

is an internal locking type.

**Excl**

is an internal locking type.

**IShare**

is an internal locking type.

**IExcl**

is an internal locking type.

**Ck\_Shr**

is an explicit (checkout) share lock. (The *Identifier* column contains the internal identifier of the object the requestor is trying to lock.)

**Ck\_Up**

is an explicit (checkout) update lock. (The *Identifier* column contains the internal identifier of the object the requestor is trying to lock.)

**Ck\_Ex**

is an explicit (checkout) exclusive lock. (The *Identifier* column contains the internal identifier of the object the requestor is trying to lock.)

**Sh\_Ix**

is an internal locking type.

**Identifier**

is the internal value the file pool server uses to identify the object that is to be locked. If the requestor is not an administrator, the Identifier column is not displayed. *Identifier* varies depending on what is displayed for *Lock*, as follows:

**When LOCK is:****IDENTIFIER is:****File**

the internal identifier the server uses for the file.

**Catalog\_Row**

the internal row identifier of the desired row.

**Catalog\_Block**

the internal block number of the desired block.

**Catalog\_Index**

null.

**Storage\_Group**

the storage group number.

**Recovery**

null.

**File\_Space**

the internal identifier the server uses for the file space.

**Directory**

the internal identifier the server uses for the directory.

**DIRC\_Resource**

the internal identifier the server uses for the DIRCONTROL resource.

**File\_Recall**

the internal identifier the server uses for the File Recall resource.

**Recall\_Exit**

the internal identifier the server uses for the Recall Exit resource.

**BFS\_Cre/Del**

the internal identifier the server uses for the Byte File System (BFS) object.

**BFS\_Obj\_Write**

the internal identifier the server uses for the Byte File System (BFS) object (also a VNODE).

**BFS\_Name\_Unal**

the internal identifier the server uses for the Byte File System (BFS) NAMECAT row.

**BFS\_Glob\_Stor**

the internal identifier the server uses for the Byte File System (BFS) object.

**BFS\_BR\_Lock**

the internal identifier the server uses for the Byte File System (BFS) file.

### Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in *z/VM: CMS Commands and Utilities Reference*.

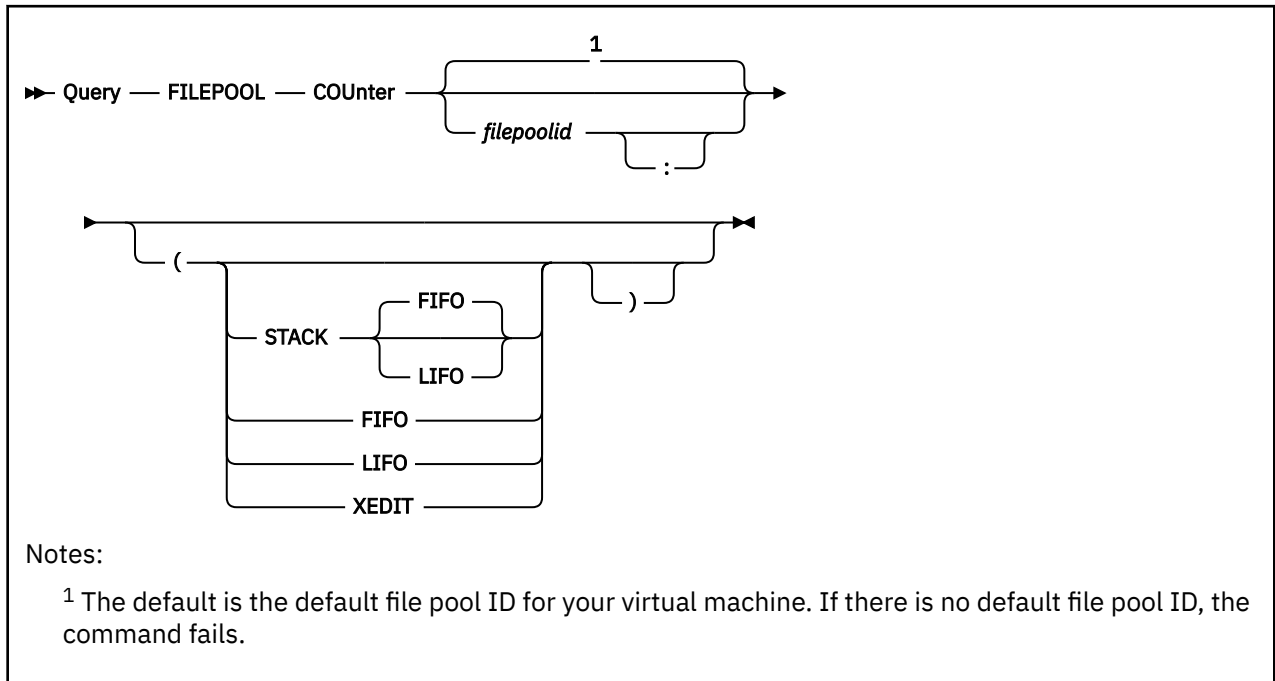
This command internally issues IDENTIFY, FINIS, NUCXDROP, and NUCXLOAD commands. Therefore, messages from those commands can also be displayed. (For more information about those commands, see *z/VM: CMS Commands and Utilities Reference*.)

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS149E Userid *userid* not valid [RC=32]
- DMS386E Missing operand(s) [RC=24]
- DMS389E Invalid filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS637E Missing node ID for the AT operand [RC=24]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=32]
- DMS653E Error executing *command* rc=*nn* [RC=40]
- DMS1223E There is no default file pool currently defined [RC=40]



# QUERY FILEPOOL COUNTER



## Authorization

File Pool Administrator or Connected User

## Purpose

Use the QUERY FILEPOOL COUNTER command to display information about an SFS file pool or CRR recovery server. It returns a logically ordered list of file pool counters.

The information returned can assist you with the task of performance analysis and timing for file pool servers.

Note that this command should not be used by permanent applications to acquire data. The content of the output from this command may change in the future.

## Operands

*filepoolid*

*filepoolid:*

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

## Options

**STACK FIFO**

**STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.

**FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

**LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

**XEDIT**

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

**Usage Notes**

1. The XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set to ON. CMS FULLSCREEN also lets you scroll through the information.
2. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL COUNTER command with the XEDIT option on the command line. When a new file is edited, there is not a chance of accidentally overlaying information already in the file.
3. If the QUERY FILEPOOL COUNTER command is issued from an exec or an assembler program for a file pool that is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
4. If the current release of z/VM is processing prior release file pool information, you will only see the information applicable for that prior release.
5. The CRR COUNTER INFORMATION will not be displayed if the server is running with the NOCRR startup parameter in effect.
6. Counters are reset when file pool server processing is started (by using the FILESERV START command).

**Responses**

The QUERY FILEPOOL COUNTER command displays file pool counter information.

An example of the output is:

```

=====
                        VM9SERV  File Pool Counters
Start-up Date mm/dd/yy          Query Date mm/dd/yy
Start-up Time hh:mm:ss         Query Time  hh:mm:ss
=====
SFS AND BYTE FILE COUNTER INFORMATION

0  Add Minidisk Requests
0  Add Storage Requests
0  Byte File Cancel Requests
0  Byte File Change Access/Modification Time Requests
0  Byte File Change Audit Requests
0  Byte File Change Mode Requests
0  Byte File Change Owner Requests
0  Byte File Check File Accessibility Requests
0  Byte File Close Directory Requests
0  Byte File Close File Requests
0  Byte File Create Block Special File Requests
0  Byte File Create Character Special File Requests
0  Byte File Create Directory Requests
0  Byte File Create External Link Requests
0  Byte File Create Link Requests
0  Byte File Create Named Pipe (FIFO) Requests
0  Byte File Create Regular File Requests
0  Byte File Create Symbolic Link Requests
0  Byte File Lock Byte Requests
0  Byte File Lookup Requests
0  Byte File Makecat Requests
0  Byte File Open Directory Requests
0  Byte File Open File New With Intent Read Requests
0  Byte File Open File New With Intent Write Requests
0  Byte File Open File Read Requests
0  Byte File Open File Write Requests
0  Byte File Pipe Access Requests

```

```

0 Byte File Pipe Close Requests
0 Byte File Pipe Open For Read Requests
0 Byte File Pipe Open For Write Requests
0 Byte File Pipe Read Requests
0 Byte File Pipe Stat Requests
0 Byte File Pipe Utime Requests
0 Byte File Pipe Write Requests
0 Byte File Read Directory Entry Requests
0 Byte File Read File Requests
0 Byte File Read Link Contents Requests
0 Byte File Remove Directory Requests
0 Byte File Rename Requests
0 Byte File Test Locked Bytes Requests
0 Byte File Token Return Requests
0 Byte File Unlink Requests
0 Byte File Unlinked File Cleanup Requests
0 Byte File Unlock Byte Requests
0 Byte File Write File Requests
0 Byte File ZAPCAT Requests
1 Cache Release Requests
0 Change DFSMS Related Attribute Requests
4 Change File Attributes Requests
0 Change Threshold Requests
11 Close Directory Requests
201 Close File Requests
0 Commit Requests
5 Connect Requests
0 Connect User Requests
3 Create Alias Requests
40 Create Directory Requests
0 Create External Object Requests
6 Create File Requests
1 Dataspace Requests
0 Delete Directory Requests
1 Delete File Requests
0 Delete Storage Requests
1 Dirattr Requests
13 File Copy Requests
0 File Pool Control Backup Requests
4 Get Directory Entry Requests
0 Get Directory Requests
7 Grant Administrator Authorization Requests
3 Grant Authorization Requests
23 Grant User Connect Requests
25 Lock Requests
0 Migrate Requests
11 Open Directory Requests
0 Open File CreateMig Requests
6 Open File New Requests
3 Open File Read Requests
7 Open File Replace Requests
191 Open File Write Requests
0 Precoordination Requests
0 Prepare Requests
7 Query Accessors Requests
51 Query Administrator Requests
0 Query Connected Users Requests
0 Query Dataspace Requests
4 Query Disable Requests
4 Query Enrolled Users Requests
24 Query File Pool Requests
2 Query Lock Conflicts Requests
10 Query User Space Requests
0 Query User Storage Group Requests
0 Read File Requests
0 Recall Requests
0 Recovery Close Catalog Requests
0 Recovery Get Catalog Requests
0 Recovery Open Catalog Requests
0 Recovery Put Catalog Requests
11 Refresh Directory Requests
0 Release Blocks Requests
2 Relocate Requests
0 Rename Requests
0 Rename Userid Requests
2 Revoke Administrator Authorization Requests
7 Revoke Authorization Requests
0 Revoke User Requests
0 Rollback Requests
0 Send DFSMS Data Requests
0 Set Reference Date Requests
1 SFS Send User Data Requests

```

## QUERY FILEPOOL COUNTER

```
0 Temporary Close Requests
25 Unlock Requests
0 Write Accounting Requests
16 Write File Requests
733 Total File Pool Requests
0 Total Byte File File Pool Requests
26330 File Pool Request Service Time (msec)
733 Local File Pool Requests
0 Remote File Pool Requests

23 Alias Definitions Examined
6 Alias Definitions Updated

5674 SAC Calls

7 Checkpoints Taken
1873 Checkpoint Time (msec)

0 Security Manager Exit Calls
0 Security Manager Exit Time (msec)
0 External Security Manager Exit Calls
0 External Security Manager Exit Time (msec)

0 DMSSFSEX Exit Calls
0 DMSSFSEX Exit Time (msec)

0 Recall DFSMS File Exit Calls
0 Recall DFSMS File Exit Time (msec)
0 Other DFSMS Exit Calls
0 Other DFSMS Exit Time (msec)

0 Storage Group Explicit Lock Conflicts
0 File Space Explicit Lock Conflicts
0 Directory Explicit Lock Conflicts
0 File Explicit Lock Conflicts
0 Storage Group Logical Lock Conflicts
0 File Space Logical Lock Conflicts
0 Directory Logical Lock Conflicts
0 File Logical Lock Conflicts
0 Catalog Lock Conflicts
0 DIRCONTROL Resource Lock Conflicts
0 File Recall Lock Conflicts
0 Recall Exit Lock Conflicts
0 Total Lock Conflicts
0 Lock Wait Time (msec)
0 Locks Denied Due to Timeout

0 Deadlocks
0 Rollbacks Due to Deadlock
11 LUW Rollbacks

0 Byte File Directory Creation/Deletion Logical Lock
Conflicts
0 Byte File File Logical Lock Conflicts
0 Byte File Global Storage Logical Lock Conflicts
0 Byte File NAMECAT Unallocated Logical Lock Conflicts
0 Byte File Token Manager Logical Lock Conflicts
0 Total Byte File Lock Conflicts
0 Byte File Byte Range Lock Waits

0 Byte File Token Conflicts Causing Callbacks
0 Byte File Callback Wait Time (msec)
0 Byte File Token Callback Timeout Retries
0 Byte File Token Callback Requestor Retries

0 Byte File Logical Lock Retries
0 Byte File Logical Lock Retries Exceeded

441 Begin LUWs
26640 Agent Holding Time (msec)

24 QSAM Requests
320 QSAM Time (msec)

10 File Blocks Read
266 File Blocks Written
587 Catalog Blocks Read
401 Catalog Blocks Written
635 Control Minidisk Blocks Read
74 Control Minidisk Blocks Written
4 Log Blocks Read
```

```

874 Log Blocks Written
2851 Total DASD Block Transfers

  10 BIO Requests to Read File Blocks
223 BIO Requests to Write File Blocks
587 BIO Requests to Read Catalog Blocks
359 BIO Requests to Write Catalog Blocks
  28 BIO Requests to Read Control Minidisk Blocks
  45 BIO Requests to Write Control Minidisk Blocks
   4 BIO Requests to Read Log Blocks
702 BIO Requests to Write Log Blocks
1958 Total BIO Requests
25691 Total BIO Request Time (msec)

  10 I/O Requests to Read File Blocks
224 I/O Requests to Write File Blocks
587 I/O Requests to Read Catalog Blocks
359 I/O Requests to Write Catalog Blocks
  28 I/O Requests to Read Control Minidisk Blocks
  45 I/O Requests to Write Control Minidisk Blocks
   4 I/O Requests to Read Log Blocks
702 I/O Requests to Write Log Blocks
1959 Total I/O Requests
=====
CRR COUNTER INFORMATION

  0 Get Capability Requests
  0 Get Logname Requests
  0 Get LUWID Requests
  0 Resync Init Requests
  0 Resync Query Direction Requests
  0 Resync Protocol Violations Requests
  0 Write Log Requests
  0 Total CRR Requests
  0 CRR Request Service Time (msec)
  0 Syncpoints
  0 Syncpoint Time (msec)
  0 Participating Resources
  0 Log Checkpoints taken
  0 Log I/O Requests
  0 BIO Request Time (msec)
=====

```

See [SFS AND BYTE FILE COUNTER INFORMATION](#) and [CRR COUNTER INFORMATION](#) of the QUERY FILEPOOL REPORT command for a description of the counters.

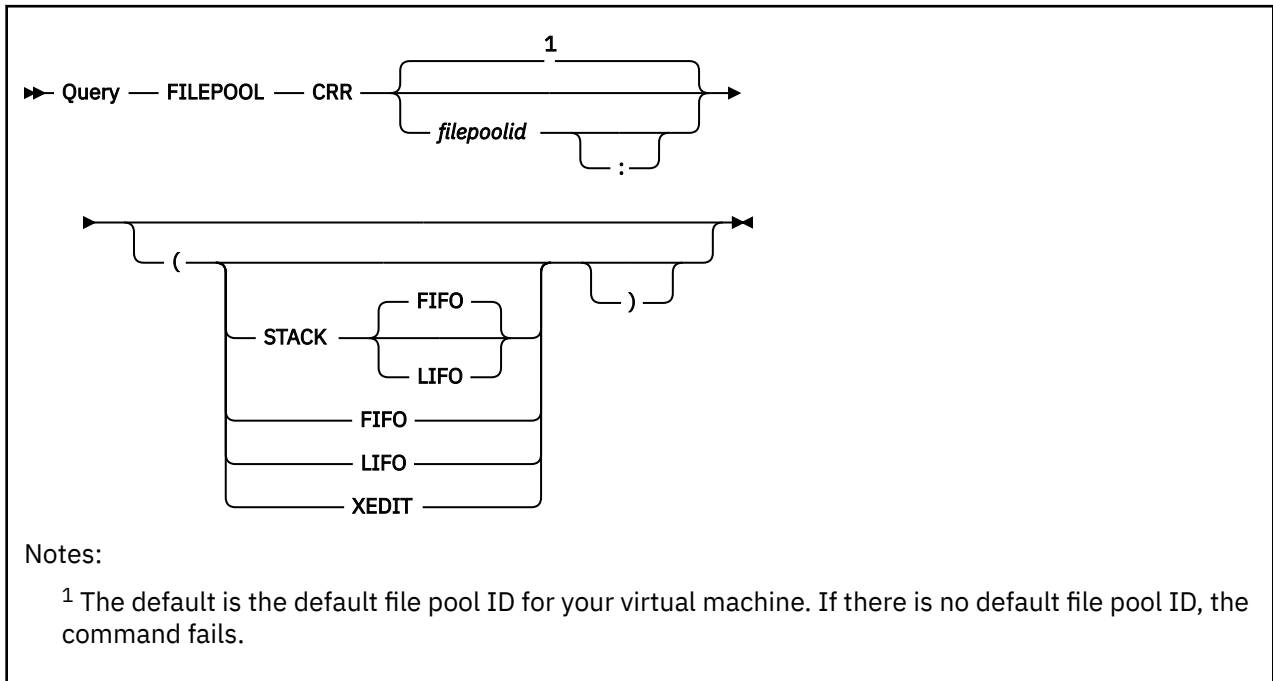
## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS688E XEDIT option only valid from XEDIT environment [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24]
- DMS1223E There is no default file pool currently defined [RC=40]

## QUERY FILEPOOL CRR (CRR only)



### Authorization

CRR Administrator

### Purpose

Use the QUERY FILEPOOL CRR command to display CRR counter information.

The information returned can assist you with the task of performance analysis and timing for CRR recovery servers.

Note that this command should not be used by permanent applications to acquire data. The content of the output from this command may change in the future.

### Operands

*filepoolid*

*filepoolid:*

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

### Options

**STACK FIFO**

**STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.

**FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

**LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

**XEDIT**

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

**Usage Notes**

1. If the server is running with the NOCRR startup parameter in effect, message DMS3484E File pool server is not a CRR recovery server will be displayed.
2. The XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set to ON. CMS FULLSCREEN also lets you scroll through the information.
3. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL CRR command with the XEDIT option on the command line. When a new file is edited, there isn't a chance of accidentally overlaying information already in the file.
4. If the QUERY FILEPOOL CRR command is issued from an exec or an assembler program for a file pool that is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
5. If the current release of z/VM is processing prior release file pool information, you will only see the information applicable for that prior release.
6. Counters are reset when CRR recovery server processing is started (by using the FILESERV START command).

**Responses**

The QUERY FILEPOOL CRR command displays CRR counter information.

An example of the output is:

```

VM9SERV File Pool CRR Counters

Start-up Date mm/dd/yy          Query Date mm/dd/yy
Start-up Time hh:mm:ss         Query Time hh:mm:ss
=====
CRR COUNTER INFORMATION

      0 Get Capability Requests
    1815 Get Logname Requests
    2249 Get LUWID Requests
      0 Resync Init Requests
      0 Resync Query Direction Requests
      0 Resync Protocol Violations Requests
      3 Write Log Requests
    4067 Total CRR Requests
    70547 CRR Request Service Time (msec)
      1 Syncpoints
    282672372 Syncpoint Time (msec)
      2 Participating Resources
      1 Log Checkpoints Taken
    4565 Log I/O Requests
    68037 BIO Request Time (msec)
=====

```

For a description of the fields, see [CRR COUNTER INFORMATION](#) of the command “[QUERY FILEPOOL REPORT](#)” on page 592.

**Messages and Return Codes**

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

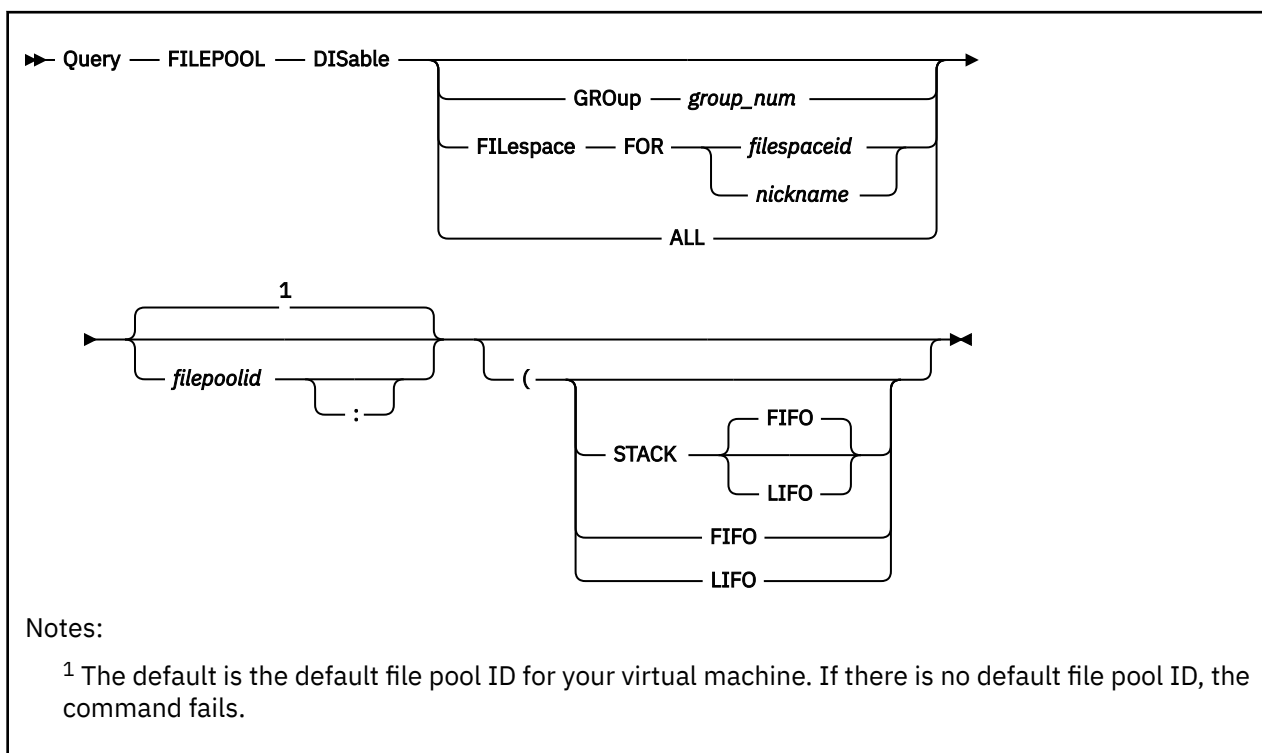
## QUERY FILEPOOL CRR

### Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS688E XEDIT option only valid from XEDIT environment [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24]
- DMS1223E There is no default file pool currently defined [RC=40]
- DMS3484E File pool server is not a CRR recovery server [RC=76]



## QUERY FILEPOOL DISABLE



### Authorization

File Pool Administrator or Connected User

### Purpose

Use the QUERY FILEPOOL DISABLE command to determine disable lock information. There are four versions of this command:

1. Use QUERY FILEPOOL DISABLE with no keyword specified to determine if your file space or its owning storage group is disabled. For more information, see Usage Note “2” on page 578.
2. Use QUERY FILEPOOL DISABLE GROUP to determine if a storage group is disabled.
3. Use QUERY FILEPOOL DISABLE FILESPACE to determine if a file space or its owning storage group is disabled.
4. Use QUERY FILEPOOL DISABLE ALL to determine all outstanding disables on all storage groups and all file spaces.

A user with file pool administration authority can enter all four versions of this command.

Any file pool user can enter QUERY FILEPOOL DISABLE with no keyword on a file space for their user ID or QUERY FILEPOOL DISABLE FILESPACE for his or her own file space, another user's file space, or a Byte File System (BFS).

### Operands

#### GRoup *group\_num*

identifies the number of a storage group for which you want information. For *group\_num*, specify a value between 2 and 32767, but not greater than the MAXDISKS parameter of the file pool. MAXDISKS is an SFS file pool server parameter in the POOLDEF file. For more information about MAXDISKS, see “FILESERV GENERATE” on page 513.

### **FILESPACE FOR *filepaceid***

identifies the file space for which you want information.

### **FILESPACE FOR *nickname***

identifies a nickname that represents a single user ID or a BFS. The nickname may not represent a list. (Use the NAMES command to define nicknames).

### **ALL**

indicates all outstanding file space and storage group disables for a file pool are to be displayed.

### ***filepoolid***

#### ***filepoolid:***

identifies the file pool being queried. This operand is optional. If not specified, the default file pool for your virtual machine is used.

## **Options**

### **STACK FIFO**

### **STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.

### **FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

### **LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

## **Usage Notes**

1. This command might be used to determine disable information when the results are lost or unknown of one of the following:
  - DISABLE operator command
  - FILEPOOL DISABLE command
  - Disable File Space (DMSDISFS) CSL routine
  - Disable Storage Group (DMSDISSG) CSL routine
  - FILEPOOL RELOAD
  - FILEPOOL UNLOAD
  - FILEPOOL BACKUP and FILEPOOL RESTORE commands
  - FILEPOOL RENAME command

For more information on disable locks on a file space or storage group, refer to the respective command in this document or the respective CSL routine listed in [z/VM: CMS Callable Services Reference](#).

2. The following are all valid ways to query your own disables: QUERY FILEPOOL DISABLE, QUERY FILEPOOL DISABLE *filepoolid*, QUERY FILEPOOL DISABLE FILESPACE FOR *myuserid*, QUERY FILEPOOL DISABLE FILESPACE FOR *myuserid filepoolid*.
3. If the QUERY FILEPOOL DISABLE command does not resolve what type of lock exists, there may be an explicit lock on the file or directory.

For a possible explicit lock on the file or directory, see the QUERY LOCK command in [z/VM: CMS Commands and Utilities Reference](#).

4. If the file space and its owning storage group was disabled because of a function, the function that created the disables will be returned.

For example, an attempt of the FILEPOOL RENAME command can cause the file space and its owning storage group to be disabled; in these cases RENAME will be returned as the creator of the disable. Note that RENAME is currently the only valid function name. For more information on RENAME, refer to the usage notes of the FILEPOOL RENAME command.

5. If the QUERY FILEPOOL DISABLE command is issued from an exec or an assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
6. To get the equivalent information from an application program, use the Query Filepool Disable CSL routine DMSQFPDS.
7. If you specify a nickname and the NODE tag, in the NAMES file, it indicates the user is on another processor. The LOCALID tag must also be specified. See the NAMES files description for more information.

**Examples**

The user machine displays responses describing the disable status of the object being queried.

EXAMPLE 1: For a QUERY FILEPOOL DISABLE, entered from user ID JONES, the output might look like:

Object Type	Object ID	Disabled	Creator	Mode	Link Status
GROUP	4	YES	CHERIE	SHARE	LINKED
FILESPACE	JONES	YES	ANDY	SHARE	-

This tells us the file space JONES was disabled in share mode by user ANDY. Also, it tells us user CHERIE also disabled storage group 4 in share mode and kept the minidisks associated with storage group 4.

EXAMPLE 2: For a QUERY FILEPOOL DISABLE GROUP 4, the output might look like:

Object Type	Object ID	Disabled	Creator	Mode	Link Status
GROUP	4	YES	JONES	EXCLUSIVE	DETACHED

This tells us user JONES disabled storage group 4 in exclusive mode and detached the minidisks associated with storage group 4.

Or,

Object Type	Object ID	Disabled	Creator	Mode	Link Status
GROUP	4	YES	JONES	SHARE	LINKED

This tells us user JONES disabled storage group 4 in share mode and kept the minidisks associated with storage group 4 linked.

Or,

Object Type	Object ID	Disabled	Creator	Mode	Link Status
GROUP	4	NO	-	-	-

This tells us storage group 4 was not disabled.

Or,

Object Type	Object ID	Disabled	Creator	Mode	Link Status
GROUP	4	YES	RENAME	IEXCLUSIVE	LINKED

This tells us storage group 4 was disabled in intent exclusive mode. This is an internal disable created by the RENAME function (see Usage Note “4” on page 578 for more information on the RENAME function).

EXAMPLE 3: For a QUERY FILEPOOL DISABLE FILESPACE FOR JONES, the output might look like:

Object Type	Object ID	Disabled	Creator	Mode	Link Status
GROUP	4	YES	CHERIE	SHARE	LINKED
FILESPACE	JONES	YES	CHERIE	SHARE	-
FILESPACE	JONES	YES	ED	SHARE	-
FILESPACE	JONES	YES	SMITH	SHARE	-

## QUERY FILEPOOL DISABLE

This tells us the file space JONES was disabled in share mode by user CHERIE, user ED, and user SMITH. Also, it tells us user CHERIE also disabled storage group 4 in share mode and kept the minidisks associated with storage group 4.

Or,

Object Type	Object ID	Disabled	Creator	Mode	Link Status
FILESPACE	JONES	YES	MARY	EXCLUSIVE	-

This tells us the file space JONES was disabled in exclusive mode by user MARY.

Or,

Object Type	Object ID	Disabled	Creator	Mode	Link Status
GROUP	3	YES	JACK	EXCLUSIVE	LINKED

This tells us the file space JONES was not disabled but its owning storage group 3 was disabled by user JACK in exclusive mode and user JACK kept the minidisks associated with storage group 3 linked.

Or,

Object Type	Object ID	Disabled	Creator	Mode	Link Status
FILESPACE	JONES	NO	-	-	-

This tells us the file space JONES was not disabled and its owning storage group was not disabled; no disables exist.

Or,

Object Type	Object ID	Disabled	Creator	Mode	Link Status
FILESPACE	JONES	YES	RENAME	EXCLUSIVE	-
GROUP	3	YES	RENAME	IEXCLUSIVE	LINKED

This tells us the file space JONES was disabled in exclusive mode and its owning storage group 3 was disabled in intent exclusive mode. These are internal disables created by the RENAME function (see Usage Note “4” on page 578 for more information on the RENAME function).

EXAMPLE 4: For a QUERY FILEPOOL DISABLE ALL, the output might look like:

Object Type	Object ID	Disabled	Creator	Mode	Link Status
GROUP	4	YES	BROWN	SHARE	LINKED
GROUP	3	YES	MARYE	EXCLUSIVE	DETACHED
GROUP	2	YES	MICHAEL	SHARE	LINKED
FILESPACE	MARY	YES	DOE	SHARE	-
FILESPACE	JONES	YES	RENAME	EXCLUSIVE	-
GROUP	5	YES	RENAME	IEXCLUSIVE	LINKED
FILESPACE	JOHN	YES	SMITH	SHARE	-
FILESPACE	ANNIE	YES	TERRI	SHARE	-
FILESPACE	JOHN	YES	TERRI	SHARE	-
FILESPACE	ZORRO	YES	TERRI	SHARE	-

This tells us the default file pool, which was previously set by the SET FILEPOOL command contains the following disables:

- storage groups 4, 3, and 2 are disabled by users BROWN, MARYE, and MICHAEL, respectively.
- file space MARY is disabled by user DOE.
- file space JONES and storage group 5 are disabled by the RENAME function. (See Usage Note “4” on page 578 for more information on the RENAME function.)
- file space JOHN is disabled by users SMITH and TERRI.
- file spaces ANNIE and ZORRO are disabled by user TERRI.

As well as applicable mode and link status information. Or,

No disable locks exist on any file space or on any storage group in file pool FPOOL:

This tells us no file space or storage group disables exist on the default file pool FPOOL: (which was previously set by the SET FILEPOOL command). You will get a return code of 0 if the response is typed or a return code of 6 if the response was to be stacked (indicating no data was stacked).

## Responses

Depending on which version of this command is entered, a single response or multiple combinations of these responses might be displayed:

Object Type	Object ID	Disabled	Creator	Mode	Link Status
GROUP	group_num	YES	userid	mode	nnnnnnnn
GROUP	group_num	NO	-	-	-
FILESPACE	userid1	YES	userid2	mode	-
FILESPACE	userid	NO	-	-	-
FILESPACE	userid	YES	function	mode	-
GROUP	group_num	YES	function	mode	nnnnnnnn

**Note:** The header is generated only if output is displayed at the terminal (that is, it is not stacked).

where

### Object Type

this column shows the type of object. It can be either FILESPACE or GROUP.

### Object ID

this column shows the *userid* (in case of Object type FILESPACE) or *group\_num* (in case of Object type GROUP).

### Disabled

this column shows whether the object (FILESPACE or GROUP) is disabled (YES) or not disabled (NO).

### Creator

this column indicates the user ID or function (see Usage Note “4” on page 578 for more information on function) who has the FILESPACE *userid* or GROUP *group\_num* disabled.

‘-’ will be displayed if the file space or the storage group is not disabled.

**Note:** If the DISABLE operator command was used to disable the object and an *owner* was specified on that command, the creator's user ID is the *owner*, not the user ID that entered the disable.

### Mode

this column indicates the mode. *Mode* can be:

#### SHARE

is displayed when a share disable lock exists on the object. A share disable lock allows users to read, but not to modify, items in the locked object.

#### EXCLUSIVE

is displayed when an exclusive disable lock exists on the object. An exclusive disable lock prevents anyone else but the Creator from reading or modifying the items in the locked object.

#### IEXCLUSIVE

is displayed when an intent exclusive disable lock exists on the object. An intent exclusive disable lock could prevent anyone else but the Creator from reading or modifying the items in the locked object. It is an internal lock.

#### - (hyphen)

is displayed when the object was not disabled.

#### \* (asterisk)

is displayed when the SFS file pool server is at a higher release level than the CMS in your machine, and the mode is a type unknown to CMS.

### Link Status

this column indicates the link status and is only applicable to Object type GROUP.

#### DETACHED

is displayed when the minidisks associated with the disabled storage group were detached.

### LINKED

is displayed when the links to the minidisks associated with the storage group were kept.

### - (hyphen)

is displayed in the case of Object type FILESPACE, or when GROUP *group\_num* was not disabled.

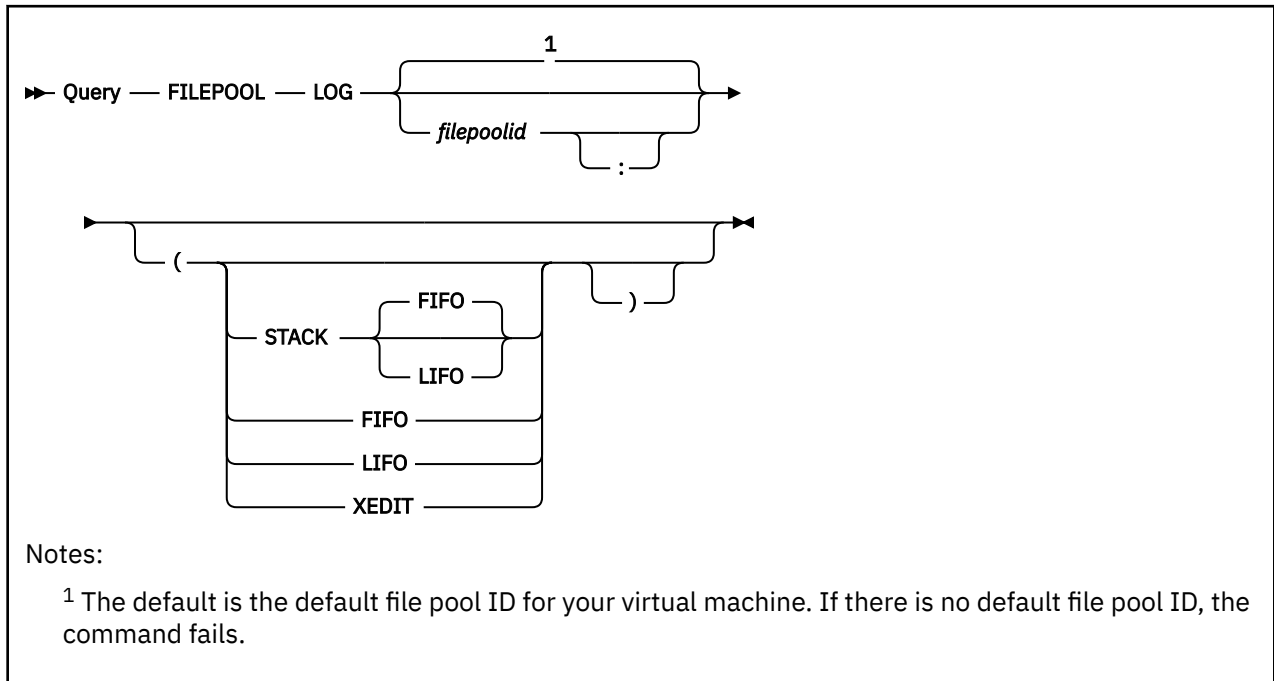
## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Invalid filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's userid* NAMES file [RC=32]
- DMS653E Error executing *command rc=nn* [RC=40]
- DMS1209E Nickname *nickname* resolved to more than one userid; query can be performed only on one userid at a time [RC=88]
- DMS1223E There is no default file pool currently defined [RC=40]
- DMS1239E You are not authorized to enter this request for *GROUP* or *ALL* [RC=76]
- DMS2023E File pool *filepoolid* does not support the requested QUERY command [RC=88]
- DMS3511E Specified storage group number *storage group number* is invalid [RC=24]
- DMS3519E Storage group *storage group number* does not exist in file pool *filepoolid* [RC=40]
- DMS3711E User *filespaceid* not enrolled in the file pool *filepoolid* [RC=40]

## QUERY FILEPOOL LOG



### Authorization

File Pool Administrator or Connected User

### Purpose

Use the QUERY FILEPOOL LOG command to display information about the SFS log minidisks.

The information returned can assist you with the task of performance analysis and timing for file pool servers. The information returned is generally not useful for CRR recovery servers or dedicated FIFO servers.

Note that this command should not be used by permanent applications to acquire data. The content of the output from this command may change in the future.

### Operands

*filepoolid*

**filepoolid:**

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

### Options

**STACK FIFO**

**STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.

**FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

**LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

**XEDIT**

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

**Usage Notes**

1. The XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set to ON. CMS FULLSCREEN also lets you scroll through the information.
2. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL LOG command with the XEDIT option on the command line. When a new file is edited, there isn't a chance of accidentally overlaying information already in the file.
3. For the LOG INFORMATION, the Backup Threshold, Date of Last Control Backup, and Time of Last Control Backup will be not be displayed if the SFS file pool server is running with the NOBACKUP startup parameter in effect.
4. If the current release of z/VM is processing prior release file pool information, you will only see the information applicable for that prior release.
5. If the QUERY FILEPOOL LOG command is issued from an exec or an assembler program for a file pool that is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.

**Responses**

The QUERY FILEPOOL LOG command displays information about the SFS log minidisks.

An example of the output is:

```

VM9SERV File Pool Logs

Start-up Date mm/dd/yy          Query Date mm/dd/yy
Start-up Time hh:mm:ss         Query Time hh:mm:ss
=====
LOG INFORMATION

First Log Virtual Address      0301
Second Log Virtual Address     0302

      290 Number of Log Minidisk 4K Blocks
      75% Percent(%) of Log Space Used
      95% LUW Rollback/Suspend Threshold (% Log Space)
      80% Backup Threshold (% Log Space)
04/14/92 Date of Last Control Backup
12:05:27 Time of Last Control Backup
=====

```

For a description of the fields, see [LOG INFORMATION](#) of the command [“QUERY FILEPOOL REPORT”](#) on [page 592](#).

**Messages and Return Codes**

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

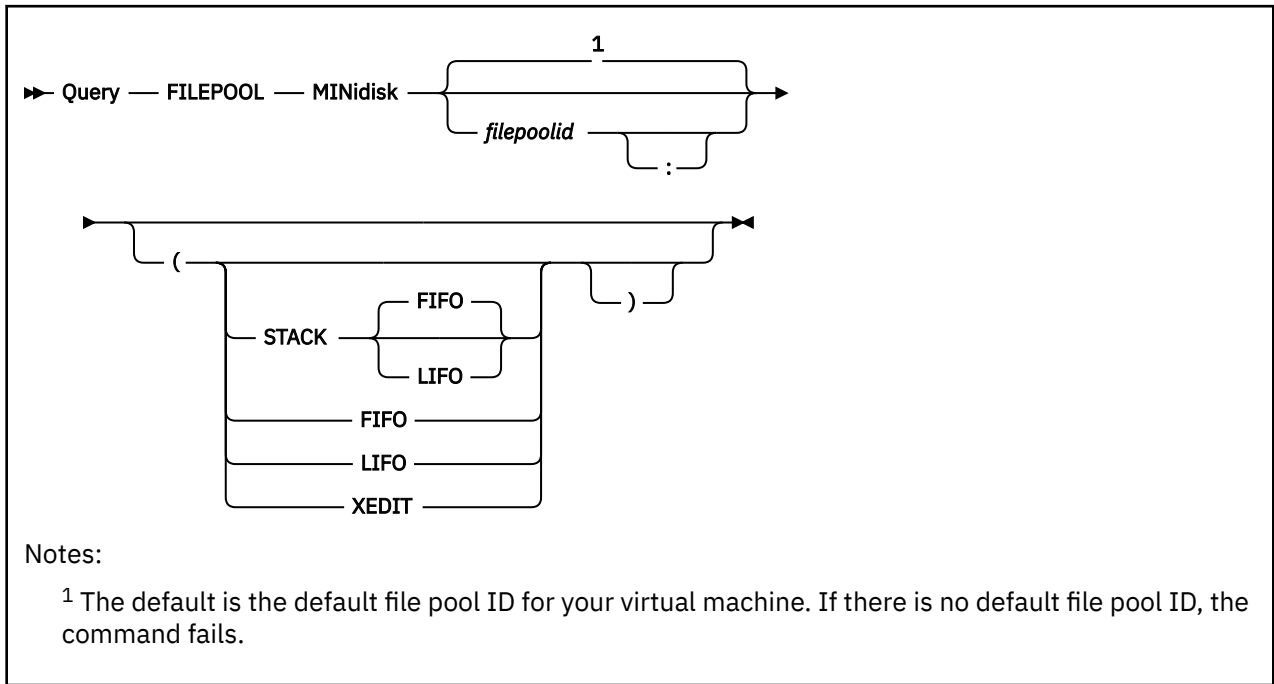
Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]



- DMS389E Incorrect filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS688E XEDIT option only valid from XEDIT environment [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24]
- DMS1223E There is no default file pool currently defined [RC=40]

## QUERY FILEPOOL MINIDISK



### Authorization

File Pool Administrator or Connected User

### Purpose

Use the QUERY FILEPOOL MINIDISK command to display all minidisk information. This includes the maximum number of minidisks and the actual number used, storage group minidisk information, log minidisk data, and control minidisk data.

The information returned can assist you with the task of performance analysis and timing for file pool servers.

Note that this command should not be used by permanent applications to acquire data. The content of the output from this command may change in the future.

### Operands

*filepoolid*

***filepoolid:***

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

### Options

**STACK FIFO**

**STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.

**FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

**LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

**XEDIT**

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

**Usage Notes**

1. The XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set to ON. CMS FULLSCREEN also lets you scroll through the information.
2. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL MINIDISK command with the XEDIT option on the command line. When a new file is edited, there is no chance of accidentally overlaying information already in the file.
3. If the QUERY FILEPOOL MINIDISK command is issued from an exec or an assembler program for a file pool that is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
4. File pool administration authority is required when the user is at the current release of z/VM and is requesting information from a file pool server that is on a release prior to VM/ESA version 1 release 2.0.
5. If the current release of z/VM is processing prior release file pool information, you will only see the information applicable for that prior release.

**Responses**

The QUERY FILEPOOL MINIDISK command displays all minidisk information.

An example of the output is:

```

VM9SERV File Pool Minidisks

Start-up Date dd/mm/yy          Query Date dd/mm/yy
Start-up Time hh:mm:ss         Query Time hh:mm:ss
=====
FILE POOL MINIDISK INFORMATION

      10 Maximum Number of Minidisks
       5 Minidisks in Use
=====
STORAGE GROUP MINIDISK INFORMATION

Storage  Minidisk  4K Blocks  4K Blocks  Virtual
Group No. Number   In-Use     Free       Address
-----
   1      1      124 - 42%   168        0303
   1      2      101 - 35%   191        0304
   1      3       71 - 24%   221        0305
   2      4      149 - 51%   143        0306
   2      5      115 - 10%  1075       0193
=====
STORAGE GROUP MINIDISK TOTALS

Storage  4K Blocks  4K Blocks
Group No. In-Use     Free
-----
   1      296 - 34%   580
   2      264 - 18%  1218
=====
LOG INFORMATION

First Log Virtual Address      0301
Second Log Virtual Address     0302

290 Number of Log Minidisk 4K Blocks
75% Percent(%) of Log Space Used
95% LUW Rollback/Suspend Threshold (% Log Space)
80% Backup Threshold (% Log Space)
dd/mm/yy Date of Last Control Backup

```

## QUERY FILEPOOL MINIDISK

```
hh:mm:ss Time of Last Control Backup
=====
CONTROL MINIDISK INFORMATION

Control Minidisk Virtual Address      0300
      1361 Number of Control Minidisk 512 Blocks
=====
```

See the following sections of the QUERY FILEPOOL REPORT command for a description of the fields:

- [FILE POOL OVERVIEW INFORMATION](#)
- [STORAGE GROUP MINIDISK INFORMATION](#)
- [STORAGE GROUP MINIDISK TOTALS](#)
- [LOG INFORMATION](#)
- [CONTROL MINIDISK INFORMATION](#)

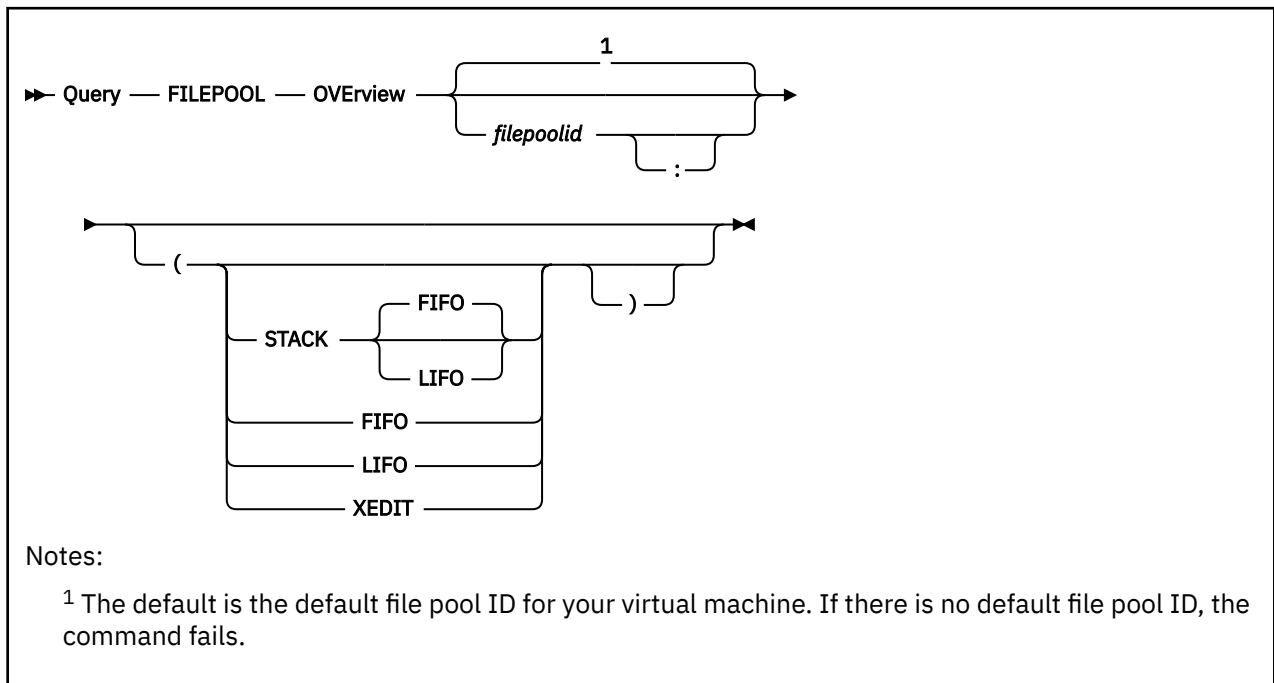
### Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS688E XEDIT option only valid from XEDIT environment [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24]
- DMS1223E There is no default file pool currently defined [RC=40]

## QUERY FILEPOOL OVERVIEW



### Authorization

File Pool Administrator or Connected User

### Purpose

Use the QUERY FILEPOOL OVERVIEW command to return overview information about a specified file pool. This includes information that allows a comparison between file pool limits and the present highest values (for example, Virtual Storage, Connections, Agents).

The information returned can assist you with the task of performance analysis and timing for file pool servers.

Note that this command should not be used by permanent applications to acquire data. The content of the output from this command may change in the future.

### Operands

*filepoolid*

*filepoolid:*

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

### Options

**STACK FIFO**

**STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.

**FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

**LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

**XEDIT**

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

**Usage Notes**

1. The XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set to ON. CMS FULLSCREEN also lets you scroll through the information.
2. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL OVERVIEW command with the XEDIT option on the command line. When a new file is edited, there isn't a chance of accidentally overlaying information already in the file.
3. If the QUERY FILEPOOL OVERVIEW command is issued from an exec or an assembler program for a file pool that is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
4. If the current release of z/VM is processing prior release file pool information, you will only see the information applicable for that prior release.

**Responses**

The QUERY FILEPOOL OVERVIEW command displays file pool overview information. An example of the output is:

```

VM9SERV File Pool Overview

Start-up Date mm/dd/yy          Query Date mm/dd/yy
Start-up Time hh:mm:ss         Query Time hh:mm:ss
=====
FILE POOL OVERVIEW INFORMATION

 6291456 Virtual Storage Size in Bytes (6144 in KB)
  4057 Virtual Storage Highest Value in KB
    0 Virtual Storage Requests Denied
    0 Virtual Storage Reclaim Value

 25 Maximum Number of Connections
 11 Connections Highest Value

 16 Total Number of Agents
  2 Active Agents Highest Value

 10 Maximum Number of Storage Groups
  2 Storage Groups in Use

 10 Maximum Number of Minidisks
  5 Minidisks in Use

1769472 Potential Addressable 4K Blocks in File Pool
 2358 Defined Addressable 4K Blocks in File Pool
1767114 Undefined Addressable 4K Blocks in File Pool
=====

```

In the FILE POOL OVERVIEW INFORMATION you will see some maximum numbers. These were established for the file pool during file pool generation (or regeneration) by the FILESERV GENERATE or FILESERV REGENERATE commands. To exceed any of these maximums, you must regenerate the file pool.

For a description of the fields, see [FILE POOL OVERVIEW INFORMATION](#) of the command “[QUERY FILEPOOL REPORT](#)” on page 592.

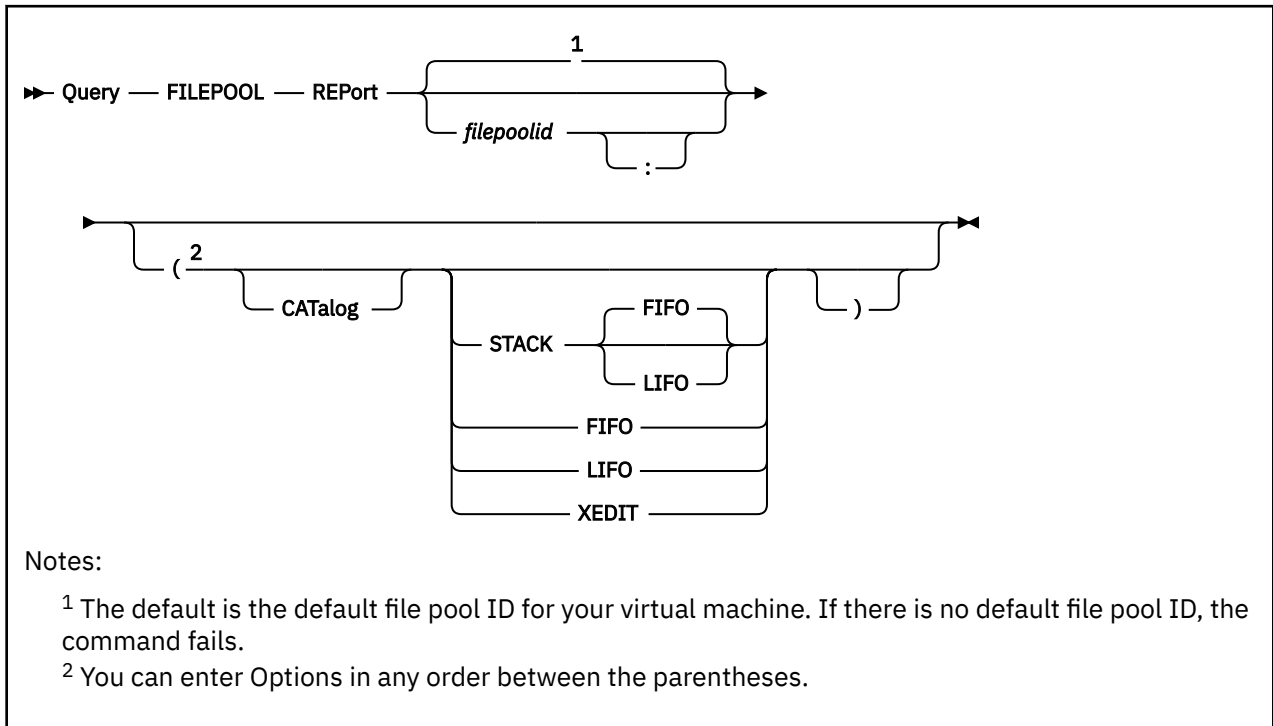
## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS688E XEDIT option only valid from XEDIT environment [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24]
- DMS1223E There is no default file pool currently defined [RC=40]

## QUERY FILEPOOL REPORT



### Authorization

File Pool Administrator or Connected User

### Purpose

Use the QUERY FILEPOOL REPORT command to display information about a file pool. The display also includes information about a file pool server processing against the file pool.

The contents of this query are similar to that of the QUERY FILEPOOL STATUS command. Additional information, as well as a more usable output format, are provided. The most commonly required data appears first in this report, while the counter data is last.

The information returned can assist you with the task of performance analysis and timing for file pool servers.

Note that this command should not be used by permanent applications to acquire data. The content of the output from this command may change in the future.

### Operands

*filepoolid*  
*filepoolid:*

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

### Options

**STACK FIFO**  
**STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.



**FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

**LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

**XEDIT**

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

**CATalog**

indicates you want information about the file pool catalogs to be displayed. File pool administration authority is required **only** when the CATALOG option is specified.

**Usage Notes**

1. If you do not need all the information that is generated by this command, but would like to see specific information only, refer to the following table to determine what QUERY FILEPOOL command will give you the information you need.

This Cross Reference table matches QUERY FILEPOOL commands with the type of information they produce. The contents of these queries are similar to that of the QUERY FILEPOOL STATUS and QUERY FILEPOOL REPORT commands.

File pool administration authority is required ONLY for acquiring CATALOG SPACE INFORMATION.

*Table 33. QUERY FILEPOOL Command Information Cross Reference*

	QUERY FILEPOOL REPORT	QUERY FILEPOOL OVERVIEW	QUERY FILEPOOL STORGRP	QUERY FILEPOOL MINIDISK	QUERY FILEPOOL AGENT	QUERY FILEPOOL LOG	QUERY FILEPOOL CATALOG	QUERY FILEPOOL COUNTER	QUERY FILEPOOL CRR
Filepool name, Start-up & Query Date/Time	X	X	X	X	X	X	X	X	X
FILE POOL OVERVIEW INFORMATION	X	X		X					
STORAGE GROUP MINIDISK INFORMATION	X			X					
STORAGE GROUP MINIDISK TOTALS	X		X	X					
AGENT INFORMATION	X				X				
LOG INFORMATION	X			X		X			
CONTROL MINIDISK INFORMATION	X			X					
CATALOG SPACE INFORMATION	X						X		
SFS AND BYTE FILE COUNTER INFORMATION	X							X	
CRR COUNTER INFORMATION	X							X	X

2. Because of the large amount of information returned, the XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set to ON. CMS FULLSCREEN also lets you scroll through the information.
3. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL REPORT command with the XEDIT option on the command line. When a new file is edited, there is no chance of accidentally overlaying information already in the file.

## QUERY FILEPOOL REPORT

4. For the LOG INFORMATION, the Backup Threshold and Date and Time of Last Control Backup will not be displayed if the SFS file pool server is running with the NOBACKUP startup parameter in effect.
5. CATALOG SPACE INFORMATION will be displayed only when requested using the CATALOG option. Use this option with discretion as considerable processing may be required to obtain the requested information.  
  
The CATALOG option is generally not useful for CRR recovery servers.
6. If the QUERY FILEPOOL REPORT command is issued from an exec or an assembler program for a file pool that is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
7. If the current release of z/VM is processing prior release file pool information, you will only see the information applicable for that prior release.
8. The CRR COUNTER INFORMATION will not be displayed if the file pool server is running with the NOCRR startup parameter in effect.
9. Counters are reset when file pool server processing is started (by using the FILESERV START command).

### Responses

The QUERY FILEPOOL REPORT command displays information in the following groups:

- FILE POOL OVERVIEW INFORMATION
- STORAGE GROUP MINIDISK INFORMATION
- STORAGE GROUP MINIDISK TOTALS
- AGENT INFORMATION
- LOG INFORMATION
- CATALOG SPACE INFORMATION (optional)
- SFS AND BYTE FILE COUNTER INFORMATION
- CRR COUNTER INFORMATION (only if CRR recovery server)

An example of the output is:

```
VM9SERV File Pool Report

Start-up Date mm/dd/yy          Query Date mm/dd/yy
Start-up Time hh:mm:ss         Query Time hh:mm:ss
=====
FILE POOL OVERVIEW INFORMATION

 6291456 Virtual Storage Size in Bytes (      6144 in KB)
  4057 Virtual Storage Highest Value in KB
    0 Virtual Storage Requests Denied
    0 Virtual Storage Reclaim Value

 25 Maximum Number of Connections
 11 Connections Highest Value

 16 Total Number of Agents
  2 Active Agents Highest Value

 10 Maximum Number of Storage Groups
  2 Storage Groups in Use

 10 Maximum Number of Minidisks
  5 Minidisks in Use

1769472 Potential Addressable 4K Blocks in File Pool
 2358 Defined Addressable 4K Blocks in File Pool
1767114 Undefined Addressable 4K Blocks in File Pool
=====
STORAGE GROUP MINIDISK INFORMATION

Storage   Minidisk   4K Blocks   4K Blocks   Virtual
Group No. Number    In-Use      Free        Address
```

```

1          1          124 - 42%          168          0303
1          2          101 - 35%          191          0304
1          3           71 - 24%          221          0305
2          4          149 - 51%          143          0306
2          5          115 - 10%         1075          0193

```

=====
STORAGE GROUP MINIDISK TOTALS

Storage Group No.	4K Blocks In-Use	4K Blocks Free
1	296 - 34%	580
2	264 - 18%	1218

=====
AGENT INFORMATION

```

16 Total Number of Agents
2  Active Agents Highest Value
2  Current Number of Agents

```

Userid	Type	Status	Agent Number	Wait	Uncommitted Blks
CHECKPT	Chkpt	Wait	2	I/O	0
ANASTOSM	User	Read	4	None	0

=====
LOG INFORMATION

```

First Log Virtual Address      0301
Second Log Virtual Address    0302

```

```

290 Number of Log Minidisk 4K Blocks
75% Percent(%) of Log Space Used
95% LUW Rollback/Suspend Threshold (% Log Space)
80% Backup Threshold (% Log Space)
mm/dd/yy Date of Last Control Backup
hh:mm:ss Time of Last Control Backup

```

=====
CONTROL MINIDISK INFORMATION

```

Control Minidisk Virtual Address      0300
1361 Number of Control Minidisk 512 Blocks

```

=====
CATALOG SPACE INFORMATION

```

503 Data Blocks
174 Occupied Data Blocks
35% Percent Occupied Data Blocks
505 Index Blocks
118 Occupied Index Blocks
23% Percent Used Index Blocks

```

=====
SFS AND BYTE FILE COUNTER INFORMATION

```

0 Add Minidisk Requests
0 Add Storage Requests
0 Byte File Cancel Requests
0 Byte File Change Access/Modification Time Requests
0 Byte File Change Audit Requests
0 Byte File Change Mode Requests
0 Byte File Change Owner Requests
0 Byte File Check File Accessibility Requests
0 Byte File Close Directory Requests
0 Byte File Close File Requests
0 Byte File Create Block Special File Requests
0 Byte File Create Character Special File Requests
0 Byte File Create Directory Requests
0 Byte File Create External Link Requests
0 Byte File Create Link Requests
0 Byte File Create Named Pipe (FIFO) Requests
0 Byte File Create Regular File Requests
0 Byte File Create Symbolic Link Requests
0 Byte File Lock Byte Requests
0 Byte File Lookup Requests
0 Byte File Makecat Requests
0 Byte File Open Directory Requests
0 Byte File Open File New With Intent Read Requests
0 Byte File Open File New With Intent Write Requests
0 Byte File Open File Read Requests
0 Byte File Open File Write Requests
0 Byte File Pipe Access Requests
0 Byte File Pipe Close Requests
0 Byte File Pipe Open For Read Requests
0 Byte File Pipe Open For Write Requests
0 Byte File Pipe Read Requests

```

## QUERY FILEPOOL REPORT

```
0 Byte File Pipe Stat Requests
0 Byte File Pipe Utime Requests
0 Byte File Pipe Write Requests
0 Byte File Read Directory Entry Requests
0 Byte File Read File Requests
0 Byte File Read Link Contents Requests
0 Byte File Remove Directory Requests
0 Byte File Rename Requests
0 Byte File Test Locked Bytes Requests
0 Byte File Token Return Requests
0 Byte File Unlink Requests
0 Byte File Unlinked File Cleanup Requests
0 Byte File Unlock Byte Requests
0 Byte File Write File Requests
0 Byte File ZAPCAT Requests
1 Cache Release Requests
0 Change DFSMS Related Attribute Requests
4 Change File Attributes Requests
0 Change Threshold Requests
11 Close Directory Requests
201 Close File Requests
0 Commit Requests
5 Connect Requests
0 Connect User Requests
3 Create Alias Requests
40 Create Directory Requests
0 Create External Object Requests
6 Create File Requests
1 Dataspace Requests
0 Delete Directory Requests
1 Delete File Requests
0 Delete Storage Requests
1 Dirattr Requests
13 File Copy Requests
0 File Pool Control Backup Requests
4 Get Directory Entry Requests
0 Get Directory Requests
7 Grant Administrator Authorization Requests
3 Grant Authorization Requests
23 Grant User Connect Requests
25 Lock Requests
0 Migrate Requests
11 Open Directory Requests
0 Open File CreateMig Requests
6 Open File New Requests
3 Open File Read Requests
7 Open File Replace Requests
191 Open File Write Requests
0 Precoordination Requests
0 Prepare Requests
7 Query Accessors Requests
51 Query Administrator Requests
0 Query Connected Users Requests
0 Query Dataspace Requests
4 Query Disable Requests
4 Query Enrolled Users Requests
17 Query File Pool Requests
```

```
2 Query Lock Conflicts Requests
10 Query User Space Requests
0 Query User Storage Group Requests
0 Read File Requests
0 Recall Requests
0 Recovery Close Catalog Requests
0 Recovery Get Catalog Requests
0 Recovery Open Catalog Requests
0 Recovery Put Catalog Requests
11 Refresh Directory Requests
0 Release Blocks Requests
2 Relocate Requests
0 Rename Requests
0 Rename Userid Requests
2 Revoke Administrator Authorization Requests
7 Revoke Authorization Requests
0 Revoke User Requests
0 Rollback Requests
0 Send DFSMS Data Requests
0 Set Reference Date Requests
1 SFS Send User Data Requests
0 Temporary Close Requests
25 Unlock Requests
```

```

    0 Write Accounting Requests
    16 Write File Requests
    726 Total File Pool Requests
    0 Total Byte File File Pool Requests
26317 File Pool Request Service Time (msec)
    726 Local File Pool Requests
    0 Remote File Pool Requests

    23 Alias Definitions Examined
    6 Alias Definitions Updated

5623 SAC Calls

    7 Checkpoints Taken
1873 Checkpoint Time (msec)

    0 Security Manager Exit Calls
    0 Security Manager Exit Time (msec)
    0 External Security Manager Exit Calls
    0 External Security Manager Exit Time (msec)

    0 DMSSFSEX Exit Calls
    0 DMSSFSEX Exit Time (msec)

    0 Recall DFSMS File Exit Calls
    0 Recall DFSMS File Exit Time (msec)
    0 Other DFSMS Exit Calls
    0 Other DFSMS Exit Time (msec)

    0 Storage Group Explicit Lock Conflicts
    0 File Space Explicit Lock Conflicts
    0 Directory Explicit Lock Conflicts
    0 File Explicit Lock Conflicts
    0 Storage Group Logical Lock Conflicts
    0 File Space Logical Lock Conflicts
    0 Directory Logical Lock Conflicts
    0 File Logical Lock Conflicts
    0 Catalog Lock Conflicts
    0 DIRCONTROL Resource Lock Conflicts
    0 File Recall Lock Conflicts
    0 Recall Exit Lock Conflicts
    0 Total Lock Conflicts
    0 Lock Wait Time (msec)
    0 Locks Denied Due to Timeout

    0 Deadlocks
    0 Rollbacks Due to Deadlock
11 LUW Rollbacks

    0 Byte File Directory Creation/Deletion Logical Lock
      Conflicts
    0 Byte File File Logical Lock Conflicts
    0 Byte File Global Storage Logical Lock Conflicts
    0 Byte File NAMECAT Unallocated Logical Lock Conflicts
    0 Byte File Token Manager Logical Lock Conflicts
    0 Total Byte File Lock Conflicts
    0 Byte File Byte Range Lock Waits

    0 Byte File Token Conflicts Causing Callbacks
    0 Byte File Callback Wait Time (msec)
    0 Byte File Token Callback Timeout Retries
    0 Byte File Token Callback Requestor Retries

    0 Byte File Logical Lock Retries
    0 Byte File Logical Lock Retries Exceeded

434 Begin LUWs
26626 Agent Holding Time (msec)

    24 QSAM Requests
    320 QSAM Time (msec)

    10 File Blocks Read
    266 File Blocks Written
    587 Catalog Blocks Read
    401 Catalog Blocks Written
    635 Control Minidisk Blocks Read
    74 Control Minidisk Blocks Written
    4 Log Blocks Read
    874 Log Blocks Written
2851 Total DASD Block Transfers

```

## QUERY FILEPOOL REPORT

```
10  BIO Requests to Read File Blocks
223 BIO Requests to Write File Blocks
587 BIO Requests to Read Catalog Blocks
359 BIO Requests to Write Catalog Blocks
28  BIO Requests to Read Control Minidisk Blocks
45  BIO Requests to Write Control Minidisk Blocks
4   BIO Requests to Read Log Blocks
702 BIO Requests to Write Log Blocks
1958 Total BIO Requests
25691 Total BIO Request Time (msec)

10  I/O Requests to Read File Blocks
224 I/O Requests to Write File Blocks
587 I/O Requests to Read Catalog Blocks
359 I/O Requests to Write Catalog Blocks
28  I/O Requests to Read Control Minidisk Blocks
45  I/O Requests to Write Control Minidisk Blocks
4   I/O Requests to Read Log Blocks
702 I/O Requests to Write Log Blocks
1959 Total I/O Requests
```

### ===== CRR COUNTER INFORMATION

```
0  Get Capability Requests
0  Get Logname Requests
0  Get LUWID Requests
0  Resync Init Requests
0  Resync Query Direction Requests
0  Resync Protocol Violations Requests
0  Write Log Requests
0  Total CRR Requests
0  CRR Request Service Time (msec)
0  Syncpoints
0  Syncpoint Time (msec)
0  Participating Resources
0  Log Checkpoints Taken
0  Log I/O Requests
0  BIO Request Time (msec)
=====
```

### Output descriptions:

Following are the fields and their meanings:

#### Start-up Date

This is the date of the last startup (FILESERV START) of the file pool.

#### Start-up Time

This is the time of the last startup (FILESERV START) of the file pool.

#### Query Date

This is the date of this query.

#### Query Time

This is the time of this query.

**FILE POOL OVERVIEW INFORMATION:** The FILE POOL OVERVIEW INFORMATION fields have the following meanings:

#### Virtual Storage Size in Bytes (      in KB)

This is the defined amount of virtual storage the file pool server has. Virtual storage amounts are supplied in both bytes and kilobytes. For example, virtual storage might be defined as 32 megabytes, and the display would show:

```
33554432 Virtual Storage Size in Bytes (      32768 in KB)
```

#### Virtual Storage Highest Value in KB

This is the greatest amount of virtual storage in kilobytes used at any one time. This value includes all virtual storage acquired by the file pool server since it was last started. If this value approaches the virtual machine size, you should increase the virtual storage size of the file pool server machine.

#### Virtual Storage Requests Denied

This is the number of virtual storage requests denied.

**Virtual Storage Reclaim Value**

This is the total number of times the file pool server has reclaimed its unused free storage. Periodically, when the server is running low on storage, it performs a reclaim process to free up additional storage.

**Maximum Number of Connections**

This is the maximum number of APPC/VM (and IUCV) connections to the file pool server machine including connections required for file pool server minidisks. This value is set by the MAXCONN operand of the OPTION directory control statement. For more information, see [“Step 4: Define a Server Machine”](#) on page 256.

**Connections Highest Value**

This is the highest number of APPC/VM (and IUCV) connections concurrently in use by the server machine. MAXCONN should be set higher than the value in this counter (perhaps by 50%). If this counter is equal to the MAXCONN setting, the server machine may have been rejecting connection requests to the file pool server.

**Note:** The MAXCONN value is an operand of the OPTION system directory control statement. It defines the maximum number of Inter-User Communication Vehicle (IUCV) connections (including APPC/VM connections) for the server machine. For more information, see [“Step 4: Define a Server Machine”](#) on page 256.

**Total Number of Agents**

This is the total number of user agents.

The primary use of agents is to function as the server's internal representation of a user. When a user requests something of the server, the server assigns that user to an agent and does the work. After completing the work, the server frees the agent and can reuse it for another user.

User agents are also used implicitly by file pool servers for certain functions. When a file pool server acts as a CRR recovery server, for example, user agents are used during the processing of various CRR requests from users and other server machines.

When the file pool server is initialized, it computes the number of user agents it should create by using the following formula based on the USERS value:

```
number_of_agents = 4 + truncate(USERS/8)
```

**Active Agents Highest Value**

This is the highest number of *user agents* that were concurrently in use during the current multiple user mode session.

For more information about Active Agents Highest Value, see [“Limit Monitoring”](#) on page 58.

**Maximum Number of Storage Groups**

This field shows the maximum number of storage groups that may be defined in the file pool. To exceed this maximum, you must regenerate the file pool.

**Storage Groups in Use**

This is the number of storage groups in use for this file pool.

**Maximum Number of Minidisks**

This field shows the maximum number of minidisks that can be assigned to all the file pool *storage groups*. The minidisk that holds the POOLDEF file, the log minidisks, and the control minidisk do not count towards this maximum. In the previous example, 10 is shown. Before adding the eleventh minidisk, you would have to regenerate the file pool with a larger MAXDISKS value.

**Minidisks in Use**

This is the number of minidisks in use for this file pool.

**Potential Addressable 4K Blocks in the File Pool**

This value tells you the number of 4KB blocks the file pool can manage in all its storage groups. The potential addressable 4KB blocks value is proportionally related to the size of the control minidisk. The word "potential" is used because the file pool may be able to handle (address) more 4KB blocks than are currently allocated to it. For instance, the file pool may be able to handle 30000 4KB

blocks, but currently only has 15000 4KB blocks of physical minidisk space allocated to it. As you add minidisks to the file pool, the number of physical blocks grows. If you try to add more physical blocks than the file pool can handle, the add minidisk (FILESERV MINIDISK command or FILEPOOL MINIDISK command) operation will be unsuccessful. To add the space, you need to regenerate the file pool. A complete discussion of how to do this is in [Chapter 11, “Regenerating a Repository File Pool,”](#) on page 217.

Note that Defined and Undefined Addressable 4K Blocks in File Pool add up to the Potential Addressable 4K Blocks in the File Pool.

### **Defined Addressable 4K Blocks in File Pool**

This is the number of 4KB blocks that are currently allocated to the file pool in all storage groups. These are the in-use and free blocks as described in STORAGE GROUP MINIDISK INFORMATION (displayed by QUERY FILEPOOL REPORT, QUERY FILEPOOL STORGRP, and QUERY FILEPOOL MINIDISK commands).

### **Undefined Addressable 4K Blocks in File Pool**

This is the maximum number of 4KB blocks that can still be added to the storage groups in the SFS file pool (using the FILESERV MINIDISK or FILEPOOL MINIDISK command). You can increase this maximum only by regenerating the SFS file pool server (FILESERV REGENERATE command).

**STORAGE GROUP MINIDISK INFORMATION:** The next group is STORAGE GROUP MINIDISK INFORMATION. For each minidisk allocated to a file pool storage group, you will see:

#### **Storage Group Number**

This is the storage group to which the minidisk belongs. Remember storage group 1 contains the catalog data. All other storage groups contain user data.

#### **Minidisk Number**

This is the number of the minidisk in the file pool. It is obtained from the numeric portion of the MDKnnnnn minidisk identifier in the POOLDEF file. This number, in effect, indicates the order in which minidisks were added to the file pool.

#### **4K Blocks In-Use**

This is the number of minidisk 4KB blocks currently in use (that is, have data in them). The number of in-use blocks is also shown as a percentage of the total number of minidisk blocks.

#### **4K Blocks Free**

This is the number of minidisk 4KB blocks not being used. A low value across the storage group indicates it may soon be necessary to add minidisks to this storage group (or move users out of it, for user storage groups).

#### **Virtual Address**

This is the virtual device number of the minidisk.

**STORAGE GROUP MINIDISK TOTALS:** The next group is STORAGE GROUP MINIDISK TOTALS. For each file pool storage group, you'll see:

#### **Storage Group Number**

This is the storage group. Remember storage group 1 contains the catalog data. All other storage groups contain user data.

#### **4K Blocks In-Use**

This is the number of storage group minidisk 4KB blocks currently in use (that is, have data in them). The number of in-use blocks is also shown as a percentage of the total number of minidisk blocks.

#### **4K Blocks Free**

This is the number of storage group minidisk 4KB blocks not being used. A low value for the storage group indicates it may soon be necessary to add minidisks to this storage group (or move users out of it, for user storage groups).

**AGENT INFORMATION:** The AGENT INFORMATION shows information about the users or internal processes for which the file pool server is currently doing work.



An *agent* is the server's internal representation of a user (or system task). There are usually far fewer agents than logged-on users. The file pool server calculates how many agents it needs based on what is specified in the USERS startup parameter.

Under AGENT INFORMATION, you will see:

**Total Number of Agents**

This is the total number of user agents.

The primary use of agents is to function as the server's internal representation of a user. When a user requests something of the file pool server, the server assigns that user to an agent and does the work. After completing the work, the server frees the agent and can reuse it for another user.

User agents are also used implicitly by file pool servers for certain functions. When the SFS file pool server acts as a CRR recovery server, user agents are used during the processing of various CRR requests from users and other server machines.

When the file pool server is initialized, it computes the number of user agents it should create by using the following formula based on the USERS startup parameter value:

```
number_of_agents = 4 + truncate(USERS/8)
```

**Active Agents Highest Value**

This is the highest number of *user agents* that were concurrently in use during the current multiple user mode session.

For more information about Active Agents Highest Value, see [“Limit Monitoring” on page 58](#).

**Current Number of Agents**

This is the number of user agents for which the file pool server is currently doing work. It is not the total number of user agents the server has created internally.

For the *Type*, *Status*, and *Wait* fields, one indication from the following list under the respective headings is displayed for each agent.

Userid	Type	Status	Agent Number	Wait	Uncommitted Blks
uuuuuuuu	Chkpt	Read	xxxxx	None	xxxxxxx
	User	Write		Communication	
	Syncpt-Logr	Wait		Checkpoint	
	Resync-Task	Inact		Lock	
	Pc→Resync	Running		Catalog_Buffer	
	Rm→Resync	Prepared		Control_Buffer	
	Resync→Rm			I/O	
	BFS_User			Connection	
				Timer	
				Subtask	
				Multi-Event	
				Suspended	
				AVSwait	
				ESM_Wait	
				DFSMS/VM_Wait	
				xx	
				Storage_Wait	
				Token_Callback	

For each agent, you will see:

**Userid**

This is the user ID of the virtual machine for which the file pool server is doing work. If the SFS file pool server is doing CRR tasks, the user ID could be @RESYNC.

**Type**

Can be:

### **Chkpt**

Means *file pool checkpoint*. A *checkpoint* is an internal file pool file pool server operation during which the changes recorded on the log minidisks are permanently made to the file pool. The file pool server takes checkpoints after a certain number of log minidisk blocks have been written.

### **User**

Means the file pool server is doing work for the user indicated in the `Userid` column.

### **Syncpt-Logr**

Means a CRR recovery server task dedicated to logging synchronization points. Various states of sync point processing are logged through this agent.

### **Resync-Task**

Means resynchronization task. The SFS file pool server creates a special type of agent to accomplish resynchronization and other tasks not directly related to the user. These tasks may be initiated by either the CRR recovery server or SFS file pool server.

### **Pc → Resync**

Means conversation from one CRR recovery server to another CRR recovery server in support of a protected conversation to perform an initial exchange of log names or resynchronization. This type applies to a user ID of `@RESYNC`.

### **Rm → Resync**

This is a CRR recovery server agent. A participating resource manager, such as an SFS file pool server, is communicating with the CRR recovery server to perform an initial exchange of log names in support of resynchronization initialization processing. This type applies to a user ID of `@RESYNC`.

### **Resync → Rm**

This is an SFS file pool agent. The CRR recovery server is communicating with the SFS file pool server to perform an exchange of log names in support of resynchronization recovery processing. This type applies to a user ID of `@RESYNC`.

### **BFS\_User**

Means the file pool server is doing Byte File System (BFS) work for the user indicated in the `Userid` column.

## **Status**

Can be:

### **Read**

Means the file pool server has not yet written anything to the log minidisks for the user. This applies to the following types of agent:

- User
- Syncpt-Logr
- Resync-Task
- Pc → Resync
- Rm → Resync
- Resync → Rm
- BFS\_User

### **Write**

Means the file pool server has written log information for the user. This applies to the following types of agent:

- User
- Syncpt-Logr
- Resync-Task
- Pc → Resync

- Rm → Resync
- Resync → Rm
- BFS\_User

**Wait**

Means the file pool checkpoint process is waiting for the file pool server to complete some work for other users. This applies only to the "checkpoint" agent.

**Inact**

Means the file pool checkpoint process is dormant. (The file pool server is not doing a checkpoint.) This applies only to the "checkpoint" agent.

**Running**

Means the file pool server is running the checkpoint process. Other user tasks remain dormant until it completes. This applies only to the "checkpoint" agent.

**Prepared**

Means the SFS file pool server is waiting for the second phase of the two-phase commit. This applies only to a "user" type of agent.

**Agent Number**

This is an internal number by which the file pool server knows the agent.

**Wait**

Can be:

**None**

Means the agent is not waiting for anything.

**Communication**

Means that for repository file pool servers, the agent is waiting for communication between the user machine for which it is processing and the file pool server machine to complete. This typically means the agent has responded to the last request from the user machine and is now still in a logical unit of work and waiting for the next request. The agent could also be waiting for communication with an external security manager (if the ESECURITY startup parameter is in effect) or a CRR recovery server.

For a CRR recovery server, means the agent is waiting for APPC communication from:

- User machine for CRR logging
- Participating resource manager for resynchronization initialization
- Another CRR recovery server for resynchronization recovery.

**Checkpoint**

Means the agent is waiting for an internal file pool server activity, known as a checkpoint, to complete.

This wait state applies to repository file pool servers.

**Lock**

Means the agent is waiting for a file pool server resource to become available. When a file pool server resource, such as a file or catalog information, is not available because it is being used by someone else, the resource is said to be *locked*.

**Catalog\_Buffer**

Means the agent is waiting for a catalog I/O buffer to become free. (All catalog buffers are currently in use.) The number of catalog buffers available is set by the CATBUFFERS startup parameter.

This wait state applies to repository file pool servers.

### **Control\_Buffer**

Means the agent is waiting for a control minidisk I/O buffer to become free. (All control buffers are currently in use.) The number of control buffers available is set by the CTLBUFFERS startup parameter.

This wait state applies to repository file pool servers.

### **I/O**

Means the agent is waiting for a file pool or CRR log I/O operation to complete on its behalf.

### **Connection**

Means the first phase of the two-phase commit has completed, and communication with the SFS file pool server has been severed. The prepared user is waiting for resynchronization, which is initiated by the CRR recovery server.

This wait state applies to SFS file pool servers.

### **Timer**

Is displayed when the agent is waiting for a time period to elapse (as specified by the RESYNCINTERVAL startup parameter or defaults to 10 minutes) before the periodic retry of resynchronization in an attempt to complete the update of a protected resource.

This wait state applies only to CRR recovery servers.

### **Subtask**

Is displayed when the agent sends a request to the file pool server, which must be suspended pending the completion of a file pool function (for example, the initial exchange of log names with the user's CRR recovery server). The file pool function is handled as a subtask, which is not directly related to the user's request that resulted in the subtask being initiated. While the subtask is executing, the original user's request is suspended in this wait state.

This wait state applies only to repository file pool servers.

### **Multi-Event**

Is displayed when the CRR recovery server communicates with multiple participating resource managers (such as the SFS file pool server) and waits for a response from a participating resource manager.

This wait state applies only to CRR recovery servers.

### **Suspended**

Is displayed when the agent is temporarily put into a waiting state by the CRR SUSPEND operator command. This wait state prevents the periodic retry of resynchronization.

This wait state applies only to CRR recovery servers.

### **AVSwait**

Is displayed when the agent is waiting for a response from AVS. The AVS virtual machine is the agent's communication partner.

This wait state applies only to CRR recovery servers.

### **ESM\_Wait**

Is displayed when the agent is waiting for an External Security Manager (ESM) response. This wait state applies only to repository file pool servers.

### **DFSMS/VM\_Wait**

Is displayed when the agent is waiting for a DFSMS/VM response. This wait state applies only to SFS repository file pool servers.

### **xx**

Is an internal number which means the file pool server is at a higher release level than the CMS in your machine, and the wait type is unknown to CMS.

### **Storage\_Wait**

Is displayed when the agent is waiting for storage to be available on the server.

**Token\_Callback**

is displayed when the file pool server is waiting for the Byte File System (BFS) client machine to respond to a request to invalidate cached file pool information.

**Uncommitted Blks**

Is the number of file pool file blocks used by this agent that have not yet been committed or rolled back. An inordinately high number in this column could indicate an errant program is writing data in an endless loop (this is commonly referred to as a *run-away write*). For more information, see [“Activating the Storage Use Exits \(SFS and BFS\)” on page 209](#). You can use the QUERY LIMITS command to determine the storage group to which the user is assigned. Note this field has meaning (nonzero value) only when the type column value is *user*.

**LOG INFORMATION:** The LOG INFORMATION shows information about the repository file pool log minidisks. The fields have the following meanings:

**First Log Virtual Address**

Is the virtual address of the first log (LOG1) minidisk.

**Second Log Virtual Address**

Is the virtual address of the second log (LOG2) minidisk.

**Number of Log Minidisk 4K Blocks**

Is the total number of 4KB blocks on each log minidisk. Both log minidisks are the same size.

**Percent(%) of Log Space Used**

Is the percent of the total log space currently used.

When BACKUP startup parameter is in effect, log space is freed when the control data is backed up. When NOBACKUP startup parameter is in effect, log space is freed during *checkpoint* processing. A checkpoint is an internal process during which all committed changes are permanently written to the file pool. Checkpoints occur after numerous log blocks are written. A checkpoint also occurs when multiple user mode operation is stopped normally (using any variation of the STOP operator command except STOP IMMEDIATE).

Note that control data backups and checkpoints done in multiple user mode may not free all the log space. Because the log is written sequentially, the only space that can be freed is the space used prior to the earliest uncommitted logical unit of work at the time of the checkpoint or control data backup.

**LUW Rollback/Suspend Threshold (% Log Space)**

If the BACKUP startup parameter is in effect, this is the percentage at which the file pool server will begin suspending the processing of SFS logical units of work and Byte File System (BFS) file pool server requests. Because the file pool server starts an automatic control data backup when the log is 80% full, an automatic backup is always in progress when the LUW Rollback/Suspend percentage is reached. The file pool server suspends work at this percentage to give the backup a chance to complete. If it did not begin suspending work at this point, the logs could fill before the control data backup has a chance to complete (which would free log space). In other words, the file pool server knows that log space will soon be free. Instead of risking a log-full (termination) condition, the file pool server suspends processing when the LUW Rollback/Suspend Threshold is reached.

Note that it is still possible for the logs to fill during a control data backup. Because the file pool server is multi-tasking, it must schedule the suspension of activity. (Certain in-progress activities cannot be interrupted.) It is possible the logs could fill before the file pool server has a chance to stop all the in-progress work. While it is unlikely this will happen, it is most likely to occur if there is a great deal of concurrent activity and the logs are small. In this case, you would enter a FILESERV BACKUP command to back up the control data and free the log space. Then you can restart multiple user mode operation.

If NOBACKUP is in effect, the LUW Rollback/Suspend Threshold is the percentage at which the file pool server will begin rolling back long-running SFS logical units of work and BFS file pool server requests. When NOBACKUP is in effect, the SFS file pool server attempts to free log space whenever a checkpoint occurs. (Usually checkpoints occur after several log blocks have been written.) Log space cannot be freed if there is a long-running SFS logical unit of work or BFS file pool server request that is still in progress. If the log reaches the LUW Rollback/Suspend Threshold, it rolls back long-running SFS logical units of work and BFS file pool server requests so the next checkpoint will free some

space. The server's goal is to prevent the log from filling, which causes the file pool server to stop. There is a chance the logs could fill even though the file pool server is rolling back SFS logical units of work and BFS file pool server requests. In this case, the file pool server stops. To recover, you would restart multiple user mode processing. Log recovery processing should free enough log space for continued operation.

### **Backup Threshold (% Log Space)**

Indicates the percentage at which a control data backup will automatically be started. This value is displayed only when the BACKUP startup parameter is in effect. If you do not see this value in your display, the NOBACKUP startup parameter is in effect.

### **Date of Last Control Backup**

This is the date of the last control backup. A backup of the control data contains the contents of the POOLDEF file, control minidisk, and the file pool catalogs at the time the backup was made.

### **Time of Last Control Backup**

This is the local time of the last control backup.

**CONTROL MINIDISK INFORMATION:** The field in the CONTROL MINIDISK INFORMATION has the following meaning:

### **Control Minidisk Virtual Address**

This is the control minidisk's virtual address.

### **Number of Control Minidisk 512 Blocks**

This is the total number of 512-byte blocks available on the control minidisk.

**CATALOG SPACE INFORMATION:** The fields in the CATALOG SPACE INFORMATION have the following meanings:

### **Data Blocks**

Is the total number of addressable 4KB blocks that can be used to hold information about base files, directories, aliases, enrolled users, and so on. Note that this number represents potential addressable blocks, not the number of DASD blocks allocated to the storage group. The amount of physical DASD space allocated to storage group 1 can be determined by the output displayed by the QUERY FILEPOOL STORGRP command.

### **Occupied Data Blocks**

Is the number of data blocks that actually have data in them. An occupied block is not necessarily full. It may have room for additional data.

### **Percent Occupied Data Blocks**

Is calculated from the preceding two values.

### **Index Blocks**

Is the total number of addressable 4KB blocks that can be used to hold index values and pointers to data blocks. Note that this number represents potential addressable blocks, not the number of DASD blocks allocated to the storage group. The amount of physical DASD space allocated to storage group 1 can be determined by the output displayed by the QUERY FILEPOOL STORGRP command.

### **Occupied Index Blocks**

Is the number of index blocks that actually have data in them. An occupied block is not necessarily full. It may have room for additional data.

### **Percent Used Index Blocks**

Is calculated from the preceding two values.

The number of potential addressable data and index blocks is determined during file pool generation (FILESERV GENERATE) or regeneration (FILESERV REGENERATE). During these processes, the server calculates the total number of addressable catalog blocks by using the MAXUSERS value in a space estimation formula. It internally allocates a few of these blocks for header information and splits the remaining blocks evenly between data blocks and index blocks.

When you add a minidisk to the catalog storage group, the file pool server does not immediately allocate these blocks for use as index blocks or data blocks. Instead, it allocates the blocks as needed. If your file

pool usage is such that 80% of all catalog storage group blocks are needed for data, that is what the SFS file pool server attempts to get from the physical DASD allocated to the storage group.

**SFS AND BYTE FILE COUNTER INFORMATION:** When a file pool server is started for multiple user mode processing, it automatically counts the occurrences of certain events. The event is counted regardless of whether it was successful or not. These counters, which appear in the SFS AND BYTE FILE COUNTER INFORMATION section of the command output, can be used in monitoring SFS file pool server performance, monitoring limits, and for performance problem determination. (See [Chapter 5, "Operation,"](#) on page 55, and [z/VM: Performance](#) for information on these topics.)

Note that the events that the SFS file pool server counts can often be caused in more than one way.

Some examples follow:

- The file pool server counts the number of Lock Requests. For example, lock requests can be caused by the CREATE LOCK command, the DISABLE operator command, or the DMSCRLOC Callable Services Library routine. The file pool server does not distinguish between any of the three. It counts only the number of times it was requested to create a lock, regardless of how the request was made.
- Some CMS commands allow the use of special pattern-matching characters that result in a number of file pool server operations. The file pool server counts the number of times it attempted to perform the operation, not the number of times a particular command was entered. Note that there are also other CMS commands that do this.
- Some of the counters report total amounts of elapsed time. Do not confuse this with processor time. To compute the time that has elapsed for an event, the server makes a note of the clock time at the start of the event and when the event completes. It then subtracts the starting time from the ending time and adds this to the total for the file pool server session. Note the time is local time.

The counters in the SFS AND BYTE FILE COUNTER INFORMATION are reset every time file pool server processing is started.

They have the following meanings:

#### **Add Minidisk Requests**

The number of times the server was requested to add minidisks while the server is in multiple user mode. This request is caused by the FILEPOOL MINIDISK command.

#### **Add Storage Requests**

The number of times file pool administrators have requested an increase in the number of file blocks allocated to an SFS file space or a Byte File System (BFS). The MODIFY USER command increases storage.

#### **Byte File Cancel Requests**

This is the number of times the file pool was requesting to cancel a specified Byte File request.

#### **Byte File Change Access/Modification Time Requests**

This is the number of times the file pool was requested to change the access or modification times of a Byte File Object.

#### **Byte File Change Audit Requests**

This is the number of times the file pool was requested to change the audit flags for a Byte File Object.

#### **Byte File Change Mode Requests**

This is the number of times the file pool was requested to change the mode associated with a Byte File Object.

#### **Byte File Change Owner Requests**

This is the number of times the file pool was requested to change the owner (UID/GID) of a Byte File Object.

#### **Byte File Check File Accessibility Requests**

This is the number of times the file pool was requested to check the accessibility of a Byte File Object.

#### **Byte File Close Directory Requests**

This is the number of times the file pool was requested to close a directory.

### **Byte File Close File Requests**

This is the number of times the file pool was requested to close a regular file.

### **Byte File Create Block Special File Requests**

This is the number of times the file pool was requested to create a block special file.

### **Byte File Create Character Special File Requests**

This is the number of times the file pool was requested to create a character special file.

### **Byte File Create Directory Requests**

This is the number of times the file pool was requested to create a Byte File directory.

### **Byte File Create External Link Requests**

This is the number of times the file pool was requested to create an external link.

### **Byte File Create Link Requests**

This is the number of times the file pool was requested to create a hard link.

### **Byte File Create Named Pipe (FIFO) Requests**

This is the number of times the file pool was requested to create a named pipe (FIFO).

### **Byte File Create Regular file Requests**

This is the number of times the file pool was requested to create a regular Byte File.

### **Byte File Create Symbolic Link Requests**

This is the number of times the file pool was requested to create a symbolic link.

### **Byte File Lock Byte Requests**

This is the number of times the file pool was requested to lock a byte range.

### **Byte File Lookup Requests**

This is the number of times the file pool was requested to lookup a Byte File Object.

### **Byte File Makecat Requests**

This is the number of times the file pool was requested to make a Byte File Object.

### **Byte File Open Directory Requests**

This is the number of times the file pool was requested to open a directory (to be subsequently used for reading).

### **Byte File Open File New With Intent Read Requests**

This is the number of times the file pool was requested to create a new file with intent READ.

### **Byte File Open File New With Intent Write Requests**

This is the number of times the file pool was requested to create a new file with intent WRITE.

### **Byte File Open File Read Requests**

This is the number of times the file pool was requested to open a file for read access.

### **Byte File Open File Write Requests**

This is the number of times the file pool was requested to open a file for write or trunc access.

### **Byte File Pipe Access Requests**

This is the number of times the file pool was requested to verify the access authorization to a named pipe.

### **Byte File Pipe Close Requests**

This is the number of times the FIFO file pool was requested to close a named pipe.

### **Byte File Pipe Open For Read Requests**

This is the number of times the FIFO file pool was requested to open a named pipe for read.

### **Byte File Pipe Open For Write Requests**

This is the number of times the FIFO file pool was requested to open a named pipe for write.

### **Byte File Pipe Read Requests**

This is the number of times the FIFO file pool was requested to read from a named pipe.

### **Byte File Pipe Stat Requests**

This is the number of times the FIFO file pool was requested to obtain the current status information about a named pipe.



**Byte File Pipe Utime Requests**

This is the number of times the file pool was requested to update the timestamps associated with a named pipe.

**Byte File Pipe Write Requests**

This is the number of times the FIFO file pool was requested to write to a named pipe.

**Byte File Read Directory Entry Requests**

This is the number of times the file pool was requested to read directory entries.

**Byte File Read File Requests**

This is the number of times the file pool was requested to read data from files.

**Byte File Read Link Contents Requests**

This is the number of times the file pool was requested to read the contents of a link.

**Byte File Remove Directory Requests**

This is the number of times the file pool was requested to remove a directory.

**Byte File Rename Requests**

This is the number of times the file pool was requested to rename a Byte File Object.

**Byte File Test Locked Bytes Requests**

This is the number of times the file pool was requested to test byte range locks held on a specific byte range.

**Byte File Token Return Requests**

This is the number of times a client requested the file pool to free a vnode token or block token.

**Byte File Unlink Requests**

This is the number of times the file pool was requested to remove a Byte File Object.

**Byte File Unlinked File Cleanup Requests**

This is the number of unlinked files removed during FILESERV START.

**Byte File Unlock Byte Requests**

This is the number of times the file pool was requested to unlock a byte range.

**Byte File Write File Requests**

This is the number of times the file pool was requested to write to a regular file.

**Byte File ZAPCAT Requests**

This is the number of times the file pool was requested to read or alter catalogs using the ZAPCAT request.

**Cache Release Requests**

The number of times the file pool server has been requested to stop sending SFS cache updates to a user machine. The *cache* in a user machine contains information about the user's accessed directories. The file pool server sends updates to the user's machine as needed to keep the user's cache current. While the updating of a cache can be suspended for many reasons (the user logs off, the APPC/VM communication link to the user is severed), the file pool server only counts explicit requests from CMS that is running in the user's machine. CMS in the user's machine decides whether to issue cache release requests to the file pool server depending on various internal conditions in the user's machine. The issuing of the request is not directly related to CMS commands or applications the user might be running.

**Change DFSMS Related Attribute Requests**

The number of times the file pool server was requested to change the DFSMS/VM Related Attributes (DRA) for a specified file or directory.

**Change File Attributes Requests**

The total number of times the file pool server was requested to modify the overwrite and/or recoverability attributes of SFS files.

**Change Threshold Requests**

The number of times the file pool server was requested to change an SFS user's space allocation threshold percentage (the threshold percentage at which the file pool server produces a space allocation warning message). This counter is incremented when the SET THRESHOLD command is processed.

### **Close Directory Requests**

The number of times the file pool server was requested to close a directory.

### **Close File Requests**

The number of times the file pool server was requested to close a file.

### **Commit Requests**

The number of times the file pool server was requested to commit work.

### **Connect Requests**

The number of times the file pool server was requested to accept a connection from a user machine.

### **Connect User Requests**

The number of times the file pool server was requested to establish a connection to a file pool for the specified user ID, or set the associated user ID for an already established connection to the specified user ID or the number of times the file pool server was requested to connect user IDs that differ from the user ID of the connecting machine.

### **Create Alias Requests**

The number of times the file pool server was requested to create an alias in an SFS directory.

### **Create Directory Requests**

The number of times the file pool server was requested to create a new SFS directory.

### **Create External Object Requests**

The number of times the file pool server was requested to create an external object in an SFS directory. This request is caused by the DMSCROB CSL routine.

### **Create File Requests**

The number of times the file pool server was requested to add a new (empty) file to an SFS directory. This request is caused by the DMSCRFIL CSL routine or the CREATE FILE command.

### **Dataspace Requests**

The total number of times the file pool server was requested to assign or remove an SFS DIRCONTROL directory from the data space eligibility list. This request is caused by the DATASPACE command.

### **Delete Directory Requests**

The number of times the file pool server was requested to delete an SFS directory.

### **Delete File Requests**

The number of times the file pool server was requested to erase a base file or SFS alias residing in the file pool.

### **Delete Storage Requests**

The number of times file pool administrators have requested a decrease in the number of file blocks allocated to an SFS file space or a Byte File System (BFS). The MODIFY USER command decreases storage.

### **Dirattr Requests**

The total number of times the file pool server was requested to change SFS directory control attributes. This request is caused by the DIRATTR command and by the DMSDIRAT CSL routine.

### **File Copy Requests**

The number of times the file pool server was requested to copy a source SFS file in the file pool it is managing to a target SFS file in the same file pool.

### **File Pool Control Backup Requests**

The number of times the file pool server was requested to take a control data backup using the FILEPOOL CONTROL BACKUP command.

### **Get Directory Entry Requests**

The number of times the file pool server was requested to provide information about a single directory entry. This request is made by the DMSEXIST Exist, DMSEXIDI Exist Directory, and DMSEXIFI Exist Files CSL routines.

### **Get Directory Requests**

The number of times the file pool server was requested to read directory records. This is the request that is made by the DMSGETDI Get Directory, DMSGETDA Get Directory - Searchall, DMSGETDD Get Directory - Dir, DMSGETDK Get Directory - Lock, DMSGETDL Get Directory - Alias, DMSGETDS

Get Directory - Searchauth, DMSGETDT Get Directory - Auth, and DMSGETDX Get Directory - File Extended CSL routines.

**Grant Administrator Authorization Requests**

The number of times the file pool server has been requested to enroll a file pool administrator. This request is caused by ENROLL ADMINISTRATOR command or GRANT ADMIN operator command.

**Grant Authorization Requests**

The number of times the file pool server has been requested to grant authority on an SFS object to a user.

**Grant User Connect Requests**

The number of times the file pool server was requested to create an SFS file space or a Byte File System (BFS). This request is caused by the ENROLL USER command.

**Lock Requests**

The number of times the file pool server has been requested to lock objects in the file pool. (Note that DISABLE operator commands are considered lock requests.) This counter does not include the number of implicit locks the file pool server has created. This counter does not include Byte File System (BFS) byte range lock requests.

**Migrate Requests**

The number of times the file pool server was requested (using the DFSMS MIGRATE or DFSMS MANAGE command) to open a file or alias and prepare to migrate 4KB file blocks.

**Open Directory Requests**

The number of times the file pool server was requested to open a directory (to be subsequently used for reading).

**Open File CreateMig Requests**

The number of times the file pool server was requested to create a file in migrated status (DMSOPBLK with the CREATMIG intent).

**Open File New Requests**

The number of times the file pool server was requested to open a new SFS file (that is, create a new file).

**Open File Read Requests**

The number of times the file pool server was requested to open a file for read access.

**Open File Replace Requests**

The number of times the file pool server was requested to open an existing file such that all existing file data is replaced with file data that is to be added. For SFS files, if the file does not exist, it is created.

**Open File Write Requests**

The number of times the file pool server was requested to open an SFS file for write access. (In this case, records can be changed or added to the file without affecting other records in the file.)

**Precoordination Requests**

The number of times the file pool server was called during the precoordination phase of a syncpoint to see if a user's file space is still exceeded. (The Precoordination request is issued only if user's file space was exceeded during the current LUW.)

**Prepare Requests**

The number of times the file pool server was requested to do the first phase of a two-phase commit.

**Query Accessors Requests**

The total number of times the file pool server was requested to return information about users who are accessing SFS DIRCONTROL directories. This request is caused by the QUERY ACCESSORS command.

**Query Administrator Requests**

The number of times the file pool server was requested to provide information about users with file pool administrator authority.

### **Query Connected Users Requests**

The number of times the file pool server was requested to provide information about users connected to it.

### **Query Dataspace Requests**

The total number of times the file pool server was requested to return information about the eligibility of SFS directories for use in data spaces. This request is caused by the QUERY DATASPACE command.

### **Query Disable Requests**

The number of times the file pool server was requested to query a storage group or a file space or all file spaces and all storage groups in a file pool to determine if they have been previously disabled. This request is caused by the QUERY FILEPOOL DISABLE command, or DMSQFPDS CSL routine, or QUERY DISABLE operator command.

### **Query Enrolled Users Requests**

The number of times the file pool server was requested to provide information about SFS enrolled users and Byte File Systems in the file pool.

### **Query File Pool Requests**

The number of times the file pool server was requested to provide information about the status of the file pool. This request is caused either by the QUERY FILEPOOL STATUS, QUERY FILEPOOL REPORT, QUERY FILEPOOL STORGRP, QUERY FILEPOOL OVERVIEW, QUERY FILEPOOL MINIDISK, QUERY FILEPOOL AGENT, QUERY FILEPOOL LOG, QUERY FILEPOOL CATALOG, QUERY FILEPOOL COUNTER, or QUERY FILEPOOL CRR command.

### **Query Lock Conflicts Requests**

The number of times the file pool server was requested to provide information about lock conflicts. This request is caused by the QUERY FILEPOOL CONFLICT command.

### **Query User Space Requests**

The number of times the file pool server was requested to provide information about the space allocated to enrolled SFS users and Byte File Systems. This request is caused by the QUERY LIMITS command, DMSQLIMA Query Limits for All Enrolled Users CSL routine, or DMSQLIMU Query Limits for a Single User CSL routine.

### **Query User Storage Group Requests**

The number of times the file pool server was requested to query a file pool for information about the user storage groups. This request is caused by the DMSQUSG CSL routine.

### **Read File Requests**

The number of times the file pool server was requested to read data from files.

### **Recall Requests**

The number of times the file pool server was requested (using the DFSMS RECALL command, or automatic recalls when files in migrated status are referenced) to open a file or alias and prepare to recall 4KB file blocks from migrated status.

### **Recovery Close Catalog Requests**

The number of times the file pool server was requested to close a previously-opened catalog object. These requests are generally made by programs that recover user data.

### **Recovery Get Catalog Requests**

The number of times the file pool server was requested to get catalog information. These requests are generally made by programs that recover user data.

### **Recovery Open Catalog Requests**

The number of times the file pool server was requested to open a catalog object. These requests are generally made by programs that recover user data.

### **Recovery Put Catalog Requests**

The number of times the file pool server was requested to write to a catalog object. These requests are generally made by programs that recover user data.

### **Refresh Directory Requests**

The number of times the file pool server was requested to refresh the SFS information maintained in the user machine cache. CMS in the user machine decides when it is necessary to have the cache refreshed based on what the user is doing.

**Release Blocks Requests**

The total number of times the file pool server was requested to release (deallocate) blocks that encountered write I/O errors in a specified storage group.

**Relocate Requests**

The number of times the file pool server was requested to move a file or sub-tree from one SFS directory to another. This request is caused by the RELOCATE command or DMSRELOC CSL routine.

**Rename Requests**

The number of times the file pool server was requested to rename an SFS file or directory. This request is caused by the RENAME command or DMSRENAM CSL routine.

**Rename Userid Requests**

The number of times the file pool server was requested to rename an SFS file space or a Byte File System (BFS). This request is made by the FILEPOOL RENAME command.

**Revoke Administrator Authorization Requests**

The number of times the file pool server was requested to delete a file pool administrator.

**Revoke Authorization Requests**

The number of times the file pool server was requested to revoke authority on an SFS file or directory from a user.

**Revoke User Requests**

The number of times the file pool server was requested to delete an SFS enrolled user or a Byte File System (BFS) from the file pool.

**Rollback Requests**

The number of times the file pool server was requested to roll back a logical unit of work. (These are referred to as voluntary rollbacks.)

**Send DFSMS Data Requests**

The number of times the file pool server was requested to pass user-defined data to the DFSMS/VM exit.

**Set Reference Date Requests**

The number of times CMS requested the file pool server to change the date of last reference for files in a SFS directory mapped to a data space.

**Note:** When a directory is mapped to a data space, CMS in the user machine references the data space directly. Because the file pool server is not used for the file access, it does not know a reference occurred. To maintain a correct date of last reference for the file, the user machine tells the file pool server machine to update its catalogs (and the dataspace) with the new date of last reference.

**SFS Send User Data Requests**

The total number of times the file pool server was requested to pass user data information to an external security manager.

**Temporary Close Requests**

The total number of times the file pool server was requested to prepare an open file for commit processing.

**Unlock Requests**

The number of times the file pool server was requested to delete an explicit lock on a file pool object. This counter does not include Byte File System (BFS) byte range unlock requests. This request is caused by DELETE LOCK command and ENABLE command.

**Write Accounting Requests**

The number of times the file pool server was requested to write accounting records for a file pool.

**Write File Requests**

The number of times the file pool server was requested to write to a file in the file pool.

**Total File Pool Requests**

The total number of SFS and BFS requests made to the file pool server during the current multiple user mode session.

### **Total Byte File File Pool Requests**

This is the total number of requests identified by "Byte File" under the "SFS AND BYTE FILE COUNTER INFORMATION" section. This is a sub-total included in the 'Total File Pool Requests'.

### **File Pool Request Service Time (msec)**

The total amount of time (in milliseconds) the file pool server took to process requests. To calculate the service time, the file pool server makes a note of the time whenever it receives a request from a user machine. When it sends the result of the request back to the user machine it again notes the time. It then subtracts the time the request was received from the time the response was sent to determine the service time for that particular request. This counter is the sum of the service times of all the SFS and BFS requests the file pool server received after last being started.

### **Local File Pool Requests**

The total number of file pool requests made by users on the same processor.

### **Remote File Pool Requests**

The total number of file pool requests made by users on different processors.

### **Alias Definitions Examined**

The number of times the file pool server had to read catalog information about SFS aliases during the processing of all requests.

### **Alias Definitions Updated**

The total number of times an SFS alias definition was created, changed, or deleted. An alias definition is changed when any of its attributes are changed (for example, number of records in the file, date, time, record format, and so on).

### **SAC Calls**

The total number of times the file pool server called its Storage Access Component (SAC). SAC is the portion of file pool server code that accesses the catalogs and user files. It also provides support for locking, catalog indexes, logging, file pool recovery, and file pool generation.

### **Checkpoints Taken**

The number of server checkpoints taken during the current execution of the server. A *checkpoint* is an internal file pool server operation during which the changes recorded on the log minidisks are permanently made to the file pool. The file pool server takes checkpoints after a certain number of log minidisk blocks have been written.

### **Checkpoint Time (msec)**

The total amount of elapsed time (in milliseconds) spent in doing server checkpoints.

### **Security Manager Exit Calls**

The number of times the file pool server called an external security manager.

### **Security Manager Exit Time (msec)**

The total amount of elapsed time (in milliseconds) spent in processing security manager calls. (The file pool server notes the clock time, calls the external security manager, and notes the clock time again upon return from the external security manager.)

### **External Security Manager Exit Calls**

The number of times the file pool server called an external security manager and that manager needed to use IUCV to communicate with a security manager running in another virtual machine.

### **External Security Manager Exit Time (msec)**

The total amount of elapsed time (in milliseconds) spent in processing authorization requests by an external security manager that needed to use IUCV to communicate with a security manager running in another virtual machine.

### **DMSSFSEX Exit Calls**

The number of times the file pool server called a DMSSFSEX CSL routine. When usage of file space, a Byte File System (BFS) space, or a user storage group reaches a predefined point, the file pool server will call one of the two exits of this type: File Space Usage or User Storage Group Full. The exits are called when the file pool server calls the IBM-supplied CSL routine DMSSFSEX.

### **DMSSFSEX Exit Time (msec)**

This is the total amount of elapsed time (in milliseconds) spent in processing DMSSFSEX requests.

**Recall DFSMS File Exit Calls**

The number of times the file pool server called a DFSMS/VM Recall File exit point (to automatically recall a file in migrated status that has been referenced).

**Recall DFSMS File Exit Time (msec)**

This is the total amount of elapsed time (in milliseconds) spent in processing DFSMS/VM Recall File Exit calls.

**Other DFSMS Exit Calls**

The number of times the file pool server called one of the following predefined DFSMS/VM exits:

- Initialization Exit
- Acquire DRAs (DFSMS/VM Related Attributes)
- Erase File Notification
- Send DFSMS/VM Data Exit
- DFSMS/VM Termination Exit

**Other DFSMS Exit Time (msec)**

This is the total amount of elapsed time (in milliseconds) spent in processing one of the following predefined DFSMS/VM exits:

- Initialization Exit
- Acquire DRAs (DFSMS/VM Related Attributes)
- Erase File Notification
- Send DFSMS/VM Data Exit
- DFSMS/VM Termination Exit

**Storage Group Explicit Lock Conflicts**

The number of times a request for a lock on a storage group was denied because an explicit lock had already been created on that storage group. The request for the lock could have been either an implicit or an explicit request.

**File Space Explicit Lock Conflicts**

The number of times a request for a lock on an SFS file space or Byte File System (BFS) was denied because an explicit lock had already been created for that file space or BFS. The request for the lock could have been either an implicit or an explicit request.

**Directory Explicit Lock Conflicts**

The number of times a request for a lock on a directory was denied because an explicit lock had already been created on that directory. The request for the lock could have been either an implicit or an explicit request.

**File Explicit Lock Conflicts**

The number of times a request for a lock on a file was denied because an explicit lock had already been created on that file. The request for the lock could have been either an implicit or an explicit request. This counter excludes lock conflicts caused by byte range locking of byte files.

**Storage Group Logical Lock Conflicts**

The number of times a request for a lock on a storage group was denied (or waited, if FILEWAIT was set to ON for an explicit lock request) because an implicit lock had already been created on that storage group. The request for the lock could have been either an implicit or an explicit request.

**File Space Logical Lock Conflicts**

The number of times a request for a lock on an SFS file space or Byte File System (BFS) was denied (or waited, if FILEWAIT was set to ON for an explicit lock request) because an implicit lock had already been created for that file space or BFS. The request for the lock could have been either an implicit or an explicit request.

**Directory Logical Lock Conflicts**

The number of times a request for a lock on a directory was denied (or waited, if FILEWAIT was set to ON) because an implicit lock had already been created on that directory. The request for the lock could have been either an implicit or an explicit request.

### **File Logical Lock Conflicts**

The number of times a request for a lock on a file was denied (or waited, if FILEWAIT was set to ON) because an implicit lock had already been created on that file. The request for the lock could have been either an implicit or an explicit request. This counter excludes lock conflicts caused by byte range locking of byte files.

### **Catalog Lock Conflicts**

The number of times a request for a lock on a part of a catalog was delayed or denied because an implicit lock had already been created on the desired item. All lock requests on catalogs are implicit. File pool server uses internal locking mechanism on the catalogs to prevent simultaneous updating of the same catalog data. SFS server is designed so that catalog locks are held for as brief a time as possible. In general, a file pool server does not deny a user request because of a catalog lock conflict. Instead, it waits for the lock to free and gives the next user control. Catalog locks are transparent to CMS users.

### **DIRCONTROL Resource Lock Conflicts**

The number of times a request was delayed or denied because an implicit lock had already been created on that directory. SFS uses internal locking mechanisms to maintain the consistency of directory control directories.

### **File Recall Lock Conflicts**

The number of times a request for a lock involving a DFSMS/VM File Recall resource was delayed or denied because an implicit lock had already been created on the desired item.

### **Recall Exit Lock Conflicts**

The number of times a request for a lock involving a DFSMS/VM Recall Exit resource was delayed or denied because an implicit lock had already been created on that item.

### **Total Lock Conflicts**

Is the sum of the following counters:

- Storage Group Explicit Lock Conflicts
- File Space Explicit Lock Conflicts
- Directory Explicit Lock Conflicts
- File Explicit Lock Conflicts
- Storage Group Logical Lock Conflicts
- File Space Logical Lock Conflicts
- Directory Logical Lock Conflicts
- File Logical Lock Conflicts
- Catalog Lock Conflicts
- DIRCONTROL Resource Lock Conflicts
- File Recall Lock Conflicts
- Recall Exit Lock Conflicts

This counter excludes lock conflicts caused by byte range locking of byte files.

### **Lock Wait Time (msec)**

The total amount of time (in milliseconds) spent waiting for locks. For each case of a lock wait, the server calculates the "lock wait time" by subtracting the time at which it suspended processing the request from the time at which it resumed processing the request. This counter is the sum of the lock wait times for all lock wait conditions that occurred after the file pool server was started.

This counter excludes lock wait time caused by byte range locking of byte files.

### **Locks Denied Due to Timeout**

The number of times a lock could not be obtained within the timeout interval. When a lock is waited for, an internal timer may be set to prevent excessive wait times. If the timer expires, the lock is denied, and the request is unsuccessful. For example, it is possible for applications to commit changes to two file pools together from DMSCOMM. If two applications were to commit changes to the same two file pools at the same time, it would be possible to create a deadlock with application A holding the locks needed by B on one of the file pools, and B holding locks needed by A on the other. In such a situation, one of the lock requests would eventually be timed out.



**Deadlocks**

The number of SFS or BFS deadlocks the file pool server detected and corrected. A *deadlock* occurs when two applications are each holding file pool resources the other needs. For example, application A opens FILEA for writing. Application B opens FILEB for writing. Next application A tries to open FILEB for writing and waits (because it had set FILEWAIT to ON). Application B simultaneously tries to open FILEA for writing. Application B waits because it also had FILEWAIT set to ON. Each application is waiting for a file the other has implicitly locked. Neither can proceed. The file pool server detects this condition and rolls back the logical unit of work that started most recently. (The application gets an error return code.)

**Note:** The file pool server is able to resolve some deadlocks internally through a retry process. Consequently, not all deadlocks result in rollback. The number of rollbacks (because of deadlock) are counted by Rollbacks Due to Deadlock. The number of rollbacks (because of deadlock or other reasons) is counted by LUW Rollbacks.

**Rollbacks Due to Deadlock**

The total number of SFS logical units of work (LUWs) and Byte File System (BFS) file pool server rolled back because of a deadlock that could not be corrected through the retry process.

**LUW Rollbacks**

The total number of SFS logical units of work (LUWs) and Byte File System (BFS) file pool server rolled back.

Rollbacks can be classified as either voluntary or involuntary. Voluntary rollbacks are requested by the application and are counted by Rollback Requests. You can compute the number of involuntary rollbacks by subtracting Rollback Requests from LUW Rollbacks. You can additionally compute how many involuntary rollbacks were caused by reasons other than deadlock by subtracting Rollbacks Because of Deadlock from LUW Rollbacks.

**Begin LUWs**

The total number of SFS and BFS logical units of work (LUWs) the file pool server has started.

**Byte File Directory Creation/Deletion Logical Lock Conflicts**

This is the number of times a request for a lock on an object (file, directory, link or symbolic link) to be created or deleted was denied (or waited for) because an implicit lock was already held on the object in a Byte File System (BFS).

**Byte File File Logical Lock Conflicts**

This is the number of times a request for a lock, unlock, or close on a file for serializing byte range lock/unlock and file closes was denied (or waited for) because an implicit lock had already been created on that file. The request for the lock was from an implicit request.

**Byte File Global Storage Logical Lock Conflicts**

This is the number of times a request for a lock on the object was denied (or waited for) because an implicit lock was already held on the object.

**Byte File NAMECAT Unallocated Logical Lock Conflicts**

This is the number of times a request for a lock on an unallocated NAMECAT row was denied because the implicit lock was already held on the row.

**Byte File Token Manager Logical Lock Conflicts**

This is the number of times the Token Manager requested a WRITE VNODE lock but had to wait for the lock because the implicit lock was already held.

**Total Byte File Lock Conflicts**

This is the sum of the following counters: Byte File Directory Creation/Deletion Logical Lock Conflicts, Byte File File Logical Lock Conflicts, Byte File Global Storage Logical Lock Conflicts, Byte File NAMECAT Unallocated Logical Lock Conflicts, Byte File Token Manager Logical Lock Conflicts.

**Byte File Byte Range Lock Waits**

This is the number of times a Byte Range Lock Request had to wait before being awarded the requested lock.

**Byte File Token Conflicts Causing Callbacks**

This is a count of the number of callbacks of tokens due to token conflicts. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file

request requires a token held by another user, the requestor must wait until the client machine that holds the token returns it (responds to a callback of that token).

### **Byte File Callback Wait Time (msecs)**

This is the time (in milliseconds) spent waiting for callbacks of tokens. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token held by another user, the requestor must wait until the client machine that holds the token returns it (responds to a callback of that token).

### **Byte File Token Callback Timeout Retries**

This is a count of the number of retries of callbacks because of a short delay of the holding client machine to respond to the token callback request. This is a retry attempt by the BFS server. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token held by another user, the requestor must wait until the client machine that holds the token returns the it (responds to a callback of that token).

### **Byte File Token Callback Requestor Retries**

This is a count of the number of requestor retries because of extended delays in call back response. This occurs when it is necessary to give up waiting for a normal callback completion because of exceeding the retry limit for callback retries (see "Token Callback Timeout Retries" counter). Return code to requestor suggests retry by the client application. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token held by another user, the requestor must wait until the client machine that holds the token returns it (responds to a callback of that token).

### **Byte File Logical Lock Retries**

This is the number of times a retry was attempted to obtain the BFS requests logical locks.

### **Byte File Logical Lock Retries Exceeded**

This is the number of times the request was denied logical locks due to the lock retry count being exceeded.

### **Agent Holding Time (msec)**

The total amount of time (in milliseconds) the server spent holding user agents. The server calculates agent holding time by subtracting the time at which the agent is acquired from the time it is released (made available for other users). This counter is the sum of the agent holding times for all agents the server has used after it was last started. Note that this counter applies to SFS file pool servers and CRR recovery servers. For more information about agents, see "Active Agents Highest Value" in the AGENT INFORMATION section of the QUERY FILEPOOL REPORT command.

### **QSAM Requests**

The number of QSAM requests the SFS file pool server has made. QSAM requests occur for control data backups and restores. They also occur for security audit trace output.

### **QSAM Time (msec)**

The time (in milliseconds) the SFS file pool server waited for all QSAM requests to complete. The SFS file pool server notes the time immediately before making a QSAM request and immediately upon return from QSAM. The difference between the two times is the "QSAM Time" for a QSAM request.

### **File Blocks Read**

The total number of 4KB blocks read from user storage groups.

### **File Blocks Written**

is the total number of 4KB blocks written to user storage groups.

### **Catalog Blocks Read**

The total number of 4KB blocks read from the file pool catalogs (storage group 1).

### **Catalog Blocks Written**

The total number of 4KB blocks written to the file pool catalogs (storage group 1).

### **Control Minidisk Blocks Read**

The total number of 512-byte blocks read from the file pool control minidisk.

### **Control Minidisk Blocks Written**

The total number of 512-byte blocks written to the file pool control minidisk.

**Log Blocks Read**

The total number of 4KB blocks read from the file pool log minidisks.

**Log Blocks Written**

The total number of 4KB blocks written to the file pool log minidisks.

**Total DASD Block Transfers**

The total number of file pool blocks read or written. (It is the sum of the preceding eight counters.)

**BIO Requests to Read File Blocks**

The total number of times block I/O was used to read blocks from user storage groups. Where possible, a file pool server tries to read multiple blocks using a single block I/O request. Therefore, this number is usually less than the File Blocks Read value.

**BIO Requests to Write File Blocks**

The total number of times block I/O was used to write blocks to user storage groups. Where possible, a file pool server tries to write multiple blocks using a single block I/O request. Therefore, this number is usually less than the File Blocks Written value.

**BIO Requests to Read Catalog Blocks**

The total number of times block I/O was used to read blocks from the file pool catalogs (storage group 1). The file pool server reads catalog blocks one at a time, as needed.

**BIO Requests to Write Catalog Blocks**

The total number of times block I/O was used to write blocks to the file pool catalogs (storage group 1). The file pool server writes catalog blocks one at a time, as needed.

**BIO Requests to Read Control Minidisk Blocks**

The total number of times block I/O was used to read blocks from the file pool control minidisk. In general, the file pool server reads control minidisk blocks one at a time, as needed.

**BIO Requests to Write Control Minidisk Blocks**

The total number of times block I/O was used to write blocks to the file pool control minidisk. Where possible, a file pool server tries to write multiple blocks using a single block I/O request. Therefore, this number is usually less than the Control Minidisk Blocks Written value.

**BIO Requests to Read Log Blocks**

The total number of times block I/O was used to read blocks from the file pool log minidisks. The file pool server reads log minidisk blocks one at a time, as needed.

**BIO Requests to Write Log Blocks**

The total number of times block I/O was used to write blocks to the file pool log minidisks. There are cases where the file pool server can group multiple log blocks into one block I/O request. Therefore, this number may be less than the Log Blocks Written value.

**Total BIO Requests**

The total number of times the file pool server used block I/O. (This counter is the sum of the preceding eight counters.)

**Total BIO Request Time (msec)**

The total amount of time (in milliseconds) that agents had to wait because they initiated a block I/O request in the server. When a user agent requests a block I/O operation, the file pool server checks the processor clock, starts the operation, and (rather than waiting for the I/O to complete) begins doing work for other users. After handling requests for other active agents, and the block I/O request has completed, the file pool server dispatches the agent that was waiting. The file pool server then checks the clock again and subtracts the start time from this time to determine how long the user agent waited. It is this value that is added to the counter.

**I/O Requests to Read File Blocks**

The number of I/O requests issued by CP to read blocks from user storage groups. This count will typically be larger than BIO Requests to Read File Blocks because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD except when minidisk caching is active and all requested blocks are found in the minidisk cache.

### **I/O Requests to Write File Blocks**

The number of I/O requests issued by CP to write blocks to user storage groups. This count will typically be larger than BIO Requests to Write File Blocks because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD.

### **I/O Requests to Read Catalog Blocks**

The number of I/O requests issued by CP to read blocks from the file pool catalogs (storage group 1). Each I/O request will normally result in a physical I/O to DASD except when minidisk caching is active and the requested block is found in the minidisk cache.

### **I/O Requests to Write Catalog Blocks**

The number of I/O requests issued by CP to write blocks to the file pool catalogs (storage group 1). Each I/O request will normally result in a physical I/O to DASD.

### **I/O Requests to Read Control Minidisk Blocks**

The number of I/O requests issued by CP to read blocks from the control minidisk. Each I/O request will normally result in a physical I/O to DASD.

### **I/O Requests to Write Control Minidisk Blocks**

The number of I/O requests issued by CP to write blocks to the control minidisk. This count will typically be larger than BIO Requests to Write Control Minidisk Blocks because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD.

### **I/O Requests to Read Log Blocks**

The number of I/O requests issued by CP to read blocks from the file pool log minidisks. Each I/O request will normally result in a physical I/O to DASD except when minidisk caching is active and the requested block is found in the minidisk cache. (Note: Minidisk caching should not be used for the log minidisks.)

### **I/O Requests to Write Log Blocks**

The number of I/O requests issued by CP to write blocks to the file pool log minidisks. This count can be larger than BIO Requests to Write Log Blocks because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD.

### **Total I/O Requests**

Is the approximate number of times the file pool server caused CP to process a channel program to read or write DASD blocks.

**CRR COUNTER INFORMATION:** The CRR COUNTER INFORMATION is displayed only when the file pool server is a CRR recovery server.

Some of the counters report total amounts of elapsed time. Do not confuse this with processor time. To compute the time that has elapsed for an event, the server makes a note of the clock time at the start of the event and when the event completes. It then subtracts the starting time from the ending time and adds this to the total for the file pool server session. Note the time is local time.

The CRR COUNTER INFORMATION fields have the following meanings:

#### **Get Capability Requests**

Is the total number of times the CRR recovery server had requests to get the capabilities of a protected conversation partner in a coordinated commit.

#### **Get Logname Requests**

Is the total number of times the CRR recovery server had requests to get the CRR recovery server's log name.

#### **Get LUWID Requests**

Is the total number of times the CRR recovery server had requests to create a new SNA LU 6.2 logical unit of work ID (LUWID).

#### **Resync Init Requests**

Is the total number of times the CRR recovery server had requests to do resynchronization activity for a resource, following a resource error during a coordinated commit.

**Resync Query Direction Requests**

Is the total number of times the CRR recovery server had requests to determine the sync point direction (commit or rollback) by means of resynchronization activity, following an error during communications with the sync point initiator.

**Resync Protocol Violations Requests**

Is the total number of times the CRR recovery server was given information a communications protocol violation was detected during a synchronization (sync) point.

**Write Log Requests**

Is the total number of times the CRR recovery server had requests to write to the CRR log.

**Total CRR Requests**

Is the total number of CRR recovery server requests (that is, the sum of the preceding CRR recovery server counts).

**CRR Request Service Time (msec)**

Is the total amount of time (in milliseconds) the CRR recovery server spent handling CRR requests. For each CRR request, the CRR recovery server calculates the holding time by subtracting the time at which the request begins from the time it ends. This counter is the sum of all CRR requests the CRR recover server processed since it was last started.

**Syncpoints**

Is the total number of times the CRR recovery server was requested to process a CRR sync point.

**Syncpoint Time (msec)**

Is the total amount of time (in milliseconds) the CRR recovery server spent handling sync points. For each sync point request, the CRR recovery server calculates the holding time by subtracting the time at which the request begins from the time it ends. This counter is the sum of all sync point requests the CRR recovery server processed after it was last started.

**Participating Resources**

Is the total number of resources that have participated in coordination or recovery.

**Note:** Participating Resources÷Syncpoints is the average number of different resource managers that have participated in each sync point.

**Log Checkpoints Taken**

Is the total number of times the CRR recovery server had done a CRR log checkpoint.

**Log I/O Requests**

Is the number of I/O requests issued by CP to read or write CRR recovery server log minidisk blocks. Each I/O request will normally result in a physical I/O to DASD.

**BIO Request Time (msec)**

Is the total amount of time (in milliseconds) agents had to wait because they initiated block I/O requests to the CRR logs. When a user agent initiates a BIO request, the CRR recovery server checks the processor clock, starts the operation, and (rather than waiting for the BIO request to complete), begins doing work for other users. After handling requests for other active agents, and the block I/O request has completed, the CRR recovery server dispatches the agent that was waiting. The CRR recovery server then checks the clock again and subtracts the start time from this time to determine how long the agent waited. It is this value that is added to the counter.

**Messages and Return Codes**

In addition to the messages listed below, this command displays other system messages. These system messages are listed in *z/VM: CMS Commands and Utilities Reference*.

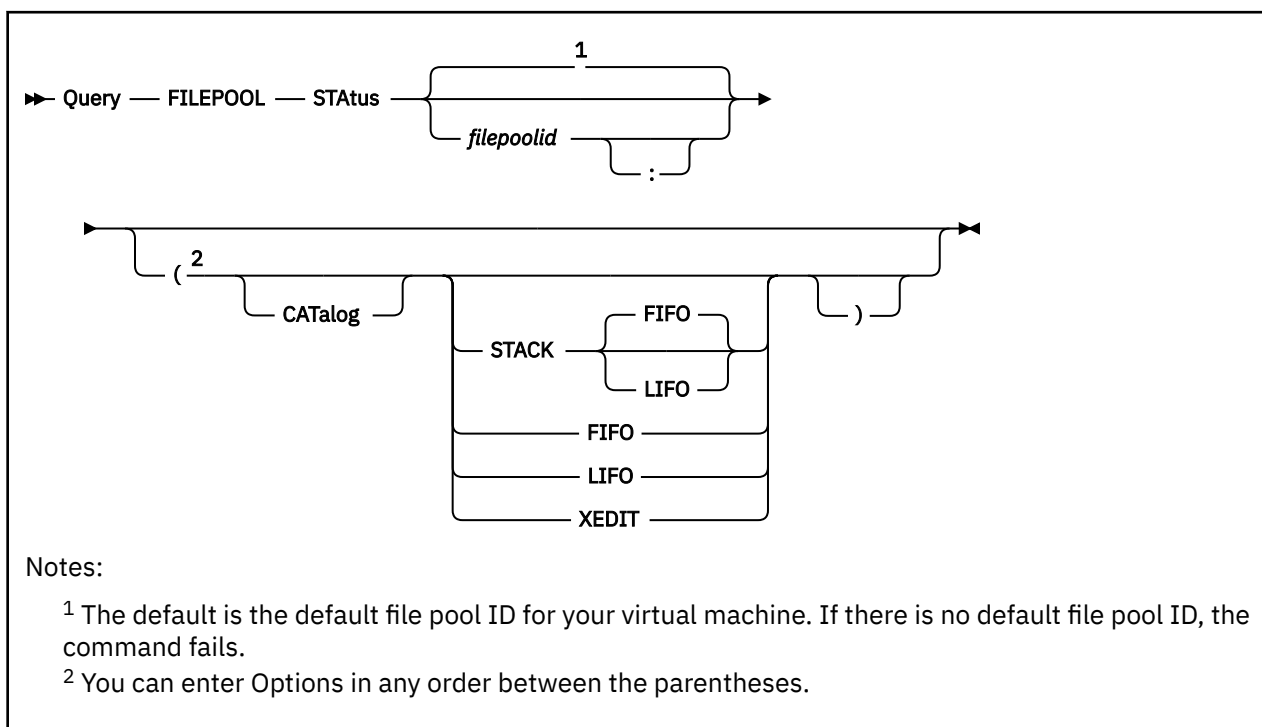
Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect filepoolid: *filepoolid* [RC=24]

## QUERY FILEPOOL REPORT

- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS688E XEDIT option only valid from XEDIT environment [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24]
- DMS1223E There is no default file pool currently defined [RC=40]

## QUERY FILEPOOL STATUS



### Authorization

File Pool Administrator or Connected User

### Purpose

Use the QUERY FILEPOOL STATUS command to display information about a file pool. The display also includes information about a file pool server and CRR recovery server processing against the file pool.

The contents of this query are similar to that of the QUERY FILEPOOL REPORT command. QUERY FILEPOOL REPORT provides some additional information and a more usable format than QUERY FILEPOOL STATUS.

The information returned can assist you with the task of performance analysis and timing for file pool servers and CRR recovery servers.

### Operands

*filepoolid*  
*filepoolid:*

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

### Options

#### CATalog

indicates you want information about the file pool catalogs to be displayed. File pool administration authority is required **only** when the CATALOG option is specified.

#### STACK FIFO STACK LIFO

places the information in the console stack rather than displaying it at the terminal. FIFO is the default.

## QUERY FILEPOOL STATUS

### FIFO

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

### LIFO

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

### XEDIT

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

## Usage Notes

1. If you do not need all the information generated by this command, but would like to see specific information only, refer to the following table to determine what QUERY FILEPOOL command will give you the information you need.

This Cross Reference table matches QUERY FILEPOOL commands with the type of information they produce. The contents of these queries are similar to that of the QUERY FILEPOOL STATUS and QUERY FILEPOOL REPORT commands.

File pool administration authority is required ONLY for acquiring CATALOG SPACE INFORMATION.

	QUERY FILEPOOL REPORT	QUERY FILEPOOL OVERVIEW	QUERY FILEPOOL STORGRP	QUERY FILEPOOL MINIDISK	QUERY FILEPOOL AGENT	QUERY FILEPOOL LOG	QUERY FILEPOOL CATALOG	QUERY FILEPOOL COUNTER	QUERY FILEPOOL CRR
Filepool name, Start-up & Query Date/Time	X	X	X	X	X	X	X	X	X
FILE POOL OVERVIEW INFORMATION	X	X		X					
STORAGE GROUP MINIDISK INFORMATION	X			X					
STORAGE GROUP MINIDISK TOTALS	X		X	X					
AGENT INFORMATION	X				X				
LOG INFORMATION	X			X		X			
CONTROL MINIDISK INFORMATION	X			X					
CATALOG SPACE INFORMATION	X						X		
SFS AND BYTE FILE COUNTER INFORMATION	X							X	
CRR COUNTER INFORMATION	X							X	X

2. Because of the large amount of information returned, the XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set to ON. CMS FULLSCREEN also lets you scroll through the information.
3. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL STATUS command with the XEDIT option on the command line. When a new file is edited, there is no chance of accidentally overlaying information already in the file.
4. For the log information, the *Backup Threshold* is not displayed if the SFS file pool server is running with the NOBACKUP startup parameter in effect.



5. CATALOG SPACE INFORMATION will be displayed only when requested by using the CATALOG option. Use this option with discretion as considerable processing may be required to obtain the requested information.

The CATALOG option is generally not useful for CRR recovery servers.

6. If the QUERY FILEPOOL STATUS command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
7. If the current release of z/VM is processing file pool information from a prior release, you will see only the information applicable for the file pool server for that prior release.
8. Counters are reset when file pool server or CRR recovery server processing is started (by using the FILESERV START command).

## Responses

The QUERY FILEPOOL STATUS command displays information in the following groups:

- File pool counter information
- CRR counter information
- File pool compare information
- File pool information
- Currently defined minidisk information
- Agent information
- Log information
- Catalog space information (optional)

An example of the output is:

```

Counter Information
    4 Active Agents Highest Value
5279 Virtual Storage Highest Value
    0 Virtual Storage Requests Denied
    2 Checkpoints Taken
1833 Checkpoint Time (msec)
    0 Security Manager Exit Calls
    0 Security Manager Exit Time (msec)
    0 External Security Manager Exit Calls
    0 External Security Manager Exit Time (msec)
    0 Add Storage Requests
    0 Cache Release Requests
    0 Change Threshold Requests
    9 Close Directory Requests
434 Close File Requests
100 Commit Requests
    0 Connect Requests
    1 Create Alias Requests
    0 Create Directory Requests
    0 Delete Directory Requests
    83 Delete File Requests
    0 Delete Storage Requests
    1 File Copy Requests
    0 Get Directory Requests
    4 Get Directory Entry Requests
    0 Grant Administrator Authorization Requests
    12 Grant Authorization Requests
    0 Grant User Connect Requests
    18 Lock Requests
    16 Open Directory Requests
    2 Open File New Requests
430 Open File Read Requests
    6 Open File Replace Requests
    1 Open File Write Requests
    0 Query Administrator Requests
    0 Query Connected Users Requests
    0 Query Enrolled Users Requests
    0 Query Lock Conflicts Requests
    1 Query File Pool Requests
  
```

## QUERY FILEPOOL STATUS

```
17 Query User Space Requests
408 Read File Requests
0 Recovery Close Catalog Requests
0 Recovery Get Catalog Requests
0 Recovery Open Catalog Requests
0 Recovery Put Catalog Requests
3100 Refresh Directory Requests
0 Relocate Requests
1 Rename Requests
0 Revoke Administrator Authorization Requests
0 Revoke Authorization Requests
0 Revoke User Requests
5 Rollback Requests
17 Unlock Requests
0 Write Accounting Requests
15 Write File Requests
4681 Total File Pool Requests
335323 File Pool Request Service Time (msec)
2264 Local File Pool Requests
2417 Remote File Pool Requests
1 Alias Definitions Examined
1 Alias Definitions Updated
771 Begin LUWs
3069788 Agent Holding Time (msec)
5 LUW Rollbacks
6770 SAC Calls
0 Storage Group Explicit Lock Conflicts
0 File Space Explicit Lock Conflicts
0 Directory Explicit Lock Conflicts
0 File Explicit Lock Conflicts
0 Storage Group Logical Lock Conflicts
0 File Space Logical Lock Conflicts
0 Directory Logical Lock Conflicts
0 File Logical Lock Conflicts
0 Catalog Lock Conflicts
5 Total Lock Conflicts
0 Lock Wait Time (msec)
0 Deadlocks
58 QSAM Requests
3385 QSAM Time (msec)
1571 File Blocks Read
113 File Blocks Written
213 Catalog Blocks Read
37 Catalog Blocks Written
57 Control Minidisk Blocks Read
110 Control Minidisk Blocks Written
15 Log Blocks Read
314 Log Blocks Written
2430 Total DASD Block Transfers
848 BIO Requests to Read File Blocks
34 BIO Requests to Write File Blocks
213 BIO Requests to Read Catalog Blocks
37 BIO Requests to Write Catalog Blocks
35 BIO Requests to Read Control Minidisk Blocks
27 BIO Requests to Write Control Minidisk Blocks
15 BIO Requests to Read Log Blocks
314 BIO Requests to Write Log Blocks
1523 Total BIO Requests
56018 Total BIO Request Time (msec)
856 I/O Requests to Read File Blocks
36 I/O Requests to Write File Blocks
213 I/O Requests to Read Catalog Blocks
37 I/O Requests to Write Catalog Blocks
35 I/O Requests to Read Control Minidisk Blocks
30 I/O Requests to Write Control Minidisk Blocks
15 I/O Requests to Read Log Blocks
314 I/O Requests to Write Log Blocks
1536 Total I/O Requests

10 Release Blocks Requests
3 Temporary Close Requests
20 SFS Send User Data Requests
0 Change File Attributes Requests
11 Dataspace Requests
45 Dirattr Requests
0 Prepare Requests
30 Query Accessors Requests
32 Query Dataspace Requests
10 Set Reference Date Requests
0 Rollbacks Due to Deadlock
5 DIRCONTROL Resource Lock Conflicts
0 Precoordination Requests
```

```

0 Connect User Requests
0 Create External Object Requests
0 Create File Requests
0 Query User Storage Group Requests
0 DMSSFSEX Exit Calls
0 DMSSFSEX Exit Time (msec)
0 File Recall Lock Conflicts
0 Recall Exit Lock Conflicts
0 Change DFSMS Related Attribute Requests
0 Migrate Requests
0 Recall Requests
0 Send DFSMS Data Requests
0 Recall DFSMS File Exit Calls
0 Recall DFSMS File Exit Time (msec)
0 Other DFSMS Exit Calls
0 Other DFSMS Exit Time (msec)
0 Rename Userid Requests
0 File Pool Control Backup Requests
0 Add Minidisk Requests
0 Query Disable Requests
0 Locks Denied Due to Timeout
0 Virtual Storage Reclaim Value
0 Open File CreateMig Requests

0 Byte File Cancel Requests
0 Byte File Change Access/Modification Time Requests
0 Byte File Change Audit Requests
0 Byte File Change Mode Requests
0 Byte File Change Owner Requests
0 Byte File Check File Accessibility Requests
0 Byte File Close Directory Requests
0 Byte File Close File Requests
0 Byte File Create Block Special File Requests
0 Byte File Create Character Special File Requests
0 Byte File Create Directory Requests
0 Byte File Create External Link Requests
0 Byte File Create Link Requests
0 Byte File Create Named Pipe (FIFO) Requests
0 Byte File Create Regular File Requests
0 Byte File Create Symbolic Link Requests
0 Byte File Lock Byte Requests
0 Byte File Lookup Requests
0 Byte File Makecat Requests
0 Byte File Open Directory Requests
0 Byte File Open File New With Intent Read Requests
0 Byte File Open File New With Intent Write Requests
0 Byte File Open File Read Requests
0 Byte File Open File Write Requests
0 Byte File Pipe Access Requests
0 Byte File Pipe Close Requests
0 Byte File Pipe Open For Read Requests
0 Byte File Pipe Open For Write Requests
0 Byte File Pipe Read Requests
0 Byte File Pipe Stat Requests
0 Byte File Pipe Utime Requests
0 Byte File Pipe Write Requests
0 Byte File Read Directory Entry Requests
0 Byte File Read File Requests

0 Byte File Read Link Contents Requests
0 Byte File Remove Directory Requests
0 Byte File Rename Requests
0 Byte File Test Locked Bytes Requests
0 Byte File Token Return Requests
0 Byte File Unlink Requests
0 Byte File Unlinked File Cleanup Requests
0 Byte File Unlock Byte Requests
0 Byte File Write File Requests
0 Byte File ZAPCAT Requests
0 Total Byte File File Pool Requests

0 Byte File Directory Creation/Deletion Logical Lock
Conflicts
0 Byte File File Logical Lock Conflicts
0 Byte File Global Storage Logical Lock Conflicts
0 Byte File NAMECAT Unallocated Logical Lock Conflicts
0 Byte File Token Manager Logical Lock Conflicts
0 Total Byte File Lock Conflicts
0 Byte File Byte Range Lock Waits

0 Byte File Token Conflicts Causing Callbacks

```

## QUERY FILEPOOL STATUS

```
0 Byte File Callback Wait Time (msec)
0 Byte File Token Callback Timeout Retries
0 Byte File Token Callback Requestor Retries

0 Byte File Logical Lock Retries
0 Byte File Logical Lock Retries Exceeded

CRR Counter Information      0 Get Capability Requests
0 Get Logname Requests
0 Get LUWID Requests
0 Resync Init Requests
0 Resync Query Direction Requests
0 Resync Protocol Violations Requests
0 Write Log Requests
0 Total CRR Requests
0 CRR Request Service Time (msec)
0 Syncpoints
0 Syncpoint Time (msec)
0 Participating Resources
0 Log Checkpoints taken
0 Log I/O Requests
0 BIO Request Time (msec)

File Pool Compare Information

3 Active Agents Highest Value
9 Connections Highest Value
3394 Virtual Storage Highest Value

File Pool Information

500 Maximum Number of Storage Groups
500 Maximum Number of Minidisks
25100288 Potential Addressable 4K Blocks in File Pool

Currently Defined Minidisk Information

Minidisk Number  Group Number  4K Blocks In-Use  4K Blocks Free
1                 1             1030 - 19%      4356
2                 2             201 - 4%       5185

Agent Information

Number of Agents      2

Userid  Type  Status  Agent Number  Wait  Uncommitted Blks
CHECKPT Chkpt  Read    2           None      0
USER01  User   Write   3           Communication

Log Information

1188 Number of Log Minidisk 4K Blocks
14% Percent(%) of Log Space Used
95% LUW Rollback/Suspend Threshold (% Log Space)
80% Backup Threshold (% Log Space)

Catalog Space Information

42519 Data Blocks
249 Occupied Data Blocks
1% Percent Occupied Data Blocks
42521 Index Blocks
216 Occupied Index Blocks
1% Percent Used Index Blocks
```

**Note:** For *Agent Information* in the previous example, the Type, Status, and Wait field for Agent Information is one of the indicators from the list for each user ID displayed.

When a server is started for multiple user mode processing, it automatically counts the occurrences of certain events. The event is counted regardless of its success. These counters, which appear in the first section of the command output, can be used in monitoring server performance, monitoring limits, and for performance problem determination. (For more information on these topics, see [Chapter 5, "Operation,"](#) on page 55, and [z/VM: Performance.](#))

Note the events the file pool server counts can often be caused in more than one way. Following are some examples.

- The file pool server counts the number of "Lock Requests". Lock requests can be caused by the CREATE LOCK command, the DISABLE operator command, or the DMSCRLOC Callable Services Library routine. The file pool server does not distinguish between any of the three. It counts only the number of times it was requested to create a lock, regardless of how the request was made.
- Some file pool commands allow the use of special pattern-matching characters that result in a number of file pool server operations. The file pool server counts the number of times it attempted to perform the operation, not the number of times a particular command was entered. Note that there are also other CMS commands that do this.
- Some of the counters report total amounts of elapsed time. Do not confuse this with processor time. To compute the time that has elapsed for an event, the file pool server makes a note of the clock time at the start of the event and when the event completes. It then subtracts the starting time from the ending time and adds this to the total for the file pool server session.

The counters are reset every time server processing is started. They have the following meanings:

#### **Active Agents Highest Value**

The highest number of *user agents* that were concurrently in use during the current multiple user mode session. The primary use of agents is to function as the server's internal representation of a user. When a user requests something of the server, the server assigns that user to an agent and does the work. After completing the work, the server frees the agent and can reuse it for another user.

User agents are also used implicitly for certain functions. When the server acts as a CRR recovery server, user agents are used during the processing of various CRR requests from users and other server machines.

When the server is initialized, it computes the number of user agents it should create by using the following formula based on the USERS value:

```
number_of_agents = 4 + truncate(USERS/8)
```

For more information about Active Agents Highest Value, see [“Limit Monitoring” on page 58](#).

#### **Virtual Storage Highest Value**

The greatest amount of virtual storage (in KB) used at any one time. This value includes all virtual storage acquired by the file poolserver after it was last started. If this value approaches the virtual machine size, you should increase the virtual storage size of the file pool server machine.

#### **Virtual Storage Requests Denied**

The number of times the file pool server tried to get virtual storage but could not. In most cases, the server cannot get virtual storage because there is none available in the virtual machine (it is all in use). Any incidence of a denied virtual storage request indicates the virtual machine might not be configured with enough virtual storage.

#### **Checkpoints Taken**

The number of file pool checkpoints taken during the current execution of the file pool server. A *checkpoint* is an internal server operation during which the changes recorded on the log minidisks are permanently made to the file pool. The file pool server takes checkpoints after a certain number of log minidisk blocks have been written.

#### **Checkpoint Time (msec)**

The total amount of elapsed time (in milliseconds) spent in doing checkpoints.

#### **Security Manager Exit Calls**

The number of times the file pool server called an external security manager.

#### **Security Manager Exit Time (msec)**

The total amount of elapsed time (in milliseconds) spent in processing security manager calls. (The file pool server notes the clock time, calls the external security manager, and notes the clock time again upon return from the external security manager.)

#### **External Security Manager Exit Calls**

The number of times the file pool server called an external security manager and that manager needed to use IUCV to communicate with a security manager running in another virtual machine.

### **External Security Manager Exit Time (msec)**

The total amount of elapsed time (in milliseconds) spent in processing authorization requests by an external security manager that needed to use IUCV to communicate with a security manager running in another virtual machine.

### **Add Storage Requests**

The number of times file pool administrators have requested an increase in the number of file blocks allocated to an SFS file space or a Byte File System (BFS). The MODIFY USER command increases storage.

### **Cache Release Requests**

The number of times the file pool server has been requested to stop sending SFS cache updates to a user machine. The *cache* in a user machine contains information about the user's accessed directories. The file pool server sends updates to the user's machine as needed to keep the user's cache current. While the updating of a cache can be suspended for many reasons (the user logs off, the APPC/VM communication link to the user is severed), the file pool server only counts explicit requests from CMS that is running in the user's machine. CMS in the user's machine decides whether to issue cache release requests to the file pool server depending on various internal conditions in the user's machine. The issuing of the request is not directly related to CMS commands or applications the user might be running.

### **Change Threshold Requests**

The number of times the file pool server was requested to change a user's space allocation threshold percentage (the threshold percentage at which the file pool server produces a space allocation warning message). This counter is incremented when the SET THRESHOLD command is processed.

### **Close Directory Requests**

The number of times the file pool server was requested to close a directory.

### **Close File Requests**

The number of times the file pool server was requested to close a file.

### **Commit Requests**

The number of times the file pool server was requested to commit work.

### **Connect Requests**

The number of times the file pool server was requested to accept a connection from a user machine.

### **Create Alias Requests**

The number of times the file pool server was requested to create an alias in an SFS directory.

### **Create Directory Requests**

The number of times the file pool server was requested to create a new SFS directory.

### **Delete Directory Requests**

The number of times the file pool server was requested to delete an SFS directory.

### **Delete File Requests**

The number of times the file pool server was requested to erase a base file or alias residing in the file pool.

### **Delete Storage Requests**

The number of times file pool administrators have requested a decrease in the number of file block allocated to an SFS file space or a Byte File System (BFS). The MODIFY USER command decreases storage.

### **File Copy Requests**

The number of times the file pool server was requested to copy a source SFS file in the file pool it is managing to a target SFS file in the same file pool.

### **Get Directory Requests**

The number of times the file pool server was requested to read directory records. The request made by the DMSGETDI Get Directory, DMSGETDA Get Directory - Searchall, DMSGETDD Get Directory - Dir, DMSGETDK Get Directory - Lock, DMSGETDL Get Directory - Alias, DMSGETDS Get Directory - Searchauth, DMSGETDT Get Directory - Auth, and DMSGETDX Get Directory - File Extended CSL routines.

**Get Directory Entry Requests**

The number of times the file pool server was requested to provide information about a single directory entry. This request is made by the DMSEXIST Exist, DMSEXIDI Exist Directory, and DMSEXIFI Exist Files CSL routines.

**Grant Administrator Authorization Requests**

The number of times the file pool server has been requested to enroll a file pool administrator. This request is caused by ENROLL ADMINISTRATOR command.

**Grant Authorization Requests**

The number of times the file pool server has been requested to grant authority on an SFS object to a user.

**Grant User Connect Requests**

The number of times the file pool server was requested to create an SFS file space or a Byte File System (BFS). This request is caused by the ENROLL USER command.

**Lock Requests**

The number of times the file pool server has been requested to lock objects in the file pool. (Note that DISABLE operator commands are considered lock requests.) This counter does not include the number of implicit locks the file pool server has created. This counter does not include Byte File System (BFS) byte range lock requests.

**Open Directory Requests**

The number of times the file pool server was requested to open a directory (to be subsequently used for reading).

**Open File New Requests**

The number of times the file pool server was requested to open a new SFS file (that is, create a new file).

**Open File Read Requests**

The number of times the file pool server was requested to open a file for read access.

**Open File Replace Requests**

The number of times the file pool server was requested to open an existing file such that existing file data is replaced with file data that is to be added. For SFS files, if the file does not exist, it is created.

**Open File Write Requests**

The number of times the file pool server was requested to open an SFS file for write access. (In this case, records can be changed or added to the file without affecting other records in the file.)

**Query Administrator Requests**

The number of times the file pool server was requested to provide information about users with file pool administrator authority.

**Query Connected Users Requests**

The number of times the file pool server was requested to provide information about users that were connected to it.

**Query Enrolled Users Requests**

The number of times the file pool server was requested to provide information about SFS enrolled users and BFS in the file pool.

**Query Lock Conflicts Requests**

The number of times the file pool server was requested to provide information about lock conflicts. This request is caused by the QUERY FILEPOOL CONFLICT command.

**Query File Pool Requests**

The number of times the file pool server was requested to provide information about the status of the file pool. This request is caused either by the QUERY FILEPOOL STATUS, QUERY FILEPOOL REPORT, QUERY FILEPOOL STORGRP, QUERY FILEPOOL OVERVIEW, QUERY FILEPOOL MINIDISK, QUERY FILEPOOL AGENT, QUERY FILEPOOL LOG, QUERY FILEPOOL CATALOG, QUERY FILEPOOL COUNTER, or QUERY FILEPOOL CRR command.

**Query User Space Requests**

The number of times the file pool server was requested to provide information about the space allocated to enrolled SFS users and Byte File Systems. This request is caused by the QUERY LIMITS

## QUERY FILEPOOL STATUS

command, DMSQLIMA Query Limits for all Enrolled Users CSL routine, or DMSQLIMU Query Limits for a Single User CSL routine.

### **Read File Requests**

The number of times the file pool server was requested to read data from files.

### **Recovery Close Catalog Requests**

The number of times the file pool server was requested to close a previously-opened catalog object. These requests are generally made by programs that recover user data.

### **Recovery Get Catalog Requests**

The number of times the file pool server was requested to get catalog information. These requests are generally made by programs that recover user data.

### **Recovery Open Catalog Requests**

The number of times the file pool server was requested to open a catalog object. These requests are generally made by programs that recover user data.

### **Recovery Put Catalog Requests**

The number of times the file pool server was requested to write to a catalog object. These requests are generally made by programs that recover user data.

### **Refresh Directory Requests**

The number of times the file pool server was requested to refresh the SFS information maintained in the user machine cache. CMS in the user machine decides when it is necessary to have the cache refreshed based on what the user is doing.

### **Relocate Requests**

The number of times the file pool server was requested to move a file or sub-tree from one SFS directory to another. This request is caused by the RELOCATE command or DMSRELOC CSL routine.

### **Rename Requests**

The number of times the file pool server was requested to rename an SFS file or directory. This request is caused by the RENAME command or DMSRENAM CSL routine.

### **Revoke Administrator Authorization Requests**

The number of times the file pool server was requested to delete a file pool administrator.

### **Revoke Authorization Requests**

The number of times the file pool server was requested to revoke authority on an SFS file or directory from a user.

### **Revoke User Requests**

The number of times the file pool server was requested to delete an SFS enrolled user or a BFS from the file pool.

### **Rollback Requests**

The number of times the file pool server was requested to roll back a logical unit of work. (These are known as voluntary rollbacks.)

### **Unlock Requests**

The number of times the file pool server was requested to delete an explicit lock on a file pool object. This counter does not include Byte File System (BFS) byte range unlock requests. This request is caused by DELETE LOCK command and ENABLE command.

### **Write Accounting Requests**

The number of times the file pool server was requested to write accounting records for a file pool.

### **Write File Requests**

The number of times the file pool server was requested to write to a file in the file pool.

### **Total File Pool Requests**

The total number of requests made to the file pool server during the current multiple user mode session. This includes all preceding requests **and** the requests at the bottom of **Counter Information** that follow Total I/O Requests (before the **CRR Counter Information** heading). This total includes **only** those counters whose names end with the word **Requests**.



**Total Byte File File Pool Requests**

This is the total number of requests identified by "Byte File" under the "SFS AND BYTE FILE COUNTER INFORMATION" section. This is a sub-total included in the 'Total File Pool Requests'.

**File Pool Request Service Time (msec)**

The total amount of time (in milliseconds) the file pool server took to process requests. To calculate the service time, the file pool server makes a note of the time whenever it receives a request from a user machine. When it sends the result of the request back to the user machine it again notes the time. It then subtracts the time the request was received from the time the response was sent to determine the service time for that particular request. This counter is the sum of the service times of all the requests the file pool server received after last being started.

**Local File Pool Requests**

The total number of file pool requests made by users on the same processor.

**Remote File Pool Requests**

The total number of file pool requests made by users on different processors.

**Alias Definitions Examined**

The number of times the file pool server had to read catalog information about SFS aliases during the processing of all requests.

**Alias Definitions Updated**

The total number of times an alias definition was created, changed, or deleted. An alias definition is changed when any of its attributes are changed (for example, number of records in the file, date, time, record format, and so on).

**Begin LUWs**

The total number of logical units of work (LUWs) the file pool server has started.

**Agent Holding Time (msec)**

The total amount of time (in milliseconds) the server spent holding user agents. The server calculates agent holding time by subtracting the time at which the agent is acquired from the time it is released (made available for other users). This counter is the sum of the agent holding times for all agents the server has used after it was last started. Note that this counter applies to both file pool servers and CRR recovery servers. For more information about agents, see [Active Agents Highest Value](#).

**LUW Rollbacks**

The total number of logical units of work (LUWs) the server rolled back.

Rollbacks can be classified as either voluntary or involuntary. Voluntary rollbacks are requested by the application and are counted by *Rollback Requests*. You can compute the number of involuntary rollbacks by subtracting *Rollback Requests* from *LUW Rollbacks*. You can additionally compute how many involuntary rollbacks were caused by reasons other than deadlock by subtracting *Rollbacks Due to Deadlock* from *LUW Rollbacks*.

**SAC Calls**

The total number of times the server called its Storage Access Component (SAC). SAC is the portion of server code that accesses the catalogs and user files. It also provides support for locking, catalog indexes, logging, file pool recovery, and file pool generation.

**Storage Group Explicit Lock Conflicts**

The number of times a request for a lock on a storage group was denied because an explicit lock had already been created on that storage group. The request for the lock could have been either an implicit or an explicit request.

**File Space Explicit Lock Conflicts**

The number of times a request for a lock on an SFS file space or Byte File System (BFS) was denied because an explicit lock had already been created for that file space or BFS. The request for the lock could have been either an implicit or an explicit request.

**Directory Explicit Lock Conflicts**

The number of times a request for a lock on a directory was denied because an explicit lock had already been created on that directory. The request for the lock could have been either an implicit or an explicit request.

### **File Explicit Lock Conflicts**

The number of times a request for a lock on a file was denied because an explicit lock had already been created on that file. The request for the lock could have been either an implicit or an explicit request. The counter excludes lock conflicts caused by byte range locking of byte files.

### **Storage Group Logical Lock Conflicts**

The number of times a request for a lock on a storage group was denied (or waited, if FILEWAIT was set ON for an explicit lock request) because an implicit lock had already been created on that storage group. The request for the lock could have been either an implicit or an explicit request.

### **File Space Logical Lock Conflicts**

The number of times a request for a lock on an SFS file space or Byte File System (BFS) was denied (or waited, if FILEWAIT was set to ON for an explicit lock request) because an implicit lock had already been created for that file space or BFS. The request for the lock could have been either an implicit or an explicit request.

### **Directory Logical Lock Conflicts**

The number of times a request for a lock on a directory was denied (or waited, if FILEWAIT was set to ON) because an implicit lock had already been created on that directory. The request for the lock could have been either an implicit or an explicit request.

### **File Logical Lock Conflicts**

The number of times a request for a lock on a file was denied (or waited, if FILEWAIT was set to ON) because an implicit lock had already been created on that file. The request for the lock could have been either an implicit or an explicit request. The counter excludes lock conflicts caused by byte range locking of byte files.

### **Catalog Lock Conflicts**

The number of times a request for a lock on a part of a catalog was delayed or denied because an implicit lock had already been created on the desired item. All lock requests on catalogs are implicit. A file pool server uses internal locking mechanisms on the catalogs to prevent simultaneous updating of the same catalog data. SFS server is designed so catalog locks are held for as brief a time as possible. In general, a file pool server does not deny a user request because of a catalog lock conflict. Instead, it waits for the lock to free and gives the next user control. Catalog locks are transparent to CMS users.

### **Total Lock Conflicts**

is the sum of the following counters:

- Storage Group Explicit Lock Conflicts
- File Space Explicit Lock Conflicts
- Directory Explicit Lock Conflicts
- File Explicit Lock Conflicts
- Storage Group Logical Lock Conflicts
- File Space Logical Lock Conflicts
- Directory Logical Lock Conflicts
- File Logical Lock Conflicts
- Catalog Lock Conflicts
- DIRCONTROL Resource Lock Conflicts
- File Recall Lock Conflicts
- Recall Exit Lock Conflicts

This counter excludes lock conflicts caused by byte range locking of byte files.

### **Lock Wait Time (msec)**

The total amount of time (in milliseconds) spent waiting for locks. For each case of a lock wait, the server calculates the *lock wait time* by subtracting the time at which it suspended processing the request from the time at which it resumed processing the request. This counter is the sum of the lock wait times for all lock wait conditions that occurred after the server was started. This counter excludes lock conflicts caused by byte range locking of byte files.

**Deadlocks**

The number of deadlocks the file pool server detected and corrected. A *deadlock* occurs when two applications are each holding file pool resources the other needs. For example, application A opens FILEA for writing. Application B opens FILEB for writing. Next application A tries to open FILEB for writing and waits (because it had set FILEWAIT to ON). Application B simultaneously tries to open FILEA for writing. Application B waits because it also had FILEWAIT set to ON. Each application is waiting for a file the other has implicitly locked. Neither can proceed. The file pool server detects this condition and rolls back the logical unit of work that started most recently. (The application gets an error return code.)

**Note:** The file pool server is able to resolve some deadlocks internally through a retry process. Consequently, not all deadlocks result in rollback. The number of rollbacks (because of deadlock) are counted by *Rollbacks Due to Deadlock*. The number of rollbacks (because of deadlock or other reasons) is counted by *LUW rollbacks*.

**QSAM Requests**

The number of QSAM requests the file pool server has made. QSAM requests occur for control data backups and restores. They also occur for security audit trace output.

**QSAM Time (msec)**

is the time (in milliseconds) the file pool server waited for all QSAM requests to complete. The file pool server notes the time immediately before making a QSAM request and immediately upon return from QSAM. The difference between the two times is the *QSAM Time* for a QSAM request.

**File Blocks Read**

The total number of 4KB blocks read from user storage groups.

**File Blocks Written**

The total number of 4KB blocks written to user storage groups.

**Catalog Blocks Read**

The total number of 4KB blocks read from the file pool catalogs (storage group 1).

**Catalog Blocks Written**

The total number of 4KB blocks written to the file pool catalogs (storage group 1).

**Control Minidisk Blocks Read**

The total number of 512-byte blocks read from the file pool control minidisk.

**Control Minidisk Blocks Written**

The total number of 512-byte blocks written to the file pool control minidisk.

**Log Blocks Read**

The total number of 4KB blocks read from the file poollog minidisks.

**Log Blocks Written**

The total number of 4KB blocks written to the file poollog minidisks.

**Total DASD Block Transfers**

The total number of file pool blocks read or written. (It is the sum of the preceding eight counters.)

**BIO Requests to Read File Blocks**

The total number of times block I/O was used to read blocks from user storage groups. Where possible, a file pool server tries to read multiple blocks using a single block I/O request. Therefore, this number is usually less than the *File Blocks Read*.

**BIO Requests to Write File Blocks**

The total number of times block I/O was used to write blocks to user storage groups. Where possible, a file pool server tries to write multiple blocks using a single block I/O request. Therefore, this number is usually less than the *File Blocks Written*.

**BIO Requests to Read Catalog Blocks**

The total number of times block I/O was used to read blocks from the file pool catalogs (storage group 1). The file pool server reads catalog blocks one at a time, as needed.

**BIO Requests to Write Catalog Blocks**

The total number of times block I/O was used to write blocks to the file pool catalogs (storage group 1). The SFS file pool server writes catalog blocks one at a time, as needed.

### **BIO Requests to Read Control Minidisk Blocks**

The total number of times block I/O was used to read blocks from the file pool control minidisk. In general, the SFS file pool server reads control minidisk blocks one at a time, as needed.

### **BIO Requests to Write Control Minidisk Blocks**

The total number of times block I/O was used to write blocks to the file pool control minidisk. Where possible, a SFS file pool server tries to write multiple blocks using a single block I/O request. Therefore, this number is usually less than the *Control Minidisk Blocks Written*.

### **BIO Requests to Read Log Blocks**

The total number of times block I/O was used to read blocks from the SFS log minidisks. The file pool server reads log minidisk blocks one at a time, as needed.

### **BIO Requests to Write Log Blocks**

The total number of times block I/O was used to write blocks to the file pool log minidisks. There are cases where the server can group multiple log blocks into one block I/O request. Therefore, this number may be less than *Log Blocks Written*.

### **Total BIO Requests**

The total number of times the file pool server used block I/O. This counter is the sum of the preceding eight counters.

### **Total BIO Request Time (msec)**

The total amount of time (in milliseconds) agents had to wait because they initiated a block I/O request in the file pool server. When a user agent requests a block I/O operation, the file pool server checks the processor clock, starts the operation, and (rather than waiting for the I/O to complete) begins doing work for other users. After handling requests for other active agents, and the block I/O request has completed, the file pool server dispatches the agent that was waiting. The file pool server then checks the clock again and subtracts the start time from this time to determine how long the user agent waited. It is this value that is added to the counter.

### **I/O Requests to Read File Blocks**

The number of I/O requests issued by CP to read blocks from user storage groups. This count will typically be larger than *BIO Requests to Read File Blocks* because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD except when minidisk caching is active and all requested blocks are found in the minidisk cache.

### **I/O Requests to Write File Blocks**

The number of I/O requests issued by CP to write blocks to user storage groups. This count will typically be larger than *BIO Requests to Write File Blocks* because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD.

### **I/O Requests to Read Catalog Blocks**

The number of I/O requests issued by CP to read blocks from the file pool catalogs (storage group 1). Each I/O request will normally result in a physical I/O to DASD except when minidisk caching is active and the requested block is found in the minidisk cache.

### **I/O Requests to Write Catalog Blocks**

The number of I/O requests issued by CP to write blocks to the file pool catalogs (storage group 1). Each I/O request will normally result in a physical I/O to DASD.

### **I/O Requests to Read Control Minidisk Blocks**

The number of I/O requests issued by CP to read blocks from the control minidisk. Each I/O request will normally result in a physical I/O to DASD.

### **I/O Requests to Write Control Minidisk Blocks**

The number of I/O requests issued by CP to write blocks to the control minidisk. This count will typically be larger than *BIO Requests to Write Control Minidisk Blocks* because a multiple block request containing blocks that reside on different cylinders is implemented in CP as separate I/O requests. Each I/O request will normally result in a physical I/O to DASD.

### **I/O Requests to Read Log Blocks**

The number of I/O requests issued by CP to read blocks from the file pool log minidisks. Each I/O request will normally result in a physical I/O to DASD except when minidisk caching is active and the

requested block is found in the minidisk cache. (Note: Minidisk caching should not be used for the log minidisks.)

### **I/O Requests to Write Log Blocks**

The number of I/O requests issued by CP to write blocks to the file pool log minidisks. Each I/O request will normally result in a physical I/O to DASD.

### **Total I/O Requests**

The approximate number of times the file pool server caused CP to process a channel program to read or write DASD blocks. This counter is the sum of the preceding eight counters.

### **Release Blocks Requests**

The total number of times the file pool server was requested to release (deallocate) blocks that encountered write I/O errors in a specified storage group.

### **Temporary Close Requests**

The total number of times the file pool server was requested to prepare an open file for commit processing.

### **SFS Send User Data Requests**

The total number of times the file pool server was requested to pass user data information to an external security manager.

### **Change File Attributes Requests**

The total number of times the file pool server was requested to modify the overwrite or recoverability attributes of SFS files.

### **Dataspace Requests**

The total number of times the file pool server was requested to assign or remove a directory from the data space eligibility list. This request is caused by the DATASPACE command.

### **Dirattr Requests**

The total number of times the file pool server was requested to change directory control attributes. This request is caused by the DIRATTR command and by the DMSDIRAT CSL routine.

### **Prepare Requests**

The number of times the file pool server was requested to do the first phase of a two-phase commit.

### **Query Accessors Requests**

The total number of times the file pool server was requested to return information about users who are accessing directory control directories. This request is caused by the QUERY ACCESSORS command.

### **Query Dataspace Requests**

The total number of times the file pool server was requested to return information about the eligibility of directories for use in data spaces. This request is caused by the QUERY DATASPACE command.

### **Set Reference Date Requests**

The number of times CMS requested the file pool server to change the date of last reference for files that are in a directory mapped to a data space.

**Note:** When a directory is mapped to a data space, CMS in the user machine references the data space directly. Because the file pool server is not used for the file access, it does not know that a reference occurred. To maintain a correct date of last reference for the file, the user machine tells the file pool server machine to update its catalogs (and the data space) with the new date of last reference.

### **Rollbacks Due to Deadlock**

The total number of SFS logical units of work (LUWs) and Byte File System (BFS) rolled back because of a deadlock that could not be corrected through a retry process.

### **DIRCONTROL Resource Lock Conflicts**

The number of times a request to lock a directory control directory was delayed or denied because an implicit lock had already been created on that directory. SFS uses internal locking mechanisms to maintain the consistency of directory control directories.

### **Precoordination Requests**

The total number of times the file pool server was called during the precoordination phase of a syncpoint to see if a user's file space is still exceeded. (The Precoordination request is issued only if a user's file space was exceeded during the current LUW).

### **Connect User Requests**

The number of times the server was requested to establish a connection to a file pool for the specified user ID, or set the associated user ID for an already established connection to the specified user ID or the number of times the server was requested to connect user IDs that differ from the user ID of the connecting machine.

### **Create External Object Requests**

The total number of times the file pool server was requested to create an external object in an SFS directory. This request is caused by the DMSCROB CSL routine.

### **Create File Requests**

The number of times the file pool server was requested to add a new (empty) file to an SFS directory. This request is caused by the DMSCRFIL CSL routine or the CREATE FILE command.

### **Query User Storage Group Requests**

The total number of times the file pool server was requested to query a file pool for information about user storage groups. This request is caused by the DMSQUSG CSL routine.

### **DMSSFSEX Exit Calls**

The number of times the file pool server called the DMSSFSEX CSL routine. When usage of SFS file space, a byte file space (BFS), or a storage group reaches a predefined point, the file pool server will call one of the two exits of this type: File Space Usage or User Storage Group Full. The exits are called when the file pool server calls the IBM-supplied CSL routine DMSSFSEX.

### **DMSSFSEX Exit Time (msec)**

The total amount of elapsed time (in milliseconds) spent in routine processing for DMSSFSEX requests.

### **File Recall Lock Conflicts**

The number of times a request for a lock involving a DFSMS/VM File Recall resource was delayed or denied because an implicit lock had already been created on the desired item.

### **Recall Exit Lock Conflicts**

The number of times a request for a lock involving a DFSMS/VM Recall Exit resource was delayed or denied because an implicit lock has already been created on that item.

### **Change DFSMS Related Attribute Requests**

The number of times the file pool server was requested to change the DFSMS/VM Related Attributes (DRAs) for a specified file or directory.

### **Migrate Requests**

The number of times the file pool server was requested (using the DFSMS MIGRATE or DFSMS MANAGE command) to open a file or alias and prepare to migrate file blocks.

### **Recall Requests**

The number of times the file pool server was requested (using the DFSMS RECALL command, or automatic recalls when files in migrated status are referenced) to open a file or alias and prepare to recall file blocks from migrated status.

### **Send DFSMS Data Requests**

The number of times the file pool server was requested to pass user-defined data to the DFSMS/VM exit.

### **Recall DFSMS File Exit Calls**

The number of times the file pool server called a DFSMS/VM Recall File exit point (to automatically recall a file in migrated status that has been referenced).

### **Recall DFSMS File Exit Time (msec)**

The total amount of elapsed time (in milliseconds) spent in processing DFSMS/VM Recall File Exit calls.

### **Other DFSMS Exit Calls**

The total number of times the file pool server called one of the following predefined DFSMS/VM exits:

- File pool Initialization Exit
- Acquire DRAs (DFSMS/VM Related Attributes)
- Erase File Notification
- Send DFSMS/VM Data Exit
- DFSMS/VM Termination Exit

#### **Other DFSMS Exit Time (msec)**

The total amount of time (in milliseconds) spent in processing one of the following predefined DFSMS/VM exits:

- File pool Initialization Exit
- Acquire DRAs (DFSMS/VM Related Attributes)
- Erase File Notification
- Send DFSMS/VM Data Exit
- DFSMS/VM Termination Exit

#### **Rename Userid Requests**

The number of times the file pool server was requested to rename a file space. This request is made by the FILEPOOL RENAME command.

#### **File Pool Control Backup Requests**

The number of times the file pool server was requested to take a control data backup using the FILEPOOL CONTROL BACKUP command.

#### **Add Minidisk Requests**

The number of times the file pool server was requested to add minidisk(s) while the server is in multiple user mode. This request is caused by the FILEPOOL MINIDISK command.

#### **Query Disable Requests**

The number of times the file pool server was requested to query a storage group or a file space or all file spaces and all storage groups in a file pool to determine if they have been previously disabled. This request is caused by the QUERY FILEPOOL DISABLE command, or DMSQFPDS CSL routine, or QUERY DISABLE operator command.

#### **Locks Denied Due to Timeout**

The number of times a lock could not be obtained within the timeout interval. When a lock is waited for, an internal timer may be set to prevent excessive wait times. If the timer expires, the lock is denied, and the request is unsuccessful. For example, it is possible for applications to commit changes to two file pools together using the DMSCOMM. If two applications were to commit changes to the same two file pools at the same time, it would be possible to create a deadlock with application A holding the locks needed by B on one of the file pools, and B holding locks needed by A on the other. In such a situation, one of the lock requests would eventually be timed out.

#### **Virtual Storage Reclaim Value**

The total number of times the file pool server has reclaimed its unused free storage. Periodically, when the server is running low on storage, it performs a reclaim process to free up additional storage.

#### **Open File CreateMig Requests**

The number of times the file pool server was requested to create a file in migrated status (DMSOPBLK CSL routine with the CREATMIG intent).

#### **Byte File Cancel Requests**

This is the number of times the file pool was requested to cancel a specified Byte File request.

#### **Byte File Change Access/Modification Time Requests**

This is the number of times the file pool was requested to change the access or modification times of a Byte File Object.

#### **Byte File Change Audit Requests**

This is the number of times the file pool was requested to change the audit flags for a Byte File Object.

### **Byte File Change Mode Requests**

This is the number of times the file pool was requested to change the mode associated with a Byte File Object.

### **Byte File Change Owner Requests**

This is the number of times the file pool was requested to change the owner (UID/GID) of a Byte File Object.

### **Byte File Check File Accessibility Requests**

This is the number of times the file pool was requested to check the accessibility of a Byte File Object.

### **Byte File Close Directory Requests**

This is the number of times the file pool was requested to close a directory.

### **Byte File Close File Requests**

This is the number of times the file pool was requested to close a regular file.

### **Byte File Create Block Special File Requests**

This is the number of times the file pool was requested to create a block special file.

### **Byte File Create Character Special File Requests**

This is the number of times the file pool was requested to create a character special file.

### **Byte File Create Directory Requests**

This is the number of times the file pool was requested to create a Byte File directory.

### **Byte File Create External Link Requests**

This is the number of times the file pool was requested to create an external link.

### **Byte File Create Link Requests**

This is the number of times the file pool was requested to create a hard link.

### **Byte File Create Named Pipe (FIFO) Requests**

This is the number of times the file pool was requested to create a named pipe (FIFO).

### **Byte File Create Regular file Requests**

This is the number of times the file pool was requested to create a regular Byte File.

### **Byte File Create Symbolic Link Requests**

This is the number of times the file pool was requested to create a symbolic link.

### **Byte File Lock Byte Requests**

This is the number of times the file pool was requested to lock a byte range.

### **Byte File Lookup Requests**

This is the number of times the file pool was requested to lookup a Byte File Object.

### **Byte File Makecat Requests**

This is the number of times the file pool was requested to make a Byte File Object.

### **Byte File Open Directory Requests**

This is the number of times the file pool was requested to open a directory (to be subsequently used for reading).

### **Byte File Open File New With Intent Read Requests**

This is the number of times the file pool was requested to create a new file with intent READ.

### **Byte File Open File New With Intent Write Requests**

This is the number of times the file pool was requested to create a new file with intent WRITE.

### **Byte File Open File Read Requests**

This is the number of times the file pool was requested to open a file for read access.

### **Byte File Open File Write Requests**

This is the number of times the file pool was requested to open a file for write or trunc access.

### **Byte File Pipe Access Requests**

This is the number of times the file pool was requested to verify the access authorization to a named pipe.

### **Byte File Pipe Close Requests**

This is the number of times the FIFO file pool was requested to close a named pipe.



**Byte File Pipe Open For Read Requests**

This is the number of times the FIFO file pool was requested to open a named pipe for read.

**Byte File Pipe Open For Write Requests**

This is the number of times the FIFO file pool was requested to open a named pipe for write.

**Byte File Pipe Read Requests**

This is the number of times the FIFO file pool was requested to read from a named pipe.

**Byte File Pipe Stat Requests**

This is the number of times the FIFO file pool was requested to obtain the current status information about a named pipe.

**Byte File Pipe Utime Requests**

This is the number of times the file pool was requested to update the timestamps associated with a named pipe.

**Byte File Pipe Write Requests**

This is the number of times the FIFO file pool was requested to write to a named pipe.

**Byte File Read Directory Entry Requests**

This is the number of times the file pool was requested to read directory entries.

**Byte File Read File Requests**

This is the number of times the file pool was requested to read data from files.

**Byte File Read Link Contents Requests**

This is the number of times the file pool was requested to read the contents of a link.

**Byte File Remove Directory Requests**

This is the number of times the file pool was requested to remove a directory.

**Byte File Rename Requests**

This is the number of times the file pool was requested to rename a Byte File Object.

**Byte File Test Locked Bytes Requests**

This is the number of times the file pool was requested to test byte range locks held on a specific byte range.

**Byte File Token Return Requests**

This is the number of times a client requested the file pool to free a vnode token or block token.

**Byte File Unlink Requests**

This is the number of times the file pool was requested to remove a Byte File Object.

**Byte File Unlinked File Cleanup Requests**

This is the number of unlinked files removed during FILESERV START.

**Byte File Unlock Byte Requests**

This is the number of times the file pool was requested to unlock a byte range.

**Byte File Write File Requests**

This is the number of times the file pool was requested to write to a regular file.

**Byte File ZAPCAT Requests**

This is the number of times the file pool was requested to read or alter catalogs using the ZAPCAT request.

**Byte File Directory Creation/Deletion Logical Lock Conflicts**

This is the number of times a request for a lock on an object (file, directory, link or symbolic link) to be created or deleted was denied (or waited for) because an implicit lock was already held on the object in a Byte File System (BFS).

**Byte File File Logical Lock Conflicts**

This is the number of times a request for a lock, unlock, or close on a file for serializing byte range lock/unlock and file closes was denied (or waited for) because an implicit lock had already been created on that file. The request for the lock was from an implicit request.

**Byte File Global Storage Logical Lock Conflicts**

This is the number of times a request for a lock on the object was denied (or waited for) because an implicit lock was already held on the object.

**Byte File NAMECAT Unallocated Logical Lock Conflicts**

This is the number of times a request for a lock on an unallocated NAMECAT row was denied because the implicit lock was already held on the row.

**Byte File Token Manager Logical Lock Conflicts**

This is the number of times the Token Manager requested a WRITE VNODE lock but had to wait for the lock because the implicit lock was already held.

**Total Byte File Lock Conflicts**

This is the sum of the following counters: Byte File Directory Creation/Deletion Logical Lock Conflicts, Byte File File Logical Lock Conflicts, Byte File Global Storage Logical Lock Conflicts, Byte File NAMECAT Unallocated Logical Lock Conflicts, Byte File Token Manager Logical Lock Conflicts.

**Byte File Byte Range Lock Waits**

This is the number of times a Byte Range Lock Request had to wait before being awarded the requested lock.

**Byte File Token Conflicts Causing Callbacks**

This is a count of the number of callbacks of tokens due to token conflicts. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token that is held by another user, the requestor must wait until the client machine that holds the token returns it (responds to a callback of that token).

**Byte File Callback Wait Time (msecs)**

This is the time (in milliseconds) spent waiting for callbacks of tokens. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token held by another user, the requestor must wait until the client machine that holds the token returns it (responds to a callback of that token).

**Byte File Token Callback Timeout Retries**

This is a count of the number of retries of callbacks because of a short delay of the holding client machine to respond to the token callback request. This is a retry attempt by the BFS server. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token held by another user, the requestor must wait until the client machine that holds the token returns it (responds to a callback of that token).

**Byte File Token Callback Requestor Retries**

This is a count of the number of requestor retries because of extended delays in call back response. This occurs when it is necessary to give up waiting for a normal callback completion because of exceeding the retry limit for callback retries (see "Token Callback Timeout Retries" counter). Return code to requestor suggests retry by the client application. Tokens control shared resources and data caching for byte file system clients/users, much like locks. When a byte file request requires a token held by another user, the requestor must wait until the client machine that holds the token returns it (responds to a callback of that token).

**Byte File Logical Lock Retries**

This is the number of times a retry was attempted to obtain the BFS requests logical locks.

**Byte File Logical Lock Retries Exceeded**

This is the number of times the request was denied logical locks due to the lock retry count being exceeded.

The CRR counter information is displayed whether the file pool server is a CRR recovery server or not. If the server is not a CRR recovery server, the CRR counter values are all zero.

Under *CRR Counter Information* you see:

CRR Counter Information

```

0 Get Capability Requests
0 Get Logname Requests
0 Get LUWID Requests
0 Resync Init Requests
0 Resync Query Direction Requests
0 Resync Protocol Violations Requests
0 Write Log Requests
0 Total CRR Requests
0 CRR Request Service Time (msec)
    
```

```

0 Syncpoints
0 Syncpoint Time (msec)
0 Participating Resources
0 Log Checkpoints taken
0 Log I/O Requests
0 BIO Request Time (msec)

```

The Counters are reset every time server processing is started. The *CRR Counter Information* fields have the following meanings:

#### **Get Capability Requests**

is the total number of times the CRR recovery server had requests to get the capabilities of a protected conversation partner in a coordinated commit.

#### **Get Logname Requests**

is the total number of times the CRR recovery server had requests to get the CRR recovery server's log name.

#### **Get LUWID Requests**

is the total number of times the CRR recovery server had requests to create a new SNA LU 6.2 logical unit of work ID (LUWID).

#### **Resync Init Requests**

is the total number of times the CRR recovery server had requests to do resynchronization activity for a resource, following a resource error during a coordinated commit.

#### **Resync Query Direction Requests**

is the total number of times the CRR recovery server had requests to determine the sync point direction (commit or rollback) by means of resynchronization activity, following an error during communications with the sync point initiator.

#### **Resync Protocol Violations Requests**

is the total number of times the CRR recovery server was given information a communications protocol violation was detected during a synchronization (sync) point.

#### **Write Log Requests**

is the total number of times the CRR recovery server had requests to write to the CRR log.

#### **Total CRR Requests**

is the total number of CRR recovery server requests (that is, the sum of the preceding CRR recovery server counts).

#### **CRR Request Service Time (msec)**

is the total amount of time (in milliseconds) the CRR recovery server spent handling CRR requests. For each CRR request, the server calculates the holding time by subtracting the time at which the request begins from the time it ends. This counter is the sum of all CRR requests the CRR recovery server processed after it was last started.

#### **Syncpoints**

is the total number of times the CRR recovery server was requested to process a CRR sync point.

#### **Syncpoint Time (msec)**

is the total amount of time (in milliseconds) the CRR recovery server spent handling sync points. For each sync point request, the server calculates the holding time by subtracting the time at which the request begins from the time it ends. This counter is the sum of all sync point requests the CRR recovery server processed after it was last started.

#### **Participating Resources**

The number of resources that have participated in coordination or recovery.

**Note:**  $\text{Participating Resources} \div \text{Syncpoints}$  is the average number of different resource managers that have participated in each sync point.

#### **Log Checkpoints Taken**

is the total number of times the CRR recovery server had done a CRR log checkpoint.

#### **Log I/O Requests**

The number of I/O requests issued by CP to read or write CRR recovery server log minidisk blocks. Each I/O request will normally result in a physical I/O to DASD.

**BIO Request Time (msec)**

is the total amount of time (in milliseconds) agents had to wait because they initiated block I/O requests to the CRR logs. When a user agent initiates a BIO request, the server checks the processor clock, starts the operation, and (rather than waiting for the BIO request to complete), begins doing work for other users. After handling requests for other active agents, and the block I/O request has completed, the server dispatches the agent that was waiting. The server then checks the clock again and subtracts the start time from this time to determine how long the agent waited. It is this value that is added to the counter.

Under *File Pool Compare Information* you see:

```
File Pool Compare Information
      3 Active Agents Highest Value
      9 Connections Highest Value
  3394 Virtual Storage Highest Value
```

The *File Pool Compare Information* fields have the following meanings:

**Active Agents Highest Value**

This is the highest number of *user agents* that were concurrently in use during the current multiple user mode session. The primary use of agents is to function as the server's internal representation of a user. When a user requests something of the server, the server assigns that user to an agent and does the work. After completing the work, the server frees the agent and can reuse it for another user.

User agents are also used implicitly for certain functions. When the server acts as a CRR recover server, user agents are used during the processing of various CRR requests from users and other server machines.

When the server is initialized, it computes the number of user agents it should create by using the following formula based on the USERS value:

```
number_of_agents = 4 + truncate(USERS/8)
```

For more information about Active Agents Highest Value, see [“Limit Monitoring” on page 58](#).

**Connections Highest Value**

This is the highest number of APPC/VM (and IUCV) connections concurrently in use by the server machine. MAXCONN should be set higher than the value in this counter (perhaps by 50%). If this counter is equal to the MAXCONN setting, the server machine may have been rejecting connection requests to the file pool or recovery server.

**Note:** The MAXCONN value is an operand of the OPTION system directory control statement. It defines the maximum number of Inter-User Communication Vehicle (IUCV) connections (including APPC/VM connections) for the server machine. For more information, see [“Step 4: Define a Server Machine” on page 256](#).

**Virtual Storage Highest Value**

This is the greatest amount of virtual storage used at any one time (in KB). This value includes all virtual storage acquired by the server after it was last started, including all virtual storage in use prior to FILESERV START. If this value approaches the virtual machine size, you should increase the virtual storage size of the server machine.

Under *File Pool Information* you see the maximums that were established for the file pool during file pool generation (or regeneration):

```
File Pool Information
      500 Maximum Number of Storage Groups
      500 Maximum Number of Minidisks
  25100288 Potential Addressable 4K Blocks in File Pool
```

To exceed any of these maximums, you must regenerate the file pool. The *File Pool Information* fields have the following meanings:

**Maximum Number of Storage Groups**

This field shows the maximum number of storage groups that may be defined in the file pool. To exceed this maximum you must regenerate the file pool.

**Maximum Number of Minidisks**

This field shows the maximum number of minidisks that can be assigned to all the file pool *storage groups*. The minidisk that holds the POOLDEF file, the log minidisks, and the control minidisk do not count towards this maximum. In the above example, 500 is shown. Before adding the five hundred-and-first minidisk, you would have to regenerate the file pool with a larger MAXDISKS value.

**Potential Addressable 4K Blocks in the File Pool**

This value tells you the number of 4KB blocks the file pool can manage in all its storage groups. The potential addressable 4KB blocks value is proportionally related to the size of the control minidisk. The word *potential* is used because the file pool may be able to handle (address) more 4KB blocks than are currently allocated to it. For instance, the file pool may be able to handle 30000 4KB blocks, but currently only has 15000 4KB blocks of physical minidisk space allocated to it. As you add minidisks to the file pool, the number of physical blocks grows. If you try to add more physical blocks than the file pool can handle, the add minidisk (FILESERV MINIDISK command or FILEPOOL MINIDISK command) operation is unsuccessful. To add the space, you need to regenerate the file pool. A complete discussion of how to do this is in [Chapter 11, “Regenerating a Repository File Pool,” on page 217.](#)

To determine the total number of physical blocks currently allocated to the file pool, add the number of *4K Blocks In-Use* to the *4K Blocks Free* for each minidisk in the file pool storage groups. The difference between this sum and the *Potential Addressable 4K Blocks in the File Pool* is the approximate number of physical 4KB blocks that can still be added to the file pool.

The next group is *Currently Defined Minidisk Information*:

Currently Defined Minidisk Information				
Minidisk Number	Group Number	4K Blocks In-Use	4K Blocks Free	
xxxxx	xxxxx	xxxxxxxxxx - yyy%	xxxxxxxxxx	
xxxxx	xxxxx	xxxxxxxxxx - yyy%	xxxxxxxxxx	

For each minidisk allocated to a file pool storage group, you see:

**Minidisk Number**

The number of the minidisk in the file pool. It is obtained from the numeric portion of the MDKnnnnn minidisk identifier in the POOLDEF file. This number indicates the order in which minidisks were added to the file pool.

**Group Number**

The storage group to which the minidisk belongs. Remember storage group 1 contains the catalog data. All other storage groups contain user data.

**4K Blocks In-Use**

The number of minidisk 4KB blocks that are currently in use (that is, have data in them). The number of in-use blocks is also shown as a percentage of the total number of minidisk blocks.

**4K Blocks Free**

The number of minidisk 4KB blocks not being used. A low value across the storage group indicates it may soon be necessary to add minidisks to this storage group (or move users out of it, for user storage groups).

The *Agent Information* shows information about the users or internal processes for which the server is currently doing work:

Agent Information					
Number of Agents		xxxxx			
Userid	Type	Status	Agent Number	Wait	Uncommitted Blks
uuuuuuuu	Chkpt	Read	xxxxx	None	xxxxxxxxx
	User	Write		Communication	
	Syncpt-Logr	Wait		Checkpoint	
	Resync-Task	Inact		Lock	
	Pc->Resync	Running		Catalog_Buffer	

Rm→Resync	Prepared	Control_Buffer
Resync→Rm		I/O
BFS_User		Connection
		Timer
		Subtask
		Multi-Event
		Suspended
		AVSwait
		ESM_Wait
		DFSMS/VM_Wait
		xx
		Storage_Wait
		Token_Callback

An *agent* is the server's internal representation of a user (or system task). There are usually far fewer agents than logged-on users. The server calculates how many agents it needs based on what is specified in the USERS startup parameter.

The *Number of Agents* in the display is the number of user agents for which the server is currently doing work. It is not the total number of user agents the server has created internally. To calculate the number of user agents the server has created internally, use the formula explained in [“Monitoring the USERS Startup Parameter”](#) on page 59.

For each agent, you see:

**Userid**

is the user ID of the virtual machine for which the server is doing work. If the file pool server is a doing CRR tasks, the user ID could be @RESYNC.

**Type**

Can be:

**Chkpt**

means file pool *checkpoint*. A *checkpoint* is an internal server operation during which the changes recorded on the log minidisks are permanently made to the file pool. The file pool server takes checkpoints after a certain number of log minidisk blocks have been written.

**User**

means the file pool server is doing work for the user indicated in the *Userid* column.

**Syncpt-Logr**

means a CRR recovery server task dedicated to logging synchronization points. Various states of sync point processing are logged through this agent.

**Resync-Task**

means resynchronization task. The server creates a special type of agent to accomplish resynchronization and other tasks not directly related to the user. These tasks may be initiated by either the CRR recovery server or the file pool server.

**Pc → Resync**

means conversation from one CRR recovery server to another CRR recovery server in support of a protected conversation to perform an initial exchange of log names or resynchronization. This type applies to a user ID of @RESYNC.

**Rm → Resync**

is a CRR recovery server agent. A participating resource manager, such as a file pool server, is communicating with the CRR recovery server to perform an initial exchange of log names in support of resynchronization initialization processing. This type applies to a user ID of @RESYNC.

**Resync → Rm**

is a file pool agent. The CRR recovery server is communicating with the file pool server to perform an exchange of log names in support of resynchronization recovery processing. This type applies to a user ID of @RESYNC.

**BFS\_User**

means the file pool server is doing Byte File System (BFS) work for the user indicated in the *Userid* column.

**Status**

Can be:

**Read**

means the server has not yet written anything to the file pool log minidisks for the user. This applies to the following types of agent:

- User
- Syncpt-Logr
- Resync-Task
- Pc → Resync
- Rm → Resync
- Resync → Rm
- BFS\_User

**Write**

Means the file pool server has written log information for the user. This applies to the following types of agent:

- User
- Syncpt-Logr
- Resync-Task
- Pc → Resync
- Rm → Resync
- Resync → Rm
- BFS\_User

**Wait**

Means the file pool checkpoint process is waiting for the file pool server to complete some work for other users. This applies only to the "checkpoint" agent.

**Inact**

means the file pool checkpoint process is dormant. (The file pool server is not doing a checkpoint.) This applies only to the *checkpoint* agent.

**Running**

means the file pool server is running the checkpoint process. Other user tasks remain dormant until it completes. This applies only to the *checkpoint* agent.

**Prepared**

means the file pool server is waiting for the second phase of the two-phase commit. This applies only to a *user* type of agent.

**Agent Number**

is an internal number by which the server knows the agent.

**Wait**

Can be:

**None**

means the agent is not waiting for anything.

**Communication**

for a repository file pool, means that agent is waiting for communication between the user machine for which it is processing and the server machine to complete. This typically means the agent has responded to the last request from the user machine and is now still in a logical unit of work and waiting for the next request. The agent could also be waiting for communication with

an external security manager (if the ESECURITY startup parameter is in effect) or a CRR recovery server.

For a CRR recovery server, means the agent is waiting for APPC communication from:

- User machine for CRR logging
- Participating resource manager for resynchronization initialization
- Another CRR recovery server for resynchronization recovery

### **Checkpoint**

means the agent is waiting for an internal server activity, known as a checkpoint, to complete.

This wait state applies to repository file pool servers.

### **Lock**

means the agent is waiting for a server resource to become available. When a server resource, such as a file or catalog information, is not available because it is being used by someone else, the resource is said to be *locked*.

### **Catalog\_Buffer**

means the agent is waiting for a catalog I/O buffer to become free. (All catalog buffers are currently in use.) The number of catalog buffers available is set by the CATBUFFERS startup parameter. This wait state applies to repository file pool servers.

### **Control\_Buffer**

means the agent is waiting for a control minidisk I/O buffer to become free. (All control buffers are currently in use.) The number of control buffers available is set by the CTLBUFFERS startup parameter.

This wait state applies to repository file pool servers.

### **I/O**

means the agent is waiting for a file pool or CRR log I/O operation to complete on its behalf.

### **Connection**

means the first phase of the two-phase commit has completed, and communication with the SFS file pool server has been severed. The prepared user is waiting for resynchronization, which is initiated by the CRR recovery server.

This wait state applies to repository file pool servers.

### **Timer**

is displayed when the agent is waiting for a time period to elapse (as specified by the RESYNCINTERVAL startup parameter or defaults to 10 minutes) before the periodic retry of resynchronization in an attempt to complete the update of a protected resource.

This wait state applies only to CRR recovery servers.

### **Subtask**

is displayed when the agent sends a request to the SFS file pool server, which must be suspended pending the completion of an SFS function (for example, the initial exchange of log names with the user's CRR recovery server). The SFS function is handled as a subtask, which is not directly related to the user's request that resulted in the subtask being initiated. While the subtask is executing, the original user's request is suspended in this wait state.

This wait state applies only to repository file pool servers.

### **Multi-Event**

is displayed when the CRR recovery server communicates with multiple participating resource managers (such as the file pool server) and waits for a response from a participating resource manager.

This wait state applies only to CRR recovery servers.



**Suspended**

is displayed when the agent is temporarily put into a waiting state by the CRR SUSPEND operator command. This wait state prevents the periodic retry of resynchronization.

This wait state applies only to CRR recovery servers.

**AVSwait**

is displayed when the agent is waiting for a response from AVS. The AVS virtual machine is the agent's communication partner.

This wait state applies only to CRR recovery servers.

**ESM\_Wait**

is displayed when the agent is waiting for an External Security Manager (ESM) response. This wait state applies only to repository file pool servers.

**DFSMS/VM\_Wait**

is displayed when the agent is waiting for a DFSMS/VM response. This wait state applies only to repository file pool servers.

**xx**

is an internal number which means the SFS file pool server is at a higher release level than the CMS in your machine, and the wait is a type that is unknown to CMS.

**Storage\_Wait**

is displayed when the agent is waiting for storage to be available on the server.

**Token\_Callback**

is displayed when the file pool server is waiting for the Byte File System (BFS) client machine to respond to a request to invalidate cached file pool information.

**Uncommitted Blks**

is the number of SFS file blocks used by this agent that have not yet been committed or rolled back. An inordinately high number in this column could indicate an errant program is writing data in an endless loop (this is commonly referred to as a *run-away write*). You can implement exits to handle scenarios like this one. For more information, see [“Activating the Storage Use Exits \(SFS and BFS\)” on page 209](#). You can use the QUERY LIMITS command to determine the storage group to which the user is assigned. Note that this field has meaning (nonzero value) only when the type column value is *user*.

The *Log Information* shows information about the file pool log minidisks:

Log Information	
1188	Number of Log Minidisk 4K Blocks
14%	Percent(%) of Log Space Used
95%	LUW Rollback/Suspend Threshold (% Log Space)
80%	Backup Threshold (% Log Space)

The fields have the following meanings:

**Number of Log Minidisk 4K Blocks**

is the total number of 4KB blocks on each log minidisk. Both log minidisks are the same size.

**Percent(%) of Log Space Used**

is the percent of the total log space currently used.

When BACKUP is in effect, log space is freed when the control data is backed up. When NOBACKUP is in effect, log space is freed during *checkpoint* processing. A checkpoint is an internal process during which all committed changes are permanently written to the file pool. Checkpoints occur after numerous log pages are written. A checkpoint also occurs when multiple user mode operation is stopped normally (using any variation of the STOP operator command except STOP IMMEDIATE).

Note that control data backups and checkpoints done in multiple user mode may not free all the log space. Because the log is written sequentially, the only space that can be freed is the space used prior to the earliest uncommitted logical unit of work at the time of the checkpoint or control data backup.

**LUW Rollback/Suspend Threshold (% Log Space)**

If the BACKUP startup parameter is in effect, this is the percentage at which the server will begin suspending the processing of SFS logical units of work and Byte File System (BFS) requests. Because the server starts an automatic control data backup when the log is 80% full, an automatic backup is always in progress when the *LUW Rollback/Suspend* percentage is reached. The server suspends work at this percentage to give the backup a chance to complete. If it did not begin suspending work at this point, the logs could fill before the control data backup has a chance to complete (which would free log space). In other words, the server knows that log space will soon be free. Instead of risking a log-full (stopping) condition, the server suspends processing when the *LUW Rollback/Suspend Threshold* is reached.

Note that it is still possible for the logs to fill during a control data backup. Because the server is multi-tasking, it must schedule the suspension of activity. (Certain in-progress activities cannot be interrupted.) It is possible the logs could fill before the server has a chance to stop all the in-progress work. While it is unlikely this will happen, it is most likely to occur if there is a great deal of concurrent activity and the logs are small. In this case, you would enter a FILESERV BACKUP command to back up the control data and free the log space. Then you can restart multiple user mode operation.

If NOBACKUP is in effect, the *LUW Rollback/Suspend Threshold* is the percentage at which the server will begin rolling back long-running file pool server requests. When NOBACKUP is in effect, the server attempts to free log space whenever a checkpoint occurs. (Usually checkpoints occur after several log blocks have been written.) Log space cannot be freed if there is a long-running file pool server request still in progress. If the log reaches the *LUW Rollback/Suspend Threshold*, it rolls back long-running SFS logical units of work and BFS requests so the next checkpoint will free some space. The server's goal is to prevent the log from filling, which causes the server to stop. There is a chance the logs could fill even though the file pool server is rolling back logical units of work. In this case, the server stops. To recover, you would restart multiple user mode processing. Log recovery processing should free enough log space for continued operation.

**Backup Threshold (% Log Space)**

indicates the percentage at which a control data backup will automatically be started. This value is displayed only when the BACKUP startup parameter is in effect. If you do not see this value in your display, the NOBACKUP startup parameter is in effect.

When you specify the CATALOG option, you also receive file pool catalog information, as follows:

```
Catalog Space Information
42519 Data Blocks
 249 Occupied Data Blocks
  1% Percent Occupied Data Blocks
42521 Index Blocks
 216 Occupied Index Blocks
  1% Percent Used Index Blocks
```

The fields in the *Catalog Space Information* have the following meanings:

**Data Blocks**

is the total number of addressable 4096-byte blocks that can be used to hold information about base files, directories, aliases, enrolled users, and so on. Note that this number represents potential addressable blocks, not the number of DASD blocks allocated to the storage group. The amount of physical DASD space allocated to storage group 1 can be determined from the *Currently Defined Minidisk Information* earlier in the QUERY FILEPOOL STATUS display.

**Occupied Data Blocks**

is the number of data blocks that actually have data in them. An occupied block is not necessarily full. It may have room for additional data.

**Percent Occupied Data Blocks**

is calculated from the preceding two values.

**Index Blocks**

is the total number of addressable 4096-byte blocks that can be used to hold index values and pointers to data blocks. Note that this number represents potential addressable blocks, not the number of DASD blocks allocated to the storage group. The amount of physical DASD space allocated

to storage group 1 can be determined from the *Currently Defined Minidisk Information* earlier in the QUERY FILEPOOL STATUS display.

### Occupied Index Blocks

is the number of index blocks that actually have data in them. An occupied block is not necessarily full. It may have room for additional data.

### Percent Used Index Blocks

is calculated from the preceding two values.

The number of potential addressable data and index blocks is determined during file pool generation (FILESERV GENERATE) or regeneration (FILESERV REGENERATE). During these processes, the file pool server calculates the total number of addressable catalog blocks by using the MAXUSERS value in a space estimation formula. It internally allocates a few of these blocks for header information and splits the remaining blocks evenly between data blocks and index blocks.

When you add a minidisk to the catalog storage group, the server does not immediately allocate these blocks for use as index blocks or data blocks. Instead, it allocates the blocks as needed. If your file pool usage is such that 80% of all catalog storage group blocks are needed for data, that is what the file pool server attempts to get from the physical DASD allocated to the storage group.

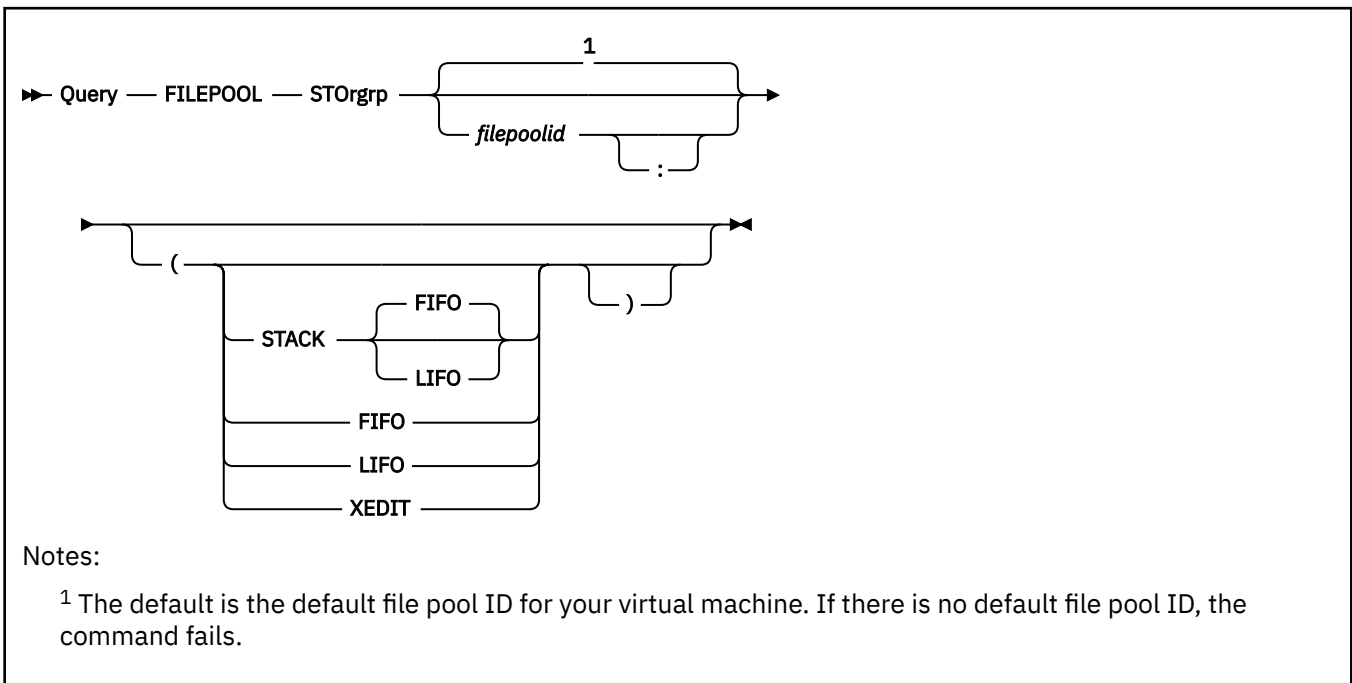
## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect filepoolid: *filepoolid* [RC=24]
- DMS391E Unexpected operand(s): *operands* [RC=24]
- DMS639E Error in NUCXLOAD routine; return code was *return code* [RC=24]
- DMS688E XEDIT option only valid from XEDIT environment [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24]
- DMS1223E There is no default file pool currently defined [RC=40]

## QUERY FILEPOOL STORGRP



### Authorization

File Pool Administrator or Connected User

### Purpose

Use the `QUERY FILEPOOL STORGRP` command to display information on storage groups. This information includes storage group numbers, 4KB blocks in use, percent usage, and 4KB blocks free.

The information returned can assist you with the task of performance analysis and timing for repository file pool servers and CRR recovery servers.

Note that this command should not be used by permanent applications to acquire data. The content of the output from this command may change in the future.

### Operands

*filepoolid*

***filepoolid:***

is the name of the file pool. If not specified, the default file pool for your virtual machine is used.

### Options

**STACK FIFO**

**STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. The headings also are stacked. FIFO is the default.

**FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to `STACK FIFO`.

**LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to `STACK LIFO`.

**XEDIT**

indicates the QUERY output should be placed in the file currently being edited (using XEDIT). The edited file must either be fixed format with a logical record length (LRECL) of 80 or variable length format. The information replaces the current line and below until the END OF FILE is reached. Then the remaining text (if any) is inserted before the END OF FILE.

**Usage Notes**

1. The XEDIT option is useful for scrolling through the data and for saving it for later browsing. You might also consider entering this command when FULLSCREEN is set to ON. CMS FULLSCREEN also lets you scroll through the information.
2. The XEDIT option must be used from within an XEDIT session. An easy way to use the option is to edit a new file and then enter the QUERY FILEPOOL STORGRP command with the XEDIT option on the command line. When a new file is edited, there is no chance of accidentally overlaying information already in the file.
3. If the QUERY FILEPOOL STORGRP command is issued from an exec or an assembler program for a file pool that is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.
4. If the current release of z/VM is processing prior release information, you will only see the information applicable for that prior release.

**Responses**

The QUERY FILEPOOL STORGRP command displays information about storage groups.

An example of the output is:

```

VM9SERV File Pool Storage Groups
Start-up Date dd/mm/yy          Query Date dd/mm/yy
Start-up Time hh:mm:ss         Query Time hh:mm:ss
=====
STORAGE GROUP INFORMATION
Storage      4K Blocks      4K Blocks
Group No.   In-Use           Free
   1         296 - 34%        580
   2         264 - 18%        1218
=====

```

For a description of the fields, see [STORAGE GROUP MINIDISK TOTALS](#) of the command “[QUERY FILEPOOL REPORT](#)” on page 592.

**Messages and Return Codes**

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

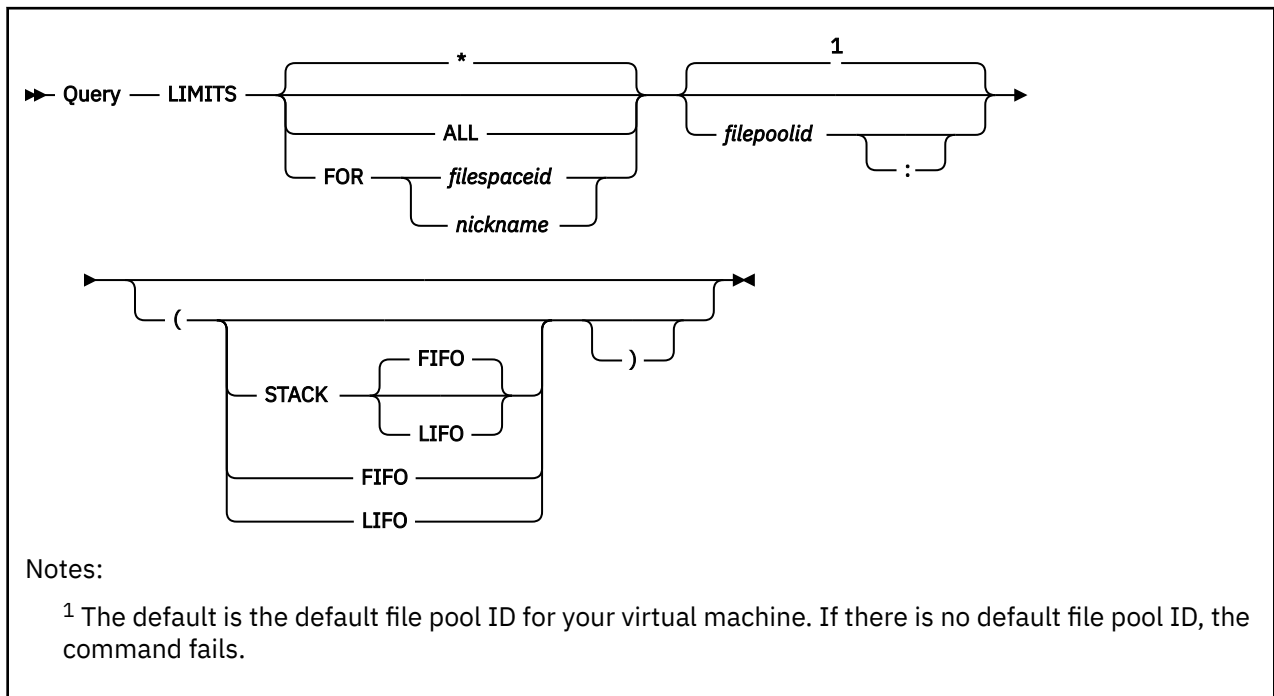
Messages:

- DMS065E *option* option specified twice [RC=24]
- DMS105S Error *nn* writing file *fn ft fm* to XEDIT [RC=31, 55, 70, 76, 99, or 100]
- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS389E Incorrect filepoolid: *filepoolid* [RC=24 ]
- DMS391E Unexpected operand(s): *operands* [RC=24 ]
- DMS639E Error in NUCXLOAD routine; return code was *nnnn* [RC=24]
- DMS689E File must be F-format 80 or V-format [RC=24 ]
- DMS1223E There is no default file pool currently defined [RC=40 ]

## QUERY FILEPOOL STORGRP

- DMS688E XEDIT option only valid from XEDIT environment [RC=24]

# QUERY LIMITS



## Authorization

File Pool Administrator or Connected User

## Purpose

The QUERY LIMITS command displays the following information about selected users in a particular file pool:

- File space ID
- Storage group to which the file space is assigned
- Number of 4KB file blocks allocated to the file space
- Number of 4KB file blocks currently used by the file space
- Warning threshold set on 4KB file blocks used for the file space

Any file pool user can enter this command for the SFS file space or for a Byte File System (BFS) (a file space created using the ENROLL USER command with the BFS option). This command provides function equivalent to the CSL Routines DMSQLIMA - SFS Query Limits and DMSQLIMU - SFS Query Limits - Single User which are described in [z/VM: CMS Callable Services Reference](#).

## Operands

**\***

indicates limits should be displayed for the file space with the same name as the issuing user ID. the command. This is the default.

**ALL**

means information for all file spaces should be returned.

**FOR filepaceid**

indicates the name of the file space for which limits are to be displayed.

## QUERY LIMITS

### **FOR *nickname***

identifies a nickname that represents a user ID, a BFS, or a list of user IDs or BFS names. (Use the NAMES command to define nicknames.)

### ***filepoolid***

#### ***filepoolid:***

identifies the file pool. If not specified, the default file pool ID for your virtual machine is used.

## Options

### **STACK FIFO**

### **STACK LIFO**

places the information in the console stack rather than displaying it at the terminal. FIFO is the default.

### **FIFO**

specifies the information is stacked in a first in, first out order. This option is equivalent to STACK FIFO.

### **LIFO**

specifies the information is stacked in a last in, first out order. This option is equivalent to STACK LIFO.

## Usage Notes

### **Common Notes for SFS and BFS**

1. When the STACK option is in effect, the column heading information is not produced. Only the data is put on the stack.
2. If you specify a nickname and the NODE tag in the NAMES file indicates the user is on another processor, the LOCALID tag must also be specified.
3. If limits are requested for a file space ID not enrolled, the output shows the file space ID and, in every other column of output, dashes.
4. If the *4K Blocks Committed* value is 100%, 99% will be displayed in that field.
5. If the QUERY LIMITS command is issued from an exec or assembler program, and the file pool specified is active (that is, has some work on it which has not been committed or rolled back) on the current default work unit, the command will cause an error.

### **Notes for SFS only**

1. The *threshold* is the point at which a user will receive a warning message indicating he has exceeded a certain percentage of his allocated space. The threshold is initially set to 90% when the user is enrolled. Users can change this value by entering a SET THRESHOLD command.
2. Note that the number of 4K blocks committed may not all have been actually written by the user who *owns* the file space. If a user grants write authority to someone else, another user may have written the blocks.
3. The following are all valid ways to query your own limits:
  - QUERY LIMITS
  - QUERY LIMITS \*
  - QUERY LIMITS *filepoolid*
  - QUERY LIMITS \* *filepoolid*
  - QUERY LIMITS FOR *myuserid*
  - QUERY LIMITS FOR *myuserid filepoolid*

### **Notes for BFS only**

1. Threshold is not defined or maintained and is displayed as 100%.



## Responses

Suppose you enter the following command to check the limits of user JOHN:

```
q limits for john
```

You see a display similar to the following:

Userid	Storage Group	4K Block Limit	4K Blocks Committed	Threshold
JOHN	2	1000	820-82%	90%

The following is an example of the output for a user not enrolled.

```
q limits for joe
```

Userid	Storage Group	4K Block Limit	4K Blocks Committed	Threshold
JOE	-	-	-	-

The following is an example of the output for:

```
q limits for dev
```

Note that DEV is a nickname.

Userid	Storage Group	4K Block Limit	4K Blocks Committed	Threshold
JOHN	3	1000	820-82%	90%
MIKE	3	2000	1620-80%	90%
CHRIS	-	-	-	-
MARY	3	4000	3600-82%	90%
SCOTT	-	-	-	-
RON	3	1000	820-82%	90%

## Messages and Return Codes

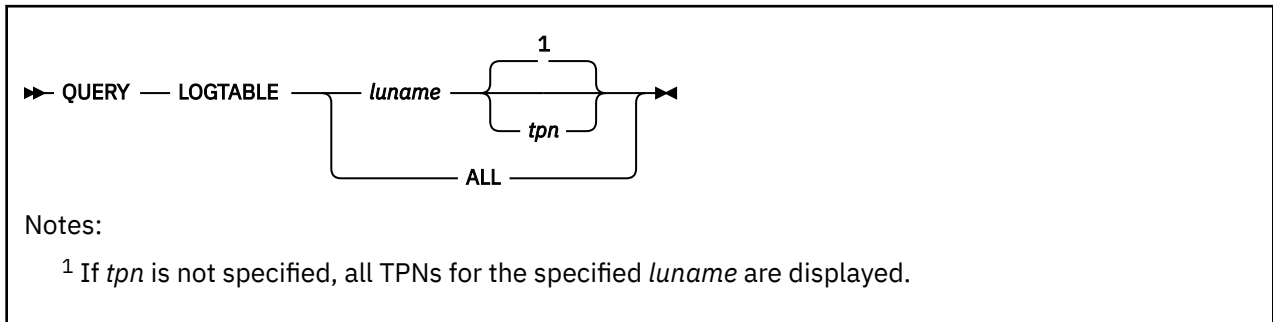
In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

This command internally issues IDENTIFY, FINIS, NUCXDROP, and NUCXLOAD commands. Therefore, messages from those commands can also be displayed. (For more information about those commands, see [z/VM: CMS Commands and Utilities Reference](#).)

Messages:

- DMS109S Virtual storage capacity exceeded [RC=109]
- DMS149E Userid *filespaceid* not valid [RC=32]
- DMS391E Unexpected operand(s): *operand* [RC=24]
- DMS637E Missing node ID for the AT operand [RC=24]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=32]
- DMS653E Error executing *command rc=nn* [RC=40]
- DMS1202E Userid must not be specified if *{ALL/\*}* is specified [RC=24]
- DMS1205I No users are enrolled in file pool *filepoolid* [RC=0]
- DMS1223E There is no default file pool currently defined lbrk.RC=40]
- DMS1238E Missing userid for *operand operand* [RC=24]
- DMS1239E You are not authorized to enter this request on behalf of *filespaceid* [RC=76]
- DMS1239E You are not authorized to enter this request for ALL users [RC=76]

## QUERY LOGTABLE



### Authorization

File Pool Operator

### Purpose

Use the QUERY LOGTABLE SFS operator command to display the specified LU name and transaction program name (TPN) for entries in the SFS log name table. The QUERY LOGTABLE command provides information about CRR recovery servers with which SFS has exchanged log names. This information can be useful for using the ERASE LUNAME command when changing identifiers such as gateway names or resource IDs. For more information about names that have changed, see [“SFS Log Name Table Changes”](#) on page 283.

**Note:** The TPN is sometimes called resource ID.

### Operands

***luname***

***luname tpn***

displays information about all SFS log name table entries with an LU name specified by *luname* and optionally displays TPNs specified by *tpn*. If *tpn* is not specified, the QUERY LOGTABLE command displays all TPNs for the specified *luname*. You can specify the *luname* as \*LOCAL to display TPNs for all LU names on this system, TSAF collection or CF collection.

**ALL**

displays information about all SFS log name table entries.

### Usage Notes

1. If *tpn* is not specified, the command will display information about all log name table entries with the specified *luname*. For example, if the file pool operator issues:

```
query logtable gdlvm8.recover
```

the display returns all SFS log name table entries that have the fully qualified LU name of GDLVM8.RECOVER.

### Responses

The following is an example of the information that could be displayed for multiple log name table entries found:

Remote LU Name	Transaction Program Name	Local LU Name	Active
Time: hh:mm:ss			
Date: mm/dd/yy			
QUERY LOGTABLE - <i>filepoolid</i>			

<i>luname</i>	<i>tpn</i>	<i>luname</i>	Y N
.	.	.	.
<i>luname</i>	<i>tpn</i>	<i>luname</i>	Y N

**Time and Date**

are the local time (*hh:mm:ss*) and date (*mm/dd/yy*) the QUERY LOGTABLE command was issued.

**QUERY LOGTABLE**

identifies the command issued and the file pool ID associated with this SFS file pool server.

**Remote LU Name**

is a heading followed by a list of the fully qualified LU name(s) associated with the CRR recovery server(s) from the SFS log name table, where the name(s) consist of:

- Optional SNA network ID
- Period of the SNA network ID is supplied
- LU name

For TPNs that are on this system or TSAF collection, the LU name is displayed as \*LOCAL.

**Transaction Program Name**

is a heading followed by a list of transaction program names (*tpn*) that are associated with the CRR recovery server at this *luname*.

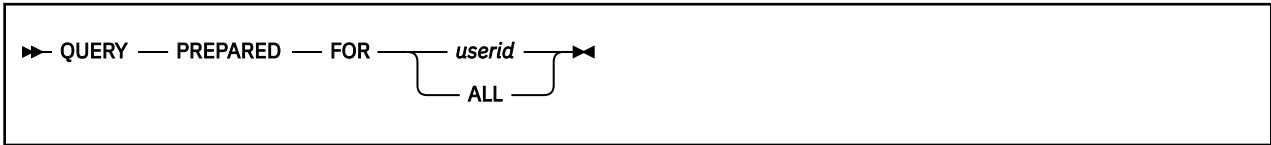
**Local LU Name**

is a heading followed by a list of local fully qualified LU names which are sent to the associated CRR recovery server as part of the log name exchange.

**Active**

is a heading followed by a Y if the associated entry currently has prepared work active or an N if the associated entry does not have prepared work active. Additional information about active prepared work can be obtained by using the QUERY PREPARED command, specifying the remote LU name and TPN given in this entry.

## QUERY PREPARED



### Authorization

File Pool Operator

### Purpose

Use the QUERY PREPARED operator command to display information about SFS prepared work and forced work. You must enter the QUERY PREPARED command before you enter the FORCE PREPARED command. The *taskid* resulting from QUERY PREPARED is used as input to the FORCE PREPARED command.

### Operands

#### *userid*

indicates which user information is to be displayed. Do not specify a nickname for *userid*. Nicknames are not recognized on server operator commands.

#### ALL

means the information for all prepared or forced users is to be displayed.

### Usage Notes

1. This command is used by an operator to display information about any SFS work in a prepared state and is waiting to be either committed or rolled back. Prepared work can be prepared-and-connected work or prepared-and-not-connected work. CRR tries to automatically complete the prepared-and-not-connected work by doing resynchronization processing.  
Prepared-and-connected work is resolved by usual sync point processing.
2. This command is used to display information about any SFS work that was prepared and forced by a previous FORCE PREPARED command.
3. The CRR QUERY LUWID command lists resources in work for a user for an LUWID instance. If one of the resources is a file pool, the QUERY PREPARED command can be used to determine additional information about the state of that work. Usually, if the CRR QUERY LUWID command shows the state to be in-doubt, the QUERY PREPARED command should show a state of prepared for that logical unit of work. An exception to this is if the file pool has determined that it is actually in read-mode (there are no updates) for the user's logical unit of work. In this case, no prepared work is displayed. This mismatch does not cause any processing errors.

### Responses

The following is an example of the information that could be displayed for prepared work by issuing the QUERY PREPARED command.

```

Time: hh:mm:ss           QUERY PREPARED - filepoolid
Date: mm/dd/yy

Access Userid
access_userid           Time: hh:mm:ss           Taskid: nnnnn
                        LUWID: nnnnnnnn.llllllll.iiiiiiiiiii.ssss
                        LUNAME: nnnnnnnn
                        TPN: .llllllll
Recovery Token: nnnnnnnn           State: current_state
    
```

Transaction Tag: *This is a transaction tag record.*

### Time and Date

are the local time (*hh:mm:ss*) and date (*mm/dd/yy*) the QUERY PREPARED command was entered.

### QUERY PREPARED

identifies the command entered and the file pool ID associated with this server.

### Access Userid

*access\_userid* is the virtual machine ID sent to SFS. The holder of this ID is either enrolled in SFS or otherwise known to be authorized.

### Time

is the time (*hh:mm:ss*) when the prepare to commit was received.

### Taskid

*nnnnn* is the agent number associated with prepared work. It is only shown when the state is prepared. Taskid is not displayed when the state is forced. You can use the FORCE PREPARED command to force prepared work.

### LUWID

is the logical unit of work ID associated with the entire coordinated transaction where:

- *nnnnnnnn.lllllllll* is the fully qualified LU name where the LUWID was created that consists of:
  - *nnnnnnnn* is the optional SNA network ID
  - *lllllllll* is the LU name
- *iiiiiiiiiii* is the instance number
- *ssss* is the sequence number

### LUNAME

is the LU name associated with the CRR recovery server.

**Note:** If the file pool server and the CRR recovery server reside in the same processor or TSAF or CS collection, the **LUNAME** will be *\*LOCAL*. For updating distributed data, where the file pool server is on a different processor (outside the TSAF or CS collection), the **LUNAME** will **not** be *\*LOCAL*.

### TPN

is the transaction program name used to identify the CRR recovery server. The LU name part of the value of the LUNAME startup parameter for the recovery server derives this TPN for the CRR recovery server.

### Recovery Token

*nnnnnnnn* is the SFS transaction ID. It is used by SFS to identify the LUWID to the CRR recovery server. This is the same Recovery Token displayed on the CRR QUERY LUWID operator command.

### State

*current\_state* is the indication of the status of the SFS work and can be one of the following:

- FORCED BACKOUT – The work has been backed out by resynchronization or by the FORCE command.
- FORCED COMMITTED – The work has been committed by resynchronization or by the FORCE command.
- PREPARED AND CONNECTED – The first phase of the two-phase commit has completed. The work is waiting for the second phase to commit or back out.
- PREPARED AND NOT CONNECTED – The first phase of the two-phase commit has completed, and the communication with the server has been severed. The prepared work is waiting for resynchronization.

### Transaction Tag

is an optional field of 1 – 80 bytes that can be used by the application program to give information about the transaction. The *Transaction Tag* is set by using the Set Transaction Tag (DMSSETAG) CSL

## QUERY PREPARED

routine or the Get Work Unit (DMSGETWU) CSL routine. For more information on DMSSETAG and DMSGETWU, see [z/VM: CMS Callable Services Reference](#).

## REVOKE ADMIN

```
➤ REVOKE — ADMIN — userid ➤
```

### Authorization

File Pool Operator

### Purpose

Use the REVOKE ADMIN operator command to delete file pool administration authority from a user.

### Operands

#### *userid*

is the virtual machine identifier from which file pool administration authority will be removed. Do not specify a nickname for *userid*. Nicknames are not recognized on server operator commands.

### Usage Notes

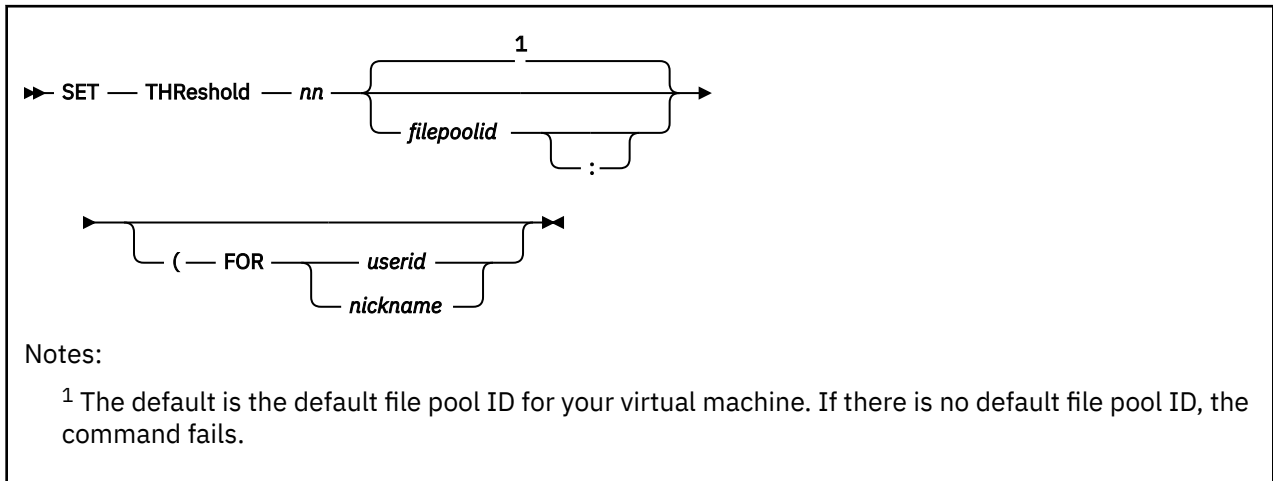
1. All messages are written to the server machine operator console.
2. The user ID will lose administration authority after any in-process logical unit of work ends and before the user ID begins the next logical unit of work.
3. When REVOKE ADMIN is entered, any explicit locks acquired by the specified user ID will not be automatically deleted. The user can remove them if he or she is still explicitly enrolled or if PUBLIC is enrolled. Or, another file pool administrator can delete the locks.

When administrator authority is deleted, the user may find he or she cannot use the QUERY LOCK command to see locks he or she created. This happens if the user has no explicit authority on the object and was able to create the lock only by virtue of administration authority. After administration authority is revoked, the QUERY LOCK command does not display the locks for the user because the user has no authorization on the file or directory. Although the user cannot display the lock, the user can still enter a DELETE LOCK command to delete it.

4. If the specified administrator is accessing other users' directories (for which he or she has no explicit authorizations), those directories remain accessed until the end of the CMS session.
 

**Note:** All authority checking for a DIRCONTROL directory is on access. Therefore, if a user accesses a DIRCONTROL directory by his administrator authority and his administrator authority is revoked, his authority on that directory remains until access of the DIRCONTROL directory is released.
5. For FILECONTROL directories, if a user has opened a file he has authority for because he is an administrator, and his administrator authority is revoked before his changes are committed, he can still commit the data but will be unable to reopen the file.
6. If the revoked user ID is specified in an ADMIN startup parameter, that user ID will be given administration authority again the next time server processing is started.

## SET THRESHOLD



### Authorization

File Pool Administrator

### Purpose

Use the SET THRESHOLD command to indicate when a warning message and return code is issued which indicates a specified amount of allocated file space in a file pool has been used.

While administrator authority is not required when this command is issued to set one's own file space threshold, it is needed if you want to change the file space threshold of another user.

### Operands

#### *nn*

is a number between 1 and 99 inclusive, indicating when this percentage of your allocated file space is used, you want a warning message.

#### *filepoolid*

#### *filepoolid:*

specifies in which file pool to set the threshold. If not specified, the default file pool identifier is used. Note that the colon is optional.

#### **FOR *userid***

#### **FOR *nickname***

indicates the threshold is to be set for the specified user's file space, as opposed to the file space for the user issuing the SET THRESHOLD command.

The *nickname* identifies a nickname that represents a user ID. The *nickname* may not represent a list of users. (Use the NAMES command to define nicknames.)

### Usage Notes

1. A default threshold percentage of 90% is set when you are enrolled with file space in a file pool.
2. Enter QUERY LIMITS \* to obtain information about your threshold setting and usage of the file space available to you.
3. The threshold warning message is issued only to the user whose application has exceeded the file space threshold limit. The administrator does not get the warning message.
4. The warning message or return code continues to be issued whenever you enter a command that affects your file space until you are below the threshold. For example, you receive the threshold



warning and then erase a file in your file space. If you are still at or above the threshold level, you will receive another threshold warning. However, you will only see this message one time between console reads. (For example, a console read occurs when you press Enter, so you would only see the message one time until you press enter again even if you are executing a program which performs multiple operations which exceed the threshold.)

5. If the SET THRESHOLD command is issued from an exec or assembler program for a file pool that is active on the specified work unit, the command fails.
6. If you are using OS simulation when writing to files in a pre-z/VM Release SFS server, the user threshold warning is treated as a disk full error. This could occur on a write or put operation. This could also occur on a read operation because CMS buffers requests.

## Examples

### Example 1:

If you have been allocated 1000 blocks of file space and you enter:

```
set threshold 95
```

the threshold is set at 95%. You will get a warning message when 950 or more blocks of your default file pool file space are used.

### Example 2:

Suppose an administrator creates a file space named TEMP with PUBLIC write authorization to its top directory. By default, this file space has a threshold percentage of 90%. There is no user on the system with a user ID of TEMP. If a user in the file pool writes to the TEMP file space and causes 90% or more of the file space to be used, that user receives a warning message the file space is 90% used. The administrator may then be notified and can modify the contents of the file space to free up some space, or the administrator can enter a SET THRESHOLD command and increase the threshold setting so the warning message no longer displays. For example, to change the threshold setting of TEMP to 95, the administrator enters:

```
set threshold 95 (FOR temp
```

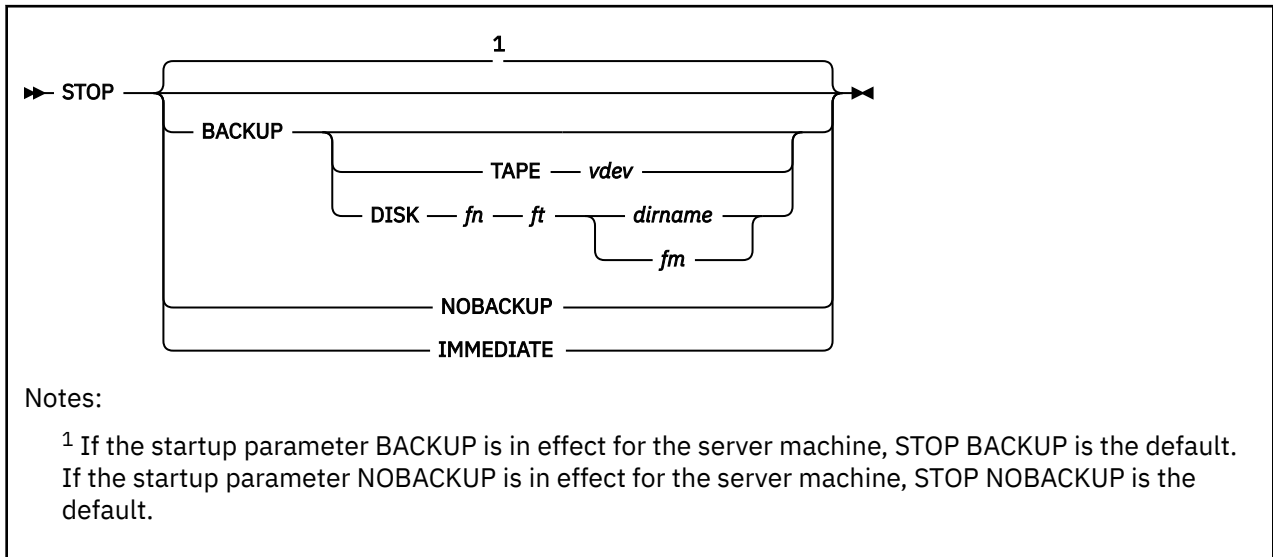
## Messages and Return Codes

In addition to the messages listed below, this command displays other system messages. These system messages are listed in [z/VM: CMS Commands and Utilities Reference](#).

Messages:

- DMS149E Userid *userid* not valid [RC=32]
- DMS637E Missing *nodeid* for the AT operand [RC=24]
- DMS647E Userid not specified for *nickname* in *userid* NAMES file [RC=32]
- DMS647E Localid not specified for *userid* at *node* in *user's* NAMES file [RC=32]
- DMS653E Error executing *command*, rc=*nn* [RC=40]
- DMS1140E You are not enrolled in file pool *filepoolid* [RC=40]
- DMS1141W User filespace threshold still exceeded for the file pool *filepoolid* [RC=4]
- DMS1167E Userid *userid* is not enrolled [RC=40]
- DMS1168E Incorrect threshold value *value* [RC=24]
- DMS1209E Nickname *nickname* resolved to more than one user ID; the threshold limit can be set for only one file space at a time [RC=88]

## STOP



### Authorization

File Pool Operator

### Purpose

Use the STOP operator command to stop multiple user mode processing for a file pool server.

### Operands

#### BACKUP

indicates server shutdown should occur after:

- All SFS users have completed in-process logical units of work
- All Byte File System (BFS) users have completed in-process file pool server requests
- All currently logged CRR recovery server sync points have completed
- All SFS and CRR work in resynchronization is completed
- Control data backup is taken

If the startup parameter BACKUP is in effect for the server machine, STOP BACKUP is the default.

#### TAPE *vdev*

identifies the virtual device number *vdev* for a standard label (SL) tape file. The control data backup will be directed to the identified tape device.

#### DISK *fn ft dirname*

#### DISK *fn ft fm*

identifies a CMS file to which the control data backup will be directed. The file can reside on a minidisk or in another file pool. The *dirname* can be the fully qualified directory name *filepoolid:userid.n1.n2..n8* of the SFS directory where the backup file is to be created. The *fm* may be the file mode of an accessed minidisk or SFS directory.

#### NOBACKUP

indicates server shutdown should occur after:

- All users have completed in-process logical units of work
- All currently logged CRR recovery server sync points have completed

- All SFS and CRR work in resynchronization is completed

No file pool backup is taken after all user processing ends.

If the startup parameter NOBACKUP was specified, STOP NOBACKUP is the default.

### IMMEDIATE

For SFS usage, indicates server shutdown should occur even if some SFS users have not completed in-process logical units of work or CRR recovery server logged syncpoints have not completed. All SFS in-process logical units of work will be rolled back or resynchronized the next time multiple user mode processing is started.

For BFS usage, indicates server shutdown should occur even if some users' in-process file pool server requests have not completed. All in-process file pool server requests will be rolled back the next time multiple user mode processing is started.

During CRR synchronization (sync) point processing, SFS logical units of work that completed the first phase of the two-phase commit will go into resynchronization the next time multiple user mode processing is started. Any SFS logical units of work that did not complete the first phase are rolled back and the user receives a return code that indicates they have to rerun this work the next time multiple user mode processing is started.

For the CRR recovery server, when the server is restarted, any logged sync points that have not completed phase one of the two-phase commit are discarded. Sync points that have completed phase one of the two-phase commit will be resynchronized.

The control data is not backed up.

### Usage Notes

1. All messages are written to the server machine operator console.
2. For CRR recovery servers, IBM does not recommend doing backups of control data (that is, use the NOBACKUP startup parameter).
3. All new file pool server connections, new SFS logical units of work, and new Byte File System (BFS) file pool server requests are disallowed when the STOP command is entered.
4. For a list of supported virtual device numbers for tapes, when you enter the STOP command with BACKUP in effect, see the description of the FILEDEF command in *z/VM: CMS Commands and Utilities Reference*.
5. When you enter the STOP command without the IMMEDIATE operand, you will have to wait for any in-process work to complete. This in-process work must complete before any control data backup is performed.

In-process SFS logical units of work will be allowed to complete. In-process BFS file pool server requests will be allowed to complete. Note that with CRR support, some SFS logical units of work may have to wait for CRR resynchronization in order to complete. For more information on SFS participation in CRR, see [Chapter 17, "Participation in CRR \(SFS only\),"](#) on page 279.

In-process CRR recovery server logged sync points will be allowed to complete. Note that some logged sync points may have to wait for CRR resynchronization in order to complete. For more information on CRR resynchronization, see ["Resynchronization Function"](#) on page 305.

When you enter the STOP command, the server issues message DMS3029I telling you how many SFS logical units of work, how many BFS requests, and how many CRR recovery server sync points are in process. As SFS logical units of work, BFS requests, and sync points complete, this message is redisplayed. This allows you to intervene with other file pool operator commands to obtain status or to assist in completing the in-process work. You can also enter the STOP IMMEDIATE command.

6. New sync points, logged by the CRR recovery server, cannot be started.
7. When IMMEDIATE is not specified, file pool server processing severs user connections as each connected user ends his or her SFS logical unit of work or BFS file pool server request. Connections that do not have an in-process SFS logical unit of work or a BFS in-process file pool server request are immediately severed.

## STOP

8. The server first verifies the integrity of catalog space data on the control minidisk when backup processing is to be performed. Any catalog verification error will cause the server to stop without performing the control data backup.
9. If the startup parameter NOBACKUP was specified and a STOP BACKUP operator command is issued, a message will be displayed indicating the BACKUP operand is ignored. STOP processing will proceed.
10. Server processing will create the file pool control data backup file either at the default destination currently in effect or at the destination explicitly specified on the STOP BACKUP command itself. The default destination for the backup file is originally determined from the contents of the POOLDEF file but this could have been changed with the DEFBACKUP operator command.
11. If a control data backup is made and the backup file resides on a minidisk, STOP BACKUP processing uses a temporary file during the backup. This file is named \$\$TEMP \$BACKUP. If STOP BACKUP processing completes successfully, the file is erased. If STOP BACKUP processing fails and the backup file resides on a minidisk, follow the recovery actions in [“Special Considerations for Control Data Backups to CMS File”](#) on page 106.
12. If the control backup is directed to tape, it is assumed the tape will be ready when the backup starts. If it is not, message DMS113S will be issued and control backup will stop.
13. It is not allowed to direct the control backup to a file that already has established data definitions using the FILEDEF command. If this is attempted, backup processing will stop. Note that the file pool server enters a FILEDEF command for security audit (AUDIT) when AUDIT is specified in the DMSPARMS file.
14. You do not need to enter a FILEDEF for ddname=BACKUP. The server creates the proper environment for the backup processing.
15. Unless a file pool server is explicitly disabled from receiving a shutdown signal from CP (by setting by the NOSHUTDOWN SIGNAL startup parameter; SHUTDOWN SIGNAL is the default), a CP SHUTDOWN, FORCE, or SIGNAL command causes the server to automatically issue the STOP command (with no operands).

If you receive a message from the z/VM system operator informing you of a pending system shutdown (or re-IPL), and you have disabled your server for automatic shutdown, be sure to enter the STOP or STOP IMMEDIATE command before the z/VM system shutdown occurs.

If the z/VM system shutdown occurs and you do not enter the STOP or STOP IMMEDIATE command for the CRR recovery server, CRR redoes the resynchronization work when the system is re-IPLed. This causes the CRR recovery server's operator console to receive resynchronization messages for work already done.

Issuing the STOP command causes an orderly termination of the servers and prevents the redisplay of resynchronization messages, but this command does not cause an orderly termination of the servers (in-process file pool repository work and CRR sync points are completed). Issuing the STOP command causes an orderly termination of the servers and prevents the redisplay of resynchronization messages (in-process SFS work, in-process BFS file pool server requests, and CRR sync points are allowed to complete).

---

## Appendix A. Sample Execs for SFS Administration

This appendix contains four sample execs you might find useful for administering a file pool. The first exec, named SFSTRANS, transfers the ownership of file pool objects from one user ID to another. This exec is useful when you are changing a person's user ID. The second exec, named REGRANT, regrants file and directory authorities for a user who has changed his user ID. The third exec, named TALLY, provides a summary of the physical space allocated to a file pool. The fourth exec, named WHO, summarizes space consumption for file pool users.

**Note:** These sample execs are stored in executable form on the system 193 disk.

## SFSTRANS EXEC

The SFSTRANS EXEC transfers the ownership of file pool objects from one user ID to another user ID. It duplicates the directory structure, all files, and all related authorizations of the *old* user ID to the *new* user ID. It accepts two arguments as input: the *old* user ID and the *new* user ID. For example, to transfer ownership from JONES to JONESR, you would enter:

```
sfstrans jones jonesr
```

Your default file pool ID is used.

You can use the SFSTRANS EXEC as a model for your own exec. With some modifications, the SFSTRANS EXEC would also be useful for moving a user from one file pool to another. (See [“Overview of Moving Users and File Spaces”](#) on page 95.)

```

/*****
/* COPYRIGHT - */
/*          5684-112 (C) COPYRIGHT IBM CORP. 1988, 1991 */
/*          LICENSED MATERIALS - PROPERTY OF IBM */
/*          SEE COPYRIGHT INSTRUCTIONS, G120-2083 */
/*          */
/* This programming example is to be used as a sample program only. */
/* Although this program may have been reviewed by IBM */
/* for accuracy in a specific environment, there is no */
/* guarantee that the same or similar results will be obtained */
/* elsewhere. The code is being provided on an 'As is' basis */
/* without any warranty expressed or implied. */
/*          */
/* Program name - SFSTRANS EXEC */
/*          */
/* Function - Transfer ownership of all directories and files from */
/*          userid1 to userid2. This */
/*          exec must be run on a userid with Shared File System */
/*          Administrator authority. Userid2 must already be */
/*          enrolled. This exec will use filemodes B and C. */
/*          After this exec has completed, a DELETE USER command */
/*          must be processed to delete the old userid from the */
/*          filepool. */
/*          */
/* Command format is: */
/*          */
/*          SFSTRANS  userid1 userid2 */
/*          */
/* Note: The FILEPOOL RENAME command should be used to rename */
/*          an SFS file space. This exec is provided as a sample */
/*          of how to use SFS functions to perform this type of */
/*          operation from a REXX program. */
/*          */
/*****
Parse Arg userid1 userid2 .
If userid2 = '' Then Do
    say 'Parameter missing'
    Exit
End
Upper userid1
Call Transfer_Directories
Call Transfer_Files
Exit

Transfer_Directories:
/*****
/* Routine name - Transfer_Directories */
/*          */
/* Function - Creates all subdirectories on userid2 just as they */
/*          were on userid1. */
/*****
dirid = userid1||'.'
queued = queued()
'MAKEBUF'
'LISTDIR' dirid '(FIFO' /* Get the list of directories*/
If rc = 0 Then Exit
queued = queued() - queued
Do queued
    pull . dirname .
    Address Command 'LISTDIR' dirname '(LIFO DIRC NOSUBDIR'

```

```

If rc = 0 Then Do
  dirtytype = 'DIRC'
  pull .
End
Else dirtytype = 'FILEC'
Parse var dirname '.' rest /* Substitute userid2 into */
newdir = userid2||'.'||rest /* the directory name */
If rest = '' Then Do /* If top directory, skip it */
  'CREATE DIR' newdir '(' dirtytype /* Create the directory */
  If rc = 0 Then Do
    'DROPBUF'
  Exit
End
End
End
End
'DROPBUF'
Return

Transfer_Files:
/*****
/* Routine name - Transfer_Files */
/*
/* Function - For each directory, transfer all of the files from */
/* userid1 to userid2. Call Issue_Grants for each */
/* directory to reestablish authorizations. */
/*
/* For a base file: */
/* it is copied to userid2's directory, Issue_Grants */
/* will be called to re-establish authorizations, and */
/* CreateAliases will be called to re-create any */
/* aliases that existed for that base file. */
/* If the base file had any aliases by other users, */
/* the base file is erased from userid1's directory. */
/*
/* For an alias: */
/* Transfer_Alias will be called to re-create the alias */
/* in userid2's directory. */
/*
/* For an external object: */
/* Information about the external object is obtained */
/* by the Query Object function (DMSQOBJ) and then */
/* the new version is created in userid'2 directory. */
/*
/* Erased and revoked aliases are not transferred. */
*****/

queued = queued()
'MAKEBUF'
'LISTDIR' dirid '(FIFO' /* Get list of directories */
queued = queued() - queued
Do queued
  pull . dirname
  say 'Processing:' dirname
  Address Command 'LISTDIR' dirname '(LIFO DIRC NOSUBDIR'
  If rc = 0 Then Do
    dirtytype = 'DIRC'
    pull .
  End
  Else dirtytype = 'FILEC'
  Parse var dirname '.' rest /* Substitute userid2 into */
  newdir = userid2||'.'||rest /* the directory name */
  'SET CMSTYPE HT'
  'ACCESS' dirname 'B'
  'ACCESS' newdir 'C (FORCERW'
  i = queued()
  'MAKEBUF'
  'LISTFILE * * B (SHARE ALLFILE FIFO /*Get files in the directory*/
  'SET CMSTYPE RT'
  i = queued() - i
  Do i
    Parse pull fn ft . . type .
    If type = 'BASE' Then Do
      Address Command 'COPYFILE' fn ft 'B = C (OLDDATE'
      If dirtytype = 'FILEC' Then
        Call Issue_Grants fn ft B fn ft C
        Call CreateAliases
      End
    If type = 'ALIAS' Then
      Call Transfer_Alias
    If type = 'EXTRNL' Then Do /* External object? */
      fileid = fn ft dirname; fileidlen=length(fileid)
      retcd=0; reason=0; nocommit='NOCOMMIT'; nocommitlen=8

```

## Sample Execs for SFS

```

objname=copies(' ',255); objtype='          '; objlen=255
date='          '; time='          '; crdate='          '
crttime='          '

/* Query the External Object to get the information */

CALL CSL 'DMSQOBJ retcd reason fileid fileidlen',
         'objname nocommit nocommitlen objtype'

/* Recreate it under the new directory */

fileid = fn ft newdir; fileidlen=length(fileid)

CALL CSL 'DMSCROB retcd reason fileid fileidlen',
         'objname objlen nocommit nocommitlen',
         'objtype date time crdate crttime'

End
End
'DROPBUF'
Call Issue_Grants dirname newdir dirtytype
End
'DROPBUF'
'RELEASE B'
'RELEASE C'
Return

Issue_Grants:
/*****
/* Routine name - Issue_Grants */
/* */
/* Function - Re-establishes authorizations on userid2's */
/* directory or file, just as they were on userid1. */
*****/
Parse Arg objects
If WORDS(objects) = 3 Then Do
  Parse Var objects old_object new_object object_type .
  If object_type = 'DIRC' Then dauth = 'DIR'
  Else dauth = 'NEW'
End
Else Do
  Parse Var objects oldfn oldft olddirid newfn newft newdirid .
  old_object = oldfn oldft olddirid
  new_object = newfn newft newdirid
End
j = queued()
'MAKEBUF'
Address Command 'QUERY AUTH' old_object '(FIFO' /* Get authorities*/
pull dirheader /* Toss out Directory header */
pull . /* Toss out issuer's auth */
j = queued() - j
Do j
  If WORDS(old_object) = 1 Then
    pull grantee read write newread newwrite
  Else
    pull . . . . grantee read write .
  If grantee ^= userid1 Then Do /* don't grant to userid1 */
    If grantee = '<PUBLIC>' Then grantee = 'PUBLIC'
    If WORDS(objects) = 6 | object_type = 'FILEC' Then
      If write = 'X' Then
        Address Command 'GRANT AUTH' new_object 'TO' grantee '(WRITE'
      Else If read = 'X' Then
        Address Command 'GRANT AUTH' new_object 'TO' grantee '(READ'
    If WORDS(objects) = 3 Then
      If newwrite = 'X' Then
        Address Command 'GRANT AUTH' new_object 'TO' grantee '(',
          dauth||'WRITE'
      Else If newread = 'X' Then
        Address Command 'GRANT AUTH' new_object 'TO' grantee '(',
          dauth||'READ'

  End
End
'DROPBUF'
Return

Create_Aliases:
/*****
/* Routine name - Create_Aliases */
/* */
/* Function - Re-creates any aliases on a base file that has been */
/* copied over to userid2. */
*****/
j = queued()

```



```

'MAKEBUF'
Address Command 'QUERY ALIAS' fn ft 'B (FIFO' /* get the aliases */
If rc = 0 Then Do
  pull . /* toss out directory header */
  j = queued() - j
  Do j
    Parse pull . . . aliasowner . afn aft adir /* get alias name*/
    If aliasowner = userid1 Then aliasowner = userid2
    Else Do
      Address Command 'ERASE' fn ft dirname
      say 'ERASE' fn ft dirname
    End
    Address Command 'CREATE ALIAS' fn ft 'C' afn aft aliasowner||adir
  End
'DROPBUF'
End
Return

```

```

Transfer_Alias:
/*****
/* Routine name - Transfer_Alias */
/*
/* Function - For an alias in userid1's directory, grant
/* authority on the base file to userid2, and then
/* re-create the alias in userid2's directory.
*****/
'MAKEBUF'
Address Command 'QUERY ALIAS' fn ft 'B (FIFO'/* What is base file? */
pull .
Parse pull . . . baseowner . bfn bft bdir . /* get base file name*/
If baseowner = userid1 Then Return
'DROPBUF'
'MAKEBUF'
Address Command 'QUERY AUTH' fn ft 'B (FIFO'/* What auth userid1? */
pull . /* toss out directory header */
Do Until grantee = userid1 /* Find userid1's authority */
  pull . . . grantee read write
End
/* Don't bother to transfer alias if the base file was owned by
/* userid1, it will be re-created when the base file is copied over*/
If baseowner ^=userid1 Then Do
  If write = 'X' Then Do /* first grant authority */
    Address Command 'GRANT AUTH' bfn bft baseowner||bdir 'TO',
      userid2 '(WRITE'
  End
  Else Do
    Address Command 'GRANT AUTH' bfn bft baseowner||bdir 'TO',
      userid2 '(READ'
  End
  Address Command 'CREATE ALIAS' bfn bft baseowner||bdir fn ft 'C'
End
'DROPBUF'
Return

```

## REGRANT EXEC

```

/*****/
/* COPYRIGHT - */
/* THIS MODULE IS "RESTRICTED MATERIALS OF IBM" */
/* 5684-112 (C) COPYRIGHT IBM CORP. 1990, 1991 */
/* LICENSED MATERIALS - PROPERTY OF IBM */
/* SEE COPYRIGHT INSTRUCTIONS, G120-2083 */
/* */
/* This programming example is to be used as a sample program only. */
/* Although this program may have been reviewed by IBM */
/* for accuracy in a specific environment, there is no */
/* guarantee that the same or similar results will be obtained */
/* elsewhere. The code is being provided on an 'As is' basis */
/* without any warranty expressed or implied. */
/* */
/*****/
/* REGRANT EXEC - This exec will regrant file and directory */
/* authorities for a user who has changed their userid. */
/* For example: if user OLDID changed their userid to NEWID, */
/* all file and directory level grants to OLDID would be */
/* changed to NEWID. All references to OLDID will then be */
/* deleted (revoked). This exec can be processed after a */
/* userid has been transferred using the SFSTRANS EXEC. */
/* */
/* SFS Administrator authority is required to execute */
/* this exec. */
/* */
/* NOTE: The FILEPOOL RENAME command should be used to rename */
/* an SFS file space. This exec is provided as a sample */
/* of how to use SFS functions to perform this type of */
/* operation from a Rexx program. */
/* */
/*****/

Say 'Enter olduserid:'
pull olduserid .
upper olduserid
Say 'Enter newuserid:'
pull newuserid .
upper newuserid
retc = 0
rs = 0
Say 'Enter Filepoolid or hit enter to use current (default) filepool:'
pull filepoolid
If filepoolid = '' then do
  'QUERY FILEPOOL CURRENT (FIFO' /* Get default filepoolid */
  pull . filepoolid .
End
Else do
  If (pos(':',filepoolid)) = 0 Then do
    say 'Invalid filepoolid, colon is required'
  Exit
End
End

/* Get list of all users in the specified filepool. */

'QUERY ENROLL USER FOR ALL' filepoolid '(FIFO'
If rc ->= 0 Then Exit
Parse Var filepoolid filepoolid ':' . /* strip off the colon now. */
pull .
i = queued()
DO i
  Pull userid .
  If userid = '<PUBLIC>' Then Iterate /* Skip userid public... */
  say time()
  say 'Processing grants for' userid
  Call DO_GRANTS
End
Exit

/* */
/* Subroutine DO_GRANTS will issue the GRANT AUTHROTIY and */
/* REVOKE AUTHORITY commands to change the authorities from */
/* the old userid to the new userid. To do this, the SFS */
/* catalogs are read and searched for any occurrence of the */

```

```

/* old userid. When it is found, the appropriate is granted */
/* to the new userid and the authority is revoked from the */
/* old userid. */
/* */

DO_GRANTS:

parm = filepoolid 'FILESPEC' userid
parmlen = Length(parm)
intent = 'READ'
intentlen = length(intent)
sg = 0
buff = ' '
bl = 24

/* Issue SFS Open Catalog function to get server information. */

CALL CSL 'DMSOPCAT retc rs parm parmlen sg intent intentlen buff bl'
If retc > 4 Then Do
  Say 'Open Cat failed for' userid ',rc = ' retc ',reason = ' rs
  Return
End
token = substr(buff,5,8)
buffer = ' '
bufferlen = 296
retc = 0
Do while retc = 0

/* Issue SFS Read Catalog function to get individual catalog */
/* records. Granted userids are included in these records. */
/* The format of these records is contained in the CSL Read */
/* Catalog documentation. Also, some Revokes may fail because */
/* the base file authority was revoked and an alias follows. */
/* This will not cause any problem. */

CALL CSL 'DMSRDCAT retc rs token buffer bufferlen'
cattype = substr(buffer,5,1)
If cattype = 'D' then do /* Dircat record */
  dirname = substr(buffer,6,136)
  dirid = strip(substr(dirname,1,8))||'.'
  dirname = substr(dirname,9)
  i = 1
  Do While (substr(dirname,1) -> ' ') & (i < 9)
    dirid = dirid||strip(substr(dirname,1,16))||'.'
    dirname = substr(dirname,17)
    i = i + 1
  End
End
If cattype = 'O' then do /* Objectcat record */
  fileid = substr(buffer,22,8)||' '||substr(buffer,30,8)
  type = substr(buffer,46,1)
end
If cattype = 'A' then do /* Authcat record */
  grantee = substr(buffer,22,8)
  If substr(buffer,30,1) = 'R' then auth = 'READ'
  If substr(buffer,30,1) = 'W' then auth = 'WRITE'
  If substr(buffer,30,1) = 'X' then auth = 'NEWREAD'
  If substr(buffer,30,1) = 'Y' then auth = 'NEWWRITE'
  If substr(buffer,30,1) = 'A' then auth = 'DIRREAD'
  If substr(buffer,30,1) = 'B' then auth = 'DIRWRITE'
  If grantee = olduserid Then Do
    If type = 'D' Then do
      If substr(fileid,1,1) = '00'X Then Do /* Top directory? */
        say 'GRANT AUTH' dirid 'TO' newuserid '(' auth
        Address Command 'GRANT AUTH' dirid 'TO' newuserid '(' auth
        If rc -> 0 Then say 'GRANT failed on' dirid 'rc = ' rc
      Else Do
        say 'REVOKE AUTH' dirid 'FROM' olduserid
        Address Command 'REVOKE AUTH' dirid 'FROM' olduserid
        If rc -> 0 Then say 'REVOKE failed on' dirid
      End
    End
  Else Do /* SFS Subdirectory */
    Parse Var fileid fn ft .
    fn = strip(fn)
    ft = strip(ft)
    subdir = fn||ft
    say 'GRANT AUTH' dirid||subdir 'TO' newuserid '(' auth
    Address Command 'GRANT AUTH' dirid||subdir 'TO' newuserid,
      '(' auth
    If rc -> 0 Then say 'GRANT failed on' dirid||subdir
  Else Do

```

## Sample Execs for SFS

```
        say 'REVOKE AUTH' dirid||subdir 'FROM' olduserid
        Address Command 'REVOKE AUTH' dirid||subdir 'FROM' olduserid
        If rc != 0 Then say 'REVOKE failed on' dirid||subdir
    End
End
Else Do
    /* SFS file or alias */
    Say 'GRANT AUTH' fileid dirid 'TO' newuserid '(' auth
    Address Command 'GRANT AUTH' fileid dirid 'TO' newuserid '(' auth
    If rc != 0 Then say 'GRANT failed on' fileid dirid ' rc = ' rc
    Else Do
        say 'REVOKE AUTH' fileid dirid 'FROM' olduserid
        Address Command 'REVOKE AUTH' fileid dirid 'FROM' olduserid
        If rc != 0 Then say 'REVOKE failed on' fileid dirid
    End
End
End
End
End

/* Issue SFS Close Catalog function for this user (filespace). */
CALL CSL 'DMSCLCAT retc rs token buff bl'
Return
```

## TALLY EXEC

```

/*****
/* COPYRIGHT - */
/*          5684-112 (C) COPYRIGHT IBM CORP. 1990, 1991 */
/*          LICENSED MATERIALS - PROPERTY OF IBM */
/*          SEE COPYRIGHT INSTRUCTIONS, G120-2083 */
/*          */
/* This programming example is to be used as a sample program only. */
/* Although this program may have been reviewed by IBM */
/* for accuracy in a specific environment, there is no */
/* guarantee that the same or similar results will be obtained */
/* elsewhere. The code is being provided on an 'As is' basis */
/* without any warranty expressed or implied. */
/*          */
/*****
/*          */
/* EXEC NAME: TALLY */
/*          */
/* FUNCTION : TALLY will tabulate the amount of space which */
/*             has been added to a storage group. It will show */
/*             the amount of space being used and also show the */
/*             amount of free space left. TALLY will also calculate */
/*             the difference between the actual physical space and */
/*             the logical allocated space. */
/*          */
/*          */
/* OPERATION: TALLY works in the following way: */
/*             1. Check for '?'. If found give help. */
/*             2. Check what the current file pool id is. */
/*             3. Compare the passed file pool id with the current id. */
/*             4. If no argument was passed set the file pool id to */
/*                the current id. */
/*             5. If the passed id is different than the current id */
/*                do a SET FILEPOOL filepoolid. */
/*             NOTE: If no filepool message is issued from the */
/*                   Q LIMITS command and does not return to exec. */
/*             6. Set up heading for output. */
/*             7. Perform a QUERY FILEPOOL STATUS and stack to */
/*                obtain the physical space used and the amount */
/*                of free space left in a storage group. */
/*             8. Perform a QUERY LIMITS ALL and stack to obtain */
/*                storage group, user, and allocated space data. */
/*             9. Next use calculate the total number of blocks in */
/*                each storage group and then calculate the difference */
/*                between the actual space and the logical */
/*                space allocation. */
/*             10. Set up output. */
/*             11. Establish parameters for output file. The fn being */
/*                 the filepool name, the ft being the current date, */
/*                 and the fm being the A-disk. */
/*             12. See if the 'file' option was specified. If so */
/*                 check to see if a file by that name exists and */
/*                 if it does ask if the user wants it erased. */
/*             13. If the 'file' option was specified then output the */
/*                 report to the A-disk. */
/*             14. If the 'file' option was not specified then output */
/*                 the report to the terminal. */
/*             15. Exit the exec. */
/*          */
/*          */
/*          */
/* SYNTAX   : TALLY filepoolid (option */
/*             option: file /* writes output to the A-disk */
/*          */
/*          */
/*          */
/* DEPENDENCIES */
/*             1. VMSP6 or later. */
/*             2. CMS STATE command */
/*             3. QUERY LIMITS ALL command */
/*             4. QUERY FILEPOOL STATUS command */
/*             5. EXECIO command */
/*             6. CMSTYPE command */
/*          */
/* ERROR CONDITIONS: The normal CMS error messages are output. */
/*          */
/* ADDITIONAL NOTES: TALLY is an administrator function. The */
/*                   proper administrative authority must have */
/*                   been established for this exec to work */
/*          */

```

## Sample Execs for SFS

```

/*          properly.          */
/*          Storage Group 1 is the catalog minidisk and          */
/*          should never contain any allocated space.          */
/*          Tally is dependent upon all storage groups          */
/*          being in sequential order and only 1 minidisk          */
/*          per group.          */
/*          */
/*****

  arg filepool . '(' option          /* check the inputs          */
  if filepool='?' then signal help    /* they want help give it  */

/*-----*/
/* The filepool is checked to provide a default and also to restore */
/* when leaving this exec. If the filepool has not been set to begin */
/* with then the normal message is output and we leave the exec.    */
/* There is no need here to provide an error routine.                */
/*-----*/

  'QUERY FILEPOOL CURRENT (STACK'      /* find current file pool */
  pull . curfilepool .                 /* save the current name  */
  if filepool='' then filepool=curfilepool /* default is current pool */
  if filepool=curfilepool then do      /* check to see if default */
    'SET FILEPOOL' filepool           /* if not set to requested */
    if rc=0 then exit
  end

/*-----*/
/* Set up the heading for the output in a variable.                  */
/*-----*/

  line.=' '                            /* initialize line to null */
  lines=1
  line.lines= CENTER('STATUS REPORT FOR' filepool,72)
  lines=lines+1
  line.lines= CENTER('DATE:' DATE('USA'),72)
  lines=lines+1
  line.lines= CENTER('TIME GENERATED:' TIME(),72)
  lines=lines+3

/*-----*/
/* Get the data needed to calculate the amount of physical 4k blocks */
/* in a storage group. PHYSICAL BLOCKS = IN-USE + FREE 4K blocks.    */
/*-----*/

  'Q FILEPOOL STATUS (STACK'
  if rc=0 then exit(rc)
  numqueued = queued()                /* total number of records */
  do w= 1 to numqueued                 /* build the buffer        */
    pull info.w
  end

/*-----*/
/* Get a count of only those records we need from the buffer. If we */
/* find a null (blank) line we've found the end of our data so leave */
/* the do loop.                                                        */
/* 119 - first line of the storage group totals                        */
/*-----*/

  start=0
  do g = 1 to numqueued
    rc=compare('FILE POOL INFORMATION',info.g, ' ')
    if rc = 0 then leave
    else start = start+1              /* start of information    */
  end
  continue=0
  do h = start to numqueued
    rc=compare('MINIDISK',word(info.h,1), ' ')
    if rc=0 then leave
    else continue=continue+1         /* storage starting point  */
  end
  count=0                             /* initial count to zero   */
  correct=start+continue
  do j = correct to numqueued
    rc=compare('AGENT INFORMATION',info.j, ' ')
    if rc=0 then leave
    else count = count+1             /* storage ending point    */
  end

/*-----*/
/* Calculate the data we want to output from the buffer and          */
/* output it.                                                          */
/* 109 - first line of information we want to output.                  */

```

```

/* group - the last line of storage group information we want output. */
/*-----*/
group=0
group=correct+count-1
do p = start to group
  line.lines=info.p
  lines=lines+1
end

/*-----*/
/* Set up the headings for the physical and allocated block      */
/* information.                                                  */
/*-----*/

lines=lines+2
line.lines='PHYSICAL/ALLOCATED BLOCK INFORMATION'
lines=lines+2
line.lines=' GROUP NUMBER # of USERS  PHYS. 4K BLOCKS  ALLOC.',
          '4K BLOCKS  DIFFERENCE'
lines=lines+1

/*-----*/
/* Get the users per storage group information and the amount    */
/* of space allocated to each.                                  */
/*-----*/

user.=0                /* initialize user. to zero*/
alloc.=0              /* initialize alloc.to zero*/
'Q LIMITS ALL (STACK'
if rc.=0 then exit
records = queued()    /* total records stacked  */
do k = 1 to records
  pull stor.k          /* get a storage group rec */
  storage_num = substr(stor.k,15,5) /* get storage group numb. */
  group_num= strip(storage_num)    /* remove unwanted blanks */
  user.group_num = user.group_num + 1 /* total users in group */
/*-----*/
/* get the total allocated blocks for group.                    */
/*-----*/
  alloc.group_num = alloc.group_num + substr(stor.k,34,6)
end

/*-----*/
/* Set up a routine to search the storage groups so we can add the */
/* physical blocks we calculated to the proper group.              */
/* group - the last line of storage group information we want to  */
/* check. (the minus 1 eliminates the null line.)                 */
/* 119 - first line of the storage group totals                   */
/* index - the group number information we wish to search.        */
/*-----*/

blocks.=0              /* initialize blocks. to 0 */
group_used=''         /* initialize group_used  */
nineteen=correct+1
do l = nineteen to (group-1) /* to a null.            */
  index=word(info.1,2)
  if FIND(group_used,index) = 0 /* have we used this number*/
  then                          /* if we haven't then      */
    groups_used = groups_used index /* store the groups used */
/*-----*/
/* Calculate the total number of physical blocks in the storage group */
/*-----*/

  blocks.index = blocks.index + substr(info.1,38,7),
                + substr(info.1,58,7)
end

/*-----*/
/* For each storage group used calculate the number of users, number */
/* of physical blocks, number of allocated blocks and the difference */
/* between the physical and allocated blocks.                          */
/* Output the results.                                                */
/*-----*/

do h = 1 to words(groups_used)-1
  diff.h=blocks.h-alloc.h
  sgroup=JUSTIFY(h,5)
  nusers=JUSTIFY(user.h,5)
  pblocks=JUSTIFY(blocks.h,9)
  ablocks=JUSTIFY(alloc.h,9)
  diffblocks=JUSTIFY(diff.h,9)

```

```

lines=lines+1
line.lines='      'sgroup'      'nusers'      'pblocks,
'      'ablocks'      'diffblocks

end

/*-----*/
/* Set up the filename, filetype, and filemode for a file which will */
/* be output to disk. */
/*-----*/
filename = strip(filepool,':') /* remove colon in filepool */
fn=filename /* fn is the filepoolid */
parse value date('USA') with mm '/' dd '/' yy /*get date for ft */
ft=yy|mm|dd /* put ft in right format */
fm='A' /* fm is the A-disk */
'SET CMSTYPE HT' /* keep messages from term */
'STATE' fn ft fm /* check for existing file */
saverc=rc /* save the return code */
'SET CMSTYPE RT' /* allow message on term. */

/*-----*/
/* Check the return code to determine if the file exists and check */
/* the option to see if user wants file to go to disk. If the file */
/* exists ask user if they want it erased and the new file placed on */
/* the A-disk. If the file doesn't exist and the disk option was */
/* specified write the status out to disk. Else write the status to */
/* the terminal. */
/*-----*/

if saverc=0 & option='FILE' then do
  say fn ft fm 'already exists. Do you wish to ERASE ? (Y)es or (N)o'
  say 'The default is (Y)es' /* if file exists ask ? */
  pull ans
  if abbrev('YES',ans,0) then do /* if yes,erase existing */
    'ERASE' fn ft fm /* file. */
  end
  else do
    say 'Exiting exec ..... No operations performed .....'
    EXIT /* if no,output message */
  end /* and exit exec. */
end
if option = 'FILE' then 'EXECIO' lines 'DISKW' fn ft fm,
' (FINIS STEM LINE.'
else /* no disk option write to */
do i = 1 to lines /* terminal */
  say line.i
end
'SET FILEPOOL' curfilepool /* re-establish original */
exit /* filepool */
HELP:
/* give them help info */
parse source . . fn ft fm .
say;say
say 'SYNTAX:' fn 'filepoolid ( options'
say '
say ' options: FILE /* writes output to A disk */'
say
exit

```



## WHO EXEC

```

/*****
/* COPYRIGHT - */
/*          5684-112 (C) COPYRIGHT IBM CORP. 1990, 1991 */
/*          LICENSED MATERIALS - PROPERTY OF IBM */
/*          SEE COPYRIGHT INSTRUCTIONS, G120-2083 */
/*          */
/* This programming example is to be used as a sample program only. */
/* Although this program may have been reviewed by IBM */
/* for accuracy in a specific environment, there is no */
/* guarantee that the same or similar results will be obtained */
/* elsewhere. The code is being provided on an 'As is' basis */
/* without any warranty expressed or implied. */
/*          */
/*****
/*          */
/* EXEC NAME: WHO */
/*          */
/* FUNCTION : WHO will sort the storage groups and output what users */
/*            (who) are enrolled in each storage group. */
/*          */
/*          */
/* OPERATION: WHO works in the following way: */
/*          1. Check for '?'. If found give help. */
/*          2. Check what the current file pool id is. */
/*          3. Compare the passed file pool id with the current id. */
/*          4. If no argument was passed set the file pool id to */
/*            the current id. */
/*          5. If the passed id is different than the current id */
/*            do a SET FILEPOOL filepoolid. */
/*          6. Set up heading for output. */
/*          7. Perform a QUERY LIMITS ALL and stack to obtain */
/*            storage group and user data. */
/*          8. Write out the stacked data to a temporary work file */
/*            called 'Storage File', on filemode A. */
/*          9. Sort the records by sorgae group number. */
/*          10. Read the records back in. */
/*          11. Check the storage group number, set up the heading */
/*             and users for the storage group. */
/*          12. Repeat #11 for all the storage groups in the filepool */
/*          13. Establish the parameters for the output file. */
/*          14. See if the 'file' option was specified. If so */
/*             check to see if a file by that name exists and */
/*             if it does ask if the user wants it erased. */
/*          15. If the 'file' option was specified then output the */
/*             report to the A-disk. */
/*          16. If the 'file' option was not specified then output */
/*             the report to the terminal. */
/*          17. Exit the exec. */
/*          */
/*          */
/* SYNTAX   : WHO filepoolid (option */
/*            option: file /* writes output to the A-disk */
/*          */
/*          */
/*          */
/* DEPENDENCIES */
/*          1. VMSP6 or later. */
/*          2. CMS STATE command */
/*          3. QUERY LIMITS ALL command */
/*          4. QUERY FILEPOOL CURRENT command */
/*          5. EXECIO command */
/*          6. CMSTYPE command */
/*          */
/* ERROR CONDITIONS: The normal CMS error messages are output. */
/*          */
/* ADDITIONAL NOTES: WHO is an administrator function. The */
/* proper administrative authority must have */
/* been established for this exec to work */
/* properly. */
/*          */
/*****

arg filepool . '(' option          /* check the inputs */
if filepool='?' then signal help   /* they want help give it */

/*-----*/

```

## Sample Execs for SFS

```

/* The filepool is checked to provide a default and also to restore */
/* when leaving this exec. If the filepool has not been set to begin */
/* with then the normal message is output and we leave the exec. */
/* There is no need here to provide an error routine. */
/*-----*/

'QUERY FILEPOOL CURRENT (STACK'          /* find current file pool */
pull . curfilepool .                    /* save the current name */
if filepool='' then filepool=curfilepool /* default is current pool */
if filepool~=curfilepool then do        /* check to see if default */
'SET FILEPOOL' filepool                 /* if not, set to requested*/
if rc~=0 then exit                       /* exit on error */
end

/*-----*/
/* Set up the heading for the output in a variable. */
/*-----*/

out.=' '                                /* initialize out. to null */
data=1                                  /* initialize data */
out.data= CENTER('STORAGE GROUP REPORT FOR' filepool,72)
data=data+1
out.data= CENTER('DATE:' DATE('USA'),72)
data=data+1
out.data= CENTER('TIME GENERATED:' TIME(),72)
data=data+3

/*-----*/
/* Get the storage group and user data. */
/*-----*/

line.=' '                                /* initialize line. to null*/
lines=1                                  /* initialize lines to 1 */
'Q LIMITS ALL (STACK'                   /* get the data and stack */
if rc ~= 0 Then do                      /* check the return code */
say 'Error' rc 'from QUERY LIMITS ALL Command'
exit(rc)
end
numqueued=queued()                     /* initialize the loop */
do k = 1 to numqueued                   /* get all the data into */
pull info.k                             /* variables. */
line.lines=info.k
lines=lines+1
end

/*-----*/
/* Write out the data and sort. (EXECIO and push commands) */
/*-----*/

'EXECIO' lines 'DISKW' storage file a '(FINIS STEM LINE.'
push 'file'                             /* file file */
push 'SORT * A 15 20'                   /* sort SG number */
'XEDIT storage file a'                  /* XEDIT work file */

/*-----*/
/* Set up to read back in the data. (EXECIO command.) */
/*-----*/

in.=' '                                  /* initialize in. to nulls */
m = 2                                    /* initialize start(m) to 2*/
marker = 2                               /* initialize marker to 2 */
do execiorc ~= 0
'EXECIO' lines 'DISKR' storage file a '(FINIS STEM IN.'
execiorc=rc
finish = lines - 1                       /* initialize finish loop */
do until finish = 0                      /* start loop */
if word(in,m,2) = marker then            /* check for SG match */
do
out.data=' '                             /* set up SG output heading*/
data = data + 1
out.data =CENTER('USERS IN STORAGE GROUP 'word(in,m,2),72)
data = data + 1
out.data = ' '
data = data + 1
out.data='User Storage Group 4K Block Limit',
'4K Blocks Committed Threshold'
data = data + 1
out.data=COPIES('-',72)
data = data + 1
do u = m to in.0 while word(in,u,2) = marker
/* output data for SG */
out.data = in.u

```

```

        data = data + 1
        m = m + 1          /* increment counter */
        finish = finish - 1 /* decrement loop counter */
    end
end
    marker = marker + 1    /* increment SG check */
end
end
'ERASE' STORAGE FILE A    /* erase work file */
/*-----*/
/* Set up the filename, filetype, and filemode for a file which will */
/* be output to disk. */
/*-----*/

filename = strip(filepool,':') /* take off the colon */
fn=filename                 /* fn is the filepoolid */
parse value date('USA') with mm '/' dd '/' yy /*get date for ft */
ft='SG'||yy||mm||dd        /* put ft in right format */
fm='A'                      /* fm is the A-disk */
'SET CMSTYPE HT'           /* keep messages from term*/
'STATE' fn ft fm           /* check for existing file*/
saverc=rc                  /* save the return code */
'SET CMSTYPE RT'          /* allow message on term. */

/*-----*/
/* Check the return code to determine if the file exists and check */
/* the option to see if user wants file to go to disk. If the file */
/* exists ask user if they want it erased and the new file placed on */
/* the A-disk. If the file doesn't exist and the disk option was */
/* specified write the status out to disk. Else write the status to */
/* the terminal. */
/*-----*/

if saverc=0 & option='FILE' then do
    say fn ft fm 'already exists. Do you wish to ERASE ? (Y)es or (N)o'
    say 'The default is (Y)es' /* if file exists ask ? */
    pull ans
    if abbrev('YES',ans,0) then do /* if yes,erase existing */
        'ERASE' fn ft fm /* file. */
    end
    else do
        say 'Exiting exec ..... No operations performed .....'
        EXIT /* if no,output message */
    end /* and exit exec. */
end
if option = 'FILE' then 'EXECIO' data 'DISKW' fn ft fm,
    ' (FINIS STEM OUT.'
else /* no disk option write to*/
do i = 1 to data /* terminal */
    say out.i /* */
end
'SET FILEPOOL' curfilepool /* re-establish original */
exit /* filepool */
HELP:
/* give them help info */
parse source . . fn ft fm .
say;say
say 'SYNTAX:' fn 'filepoolid ( options'
say
say ' options: FILE /* writes output to A disk */'
say
exit

```



## Appendix B. File Pool Server Maximums

This appendix documents some of the maximums of a file pool server. In [Table 35 on page 685](#), many parameters are shown as having no architected limit. While there may not be an architected limit, numerous factors (such as available DASD space) do often impose practical maximums. See [“A Note about File Pool Maximums” on page 28](#) for more about maximums.

Table 35. File Pool Server Maximums

Parameter	Maximum
File pools per z/VM system	no architected limit
Number of IUCV links per virtual machine	65535
Number of storage groups	32767
Number of 4KB blocks per storage group minidisk	99,999,999
Number of storage group minidisks	32767
Number of file spaces per file pool	32767
Number of blocks per file pool	$2^{31}-1$
Number of blocks per file space	$2^{31}-1$
Size of the control minidisk	1,600,000 512-byte blocks
Size of a file pool repository log minidisk	524,200 4KB blocks
Size of a CRR log minidisk	524,200 4KB blocks
Size of a minidisk within a storage group	1 Volume
Size of a storage group	$2^{31}-1$ blocks
Size of a file	$2^{31}-1$ blocks
Number of files or other objects per file pool	no architected limit

### Notes:

1. The current maximum number of storage groups and minidisks is defined by the MAXDISKS value for the file pool. MAXDISKS is originally set during file pool generation, and can be reset using the FILESERV REGENERATE command. Use the QUERY FILEPOOL OVERVIEW command to find the current values for the maximum number of storage groups and minidisks.
2. The number of blocks per file pool is the absolute maximum the file pool server processing can map. Each block is 4096 bytes. The current maximum number of blocks for a file pool is defined by the size of the control minidisk. That is, the size of the control minidisk effectively limits the potential addressable space of the file pool. The size of the control minidisk can be increased using the FILESERV REGENERATE command. Use the QUERY FILEPOOL OVERVIEW command to find the current value for the potential addressable 4KB blocks in the file pool.

Note that not all of the file pool blocks store permanent data. Some of the blocks would be needed to support the "shadow" copies that file pool server processing sometimes creates to support multiple users and to support recovery of updated files.

3. The MAXUSERS file pool generation value effectively limits the logical size of the file pool catalog and thus the number of file pool objects which can be created. MAXUSERS is originally set during file pool generation, and can be reset using the FILESERV REGENERATE command.



---

## Appendix C. File Pool Server Exit Considerations

### PI

This appendix contains a list of general considerations for writing file pool server exits.

**Note:** There are special considerations for writing storage use exits. See [“Special Considerations for Writing Your Own Exit Routine”](#) on page 211.

- Any user-provided exits called by the server should not try to issue any file pool server request or try to establish APPC connections back to the server within the same virtual machine.
- Use of the interval timer in server exits could affect the integrity of the file pool server.
- The following items discuss the use of interrupts in server exits:
  - Disabling of interrupts in server exits is not recommended because it may produce unpredictable results.
  - SSM cannot be used by server exit code because this instruction runs in a BCMODE machine but produces undesirable results in an XA-mode or XC-mode machine.
  - Use of the ENABLE macro is recommended because it provides an architecture-independent way to set the interrupt mask in the PSW.
  - The ENABLE macro does not return the current PSW mask setting.
  - The server will always issue an exit with all interrupts enabled.
  - If absolutely required, the server exit code should use the ENABLE macro to disable interrupts.
  - If interrupts are disabled, the server exit code should ensure all interrupts are enabled with the ENABLE macro before returning to the server.

**Note:** Because of performance considerations, the server **will not** use the ENABLE macro to enable all interrupts on return from an exit.

### PI end





## Appendix D. Mapping SFS Authorization Calls to RACROUTE Requests

### PI

This appendix describes the mapping of SFS calls through the authorization CSL interface to RACROUTE requests and other calls to the External Security Manager (ESM). The IBM-supplied ESM initialization/termination and ESM program check CSL routine (DMSECIT) calls the RPIUCMS module. The IBM-supplied ESM authorization checking CSL routines (DMSOAUTH, DMSAAUTH, and DMSUAUTH) call the RACROUTE macro. For information about coding RPIUCMS and RACROUTE, see *z/VM: Security Server RACROUTE Macro Reference*.

### Authorization Checking Routine Processing

The following sections provide a brief overview of the processing done by each of the IBM-supplied authorization checking CSL routines.

#### SFS Operator Command Authorization Checking Routine

When a user enters an SFS operator command:

1. SFS rebuilds the command to the form shown in [Table 36 on page 689](#). SFS calls the operator command authorization checking CSL routine (identified in the DMSESM PROFILE), passing the operator command in the command field.

*Table 36. Rebuilt SFS Operator Commands*

Rebuilt Command
AUDIT ON PARTIAL
AUDIT ON ALL
AUDIT OFF
AUDIT OFF <i>fn ft</i> (REPLACE
AUDIT OFF CLOSE
AUDIT OFF CLOSE <i>fn ft</i> (REPLACE
AUDIT OFF NOCLOSE
AUDIT CRR ON
AUDIT CRR OFF
AUDIT CRR ONLY
AUDIT CLOSE <i>fn ft</i> (REPLACE
BACKUP
BACKUP TAPE <i>vdev</i>
BACKUP DISK <i>fn ft dirname</i>
BACKUP DISK <i>fn ft fm</i>
CRR ERASE LU <i>luname tpn</i>
CRR ERASE LU <i>luname lognametoken</i> (LNTOKEN

*Table 36. Rebuilt SFS Operator Commands (continued)*

<b>Rebuilt Command</b>
CRR ERASE LUWID <i>token</i>
CRR QUERY LOG
CRR QUERY LOGTABLE <i>luname tpn</i> (BEFORE <i>mm/dd/yy</i> )
CRR QUERY LOGTABLE ALL (BEFORE <i>mm/dd/yy</i> )
CRR QUERY LU <i>luname</i> (SYNCPT)
CRR QUERY LU <i>luname</i> (RESYNC)
CRR QUERY LU <i>luname</i> (PENDING)
CRR QUERY LU ALL (SYNCPT)
CRR QUERY LU ALL (RESYNC)
CRR QUERY LU ALL (PENDING)
CRR QUERY LUWID <i>token</i> (ALL)
CRR RESUME <i>token</i>
CRR RESYNC <i>token index</i> COMMIT NOPURGE
CRR RESYNC <i>token index</i> COMMIT PURGE
CRR RESYNC <i>token index</i> BACKOUT NOPURGE
CRR RESYNC <i>token index</i> BACKOUT PURGE
CRR SUSPEND <i>token</i>
DEFBACKUP TAPE <i>vdev</i>
DEFBACKUP DISK <i>fn ft dirname</i>
DEFBACKUP DISK <i>fn ft fm</i>
DISABLE GROUP <i>num</i> SHARE NODETACH FOR <i>owner</i>
DISABLE GROUP <i>num</i> EXCLUSIVE NODETACH FOR <i>owner</i>
DISABLE GROUP <i>num</i> EXCLUSIVE DETACH FOR <i>owner</i>
DISABLE FILESPACE <i>userid</i> EXCLUSIVE FOR <i>owner</i>
DISABLE FILESPACE <i>userid</i> SHARE FOR <i>owner</i>
ENABLE GROUP <i>num</i> FOR <i>owner</i>
ENABLE FILESPACE <i>userid</i>
ENABLE FILESPACE <i>userid</i> FOR <i>owner</i>
ENABLE FILESPACE <i>userid</i> FUNCTION <i>function-name</i>
ERASE LUNAME <i>luname</i>
ERASE LUNAME <i>luname tpn</i> (ALL)
ERASE LUNAME <i>luname tpn</i> (LOGTABLE)
ERASE LUNAME <i>luname tpn</i> (FORCED)
ETRACE ON

Table 36. Rebuilt SFS Operator Commands (continued)

**Rebuilt Command**

ETTRACE OFF

FORCE USER *userid*FORCE USER *userid* ALLFORCE PREPARED *taskid* COMMITFORCE PREPARED *taskid* BACKOUT

ITRACE ON

ITRACE ON *buffersize*

ITRACE OFF

QUERY DEFBACKUP

QUERY DEFBACKUP DEFAULT

QUERY DEFBACKUP CURRENT

QUERY DISABLE GROUP *group-number*QUERY DISABLE FILESPACE *userid*QUERY LOGTABLE *luname*QUERY LOGTABLE *luname tpn*

QUERY LOGTABLE ALL

QUERY PREPARED FOR *userid*

QUERY PREPARED FOR ALL

STOP

STOP BACKUP

STOP BACKUP TAPE *vdev*STOP BACKUP DISK *fn ft dirname*STOP BACKUP DISK *fn ft fm*

STOP NOBACKUP

STOP IMMEDIATE

2. The IBM-supplied routine, DMSOAUTH:
- a. Tokenizes the command
  - b. Prefixes the file pool name
  - c. Compresses blanks in the resulting token string
  - d. Separates each token in the string with a period

For example, the SFS operator command

```
AUDIT ON ALL
```

becomes

```
filepool.AUDIT.ON.ALL
```

3. DMSOAUTH then calls the RACROUTE REQUEST=AUTH macro, passing the command token string in the ENTITYX field.

## SFS Administrator Command Authorization Checking Routine

When a user enters an SFS administrator command (or uses a program that calls a corresponding CSL routine):

1. SFS rebuilds the command or CSL routine to the form shown in [Table 37 on page 692](#). If both a command and a CSL routine exist, the command form is used. If no command exists, the CSL form is used. If neither a command nor a CSL routine exist, a unique form is created. The table also shows the file pool request code for the operation.

- a. The \*LIST\* token in the following strings indicates a list of user IDs is associated with the command.

The IBM-supplied routine, DMSAAUTH, builds the list of user IDs into the token string. For example, a DELETE USER command for deleting 5 users might contain the following value specified in the ENTITYX keyword:

```
filepool.DELETE.USER.USER1.USER2.USER3.USER4.USER5
```

If the list of users will not fit within a string of 255 characters, multiple RACROUTE calls will be made, each specifying a different set of user IDs. For example,

```
:
filepool.DELETE.USER.USER201.USER202.USER203
filepool.DELETE.USER.USER204.USER205.USER206
filepool.DELETE.USER.USER207.USER208.USER209
```

- b. The QUERY FILEPOOL STATUS (CATALOG, QUERY FILEPOOL REPORT (CATALOG, and QUERY FILEPOOL CATALOG commands will all be rebuilt as QUERY FILEPOOL CATALOG.
- c. Some of the commands are both *administrator* and *regular* SFS user commands. The *regular* user output will be sent to the user if the ESM rejects this command check.

Table 37. Rebuilt SFS Command and CSL Routine Strings

Request Code	Rebuilt Command or CSL Routine String (See note)
X'16'	BFS DMSCRLOC
X'23'	BFS DMSERASE
X'15'	BFS DMSEXIST
X'18'	BFS DMSOPDIR
X'17'	BFS DMSOPEN
X'20'	CONNECT
X'65'	CONNECT USER <i>userid</i>
X'46'	DATASPACE ASSIGN <i>dirid</i>
X'46'	DATASPACE RELEASE <i>dirid</i>
X'1C'	DELETE LOCK <i>fn ft dirid</i> ( FROM <i>userid</i>
X'3C'	DELETE PUBLIC
X'3C'	DELETE USER *LIST*
X'47'	DIRATTR <i>dirid</i> DIRCONTROL ( FORCE NOFORCE
X'47'	DIRATTR <i>dirid</i> FILECONTROL

Table 37. Rebuilt SFS Command and CSL Routine Strings (continued)

<b>Request Code</b>	<b>Rebuilt Command or CSL Routine String (See note)</b>
X'22'	DMSCRALI UNRESOLVED
X'16'	DMSDISFS <i>userid</i> SHARE EXCLUSIVE
X'16'	DMSDISSG <i>nnnnn</i> SHARE EXCLUSIVE DETACH NODETACH
X'1C'	DMSENAFS <i>userid</i>
X'1C'	DMSENAFS <i>userid</i> FUNCTION <i>funct_name</i>
X'1C'	DMSENASG <i>nnnnn</i>
X'17'	DMSOPBLK <i>fn ft dirid</i> CREATMIG
X'42'	DMSOPCAT DIRECTORY <i>dirid</i> 0 READ FILEATTR
X'42'	DMSOPCAT FILESPACE <i>userid</i> 0 READ WRITE FILEATTR
X'42'	DMSOPCAT GROUP <i>nnnnn</i> READ WRITE FILEATTR READEXT
X'63'	DMSQUSG <i>nnnnn</i>
X'45'	DMSRELBK <i>nnnnn</i>
X'30'	DMSWRACC
X'34'	ENROLL PUBLIC
X'34'	ENROLL USER *LIST* ( BLOCKS <i>blks</i> STORGROUP <i>storgrp</i>
X'67'	FILEPOOL CONTROL BACKUP
X'67'	FILEPOOL CONTROL BACKUP DISK <i>fn ft dirid</i>
X'67'	FILEPOOL CONTROL BACKUP TAPE <i>vdev</i>
X'66'	FILEPOOL MINIDISK
X'68'	FILEPOOL RENAME <i>userid1 userid2</i>
X'30'	MODIFY USER + <i>blocks</i> FOR *LIST*
X'30'	MODIFY USER - <i>blocks</i> FOR *LIST*
X'48'	QUERY ACCESSORS <i>dirid</i>
X'48'	QUERY ACCESSORS <i>dirid</i> ( DATASPACE
X'48'	QUERY ACCESSORS <i>filepoolid</i> :
X'48'	QUERY ACCESSORS <i>filepoolid</i> : ( DATASPACE
X'49'	QUERY DATASPACE <i>filepoolid</i> :
X'49'	QUERY DATASPACE <i>dirid</i>
X'39'	QUERY FILEPOOL CATALOG
X'38'	QUERY FILEPOOL CONFLICT <i>userid</i>
X'0F'	QUERY FILEPOOL DISABLE ALL
X'0F'	QUERY FILEPOOL DISABLE FILESPACE FOR <i>userid</i>
X'0F'	QUERY FILEPOOL DISABLE GROUP <i>nnnn</i>
X'3A'	QUERY LIMITS ALL

Table 37. Rebuilt SFS Command and CSL Routine Strings (continued)

Request Code	Rebuilt Command or CSL Routine String (See note)
X'3A'	QUERY LIMITS FOR <i>userid</i>
X'27'	RELOCATE <i>dirid1</i> TO <i>dirid2</i>
X'27'	RELOCATE <i>fn ft dirid1</i> TO <i>dirid2</i>
X'28'	RENAME <i>dirid1</i> <i>dirid2</i>
X'31'	SET THRESHOLD <i>nn</i> (FOR <i>userid</i> )
X'60'	SMSCDRA <i>fn ft dirid dra1 dra2 dra3</i>
X'23'	SMSERASE <i>fn ft dirid</i> ACFONLY
X'17'	SMSOPEN <i>fn ft dirid</i> WRITE NEW REPLACE ACF
X'17'	SMSOPENX <i>fn ft dirid</i> MIGRATE RECALL

**Note:**

In rebuilding the command or routine string, SFS substitutes a period for any null parameter value.

2. SFS calls the command authorization checking CSL routine (identified in the DMSESM PROFILE), passing the rebuilt command or routine string in the command field.
3. The IBM-supplied routine, DMSAAUTH:
  - a. Tokenizes the command or routine
  - b. Prefixes the file pool name
  - c. Compresses blanks in the resulting token string
  - d. Separates each token in the string with a period

For example, the SFS command

```
QUERY FILEPOOL REPORT (CATALOG
```

becomes

```
filepool.QUERY.FILEPOOL.CATALOG
```

4. DMSAAUTH then calls the RACROUTE REQUEST=AUTH macro, passing the command/routine token string in the ENTITYX field.

## SFS Object Authorization Routine

Generally, transactions on SFS objects map to more than one ESM request. CMS may make several requests, as may the SFS server. For example, an access of an SFS directory results in a REFRESH DIRECTORY request from CMS to the SFS server. Then the server is obliged to make several requests of the ESM:

- A request for access to the directory
- A query as to the number of generic profiles, if any
- A request to cover any files or directories
- A query as to whether each file or directory is protected by the ESM

Each file, external object, directory, and alias name in the object list contains all the parent directory names, not merely the file name and file type of a file.

Notice that some object transactions do not have SFS requests that map directly to them. For example, to call DIRLIST when the directory is not accessed results in the directory being refreshed. Notice, too, some parameters on the object authorization checking CSL routine (the IBM-supplied routine is DMSUAUTH) have no equivalent function in RACROUTE and are not passed. However, these parameters are provided for other ESMs that might use them.

## User Data Structure

For every authorization call from SFS for a non-operator CSL routine (SFS command check) or an SFS object, the IBM-supplied authorization checking CSL routine (DMSAAUTH or DMSUAUTH, respectively) builds a user data structure from information that SFS supplies in parameter 4 (file pool request code), parameter 12 (user data flags), and parameter 13 (user data string). Table 38 on page 695 describes the format of the user data structure. ESM user data can be passed only in certain CSL routines, listed in Table 20 on page 190. However, the authorization checking routine builds the user data structure for all command and object requests, whether the SFS server passes any user data on the call. The authorization routine then passes the address of the control block containing the user data structure to the ESM in the USERWRD parameter on the RACROUTE call.

Table 38. Format of the ESM User Data Structure

Hex Offset	Name	Field Type	Description
00	Request_code	Integer	Identifies the file pool request the server is performing for the user.
04	UD_flag	Char(1)	Identifies whether user data is being passed on this call. The server sets this field to 1 if there is data in the UD_content area. If the application did not supply the <i>userdata</i> on the CSL routine that caused the server function to be processed, the server sets this field to 0. (In this case, any data in UD_content should be ignored.)
05	UD_processed	Char(1)	The server uses this field to determine whether your ESM processed the user data. Prior to calling the ESM, the server sets this flag to 0. If your ESM processes user data, you should set this flag to 1 before returning to the server.
06	Connection_Flag	Char(1)	When Connection_Flag is 1, it indicates there is more information following the UD_content field.
07		Char(1)	Reserved for IBM use.
08	UD_length	Integer	Length of the following UD_content field. The UD_content field can be up to 80 bytes long.
0C to n	UD_content	Varying Character	When UD_flag is 1, UD_content contains the ESM user data. The user specifies this data in the <i>userdata</i> parameter of various CSL routines. When UD_flag is 0, UD_content should be ignored.
n + 4	Connection_type	(Fixed 32)	User's Connection_type field indicates whether the connection is local or remote. If Connection_type is set to 0, the connection is local. If Connection_type is set to 1, the connection is remote.
n + 8	Path ID_number	(Fixed 32)	Identifies the user's APPC path ID number.
n + C	Unique_token	(Fixed 32)	Identifies the SFS unique token for this user's connection.

Table 38. Format of the ESM User Data Structure (continued)

Hex Offset	Name	Field Type	Description
n + D	Valid_flag	(Char 1)	Identifies whether information in the following three fields is valid. The server sets this field to 0 if the CPEVPLKL, CPEVRLUL, and CPEVRLUN information in the following fields is invalid (unavailable). The server sets this field to 1 if the information is valid.
n + E		(Char 3)	Reserved for IBM use.
n + 10	CPEVPLKL information	(Char 16)	Identifies the user's CPEVPLKL information (from an APPCVM Connect). See <a href="#">z/VM: CP Programming Services</a> for details.
n + 20	CPEVRLUL information	(Fixed 32)	Length of the following field, CPEVRLUN. See <a href="#">z/VM: CP Programming Services</a> for details.
n + 24	CPEVRLUN information	(Char *)	Identifies the user's CPEVRLUN information (from an APPCVM Connect). See <a href="#">z/VM: CP Programming Services</a> for details.

If the Request\_code is for a CSL routine that allows ESM user data, your ESM should check the UD\_flag next. This flag indicates whether the user specified the *userdata* parameter on the CSL routine.

If the UD\_flag is 0, the user omitted the *userdata* parameter. This may or may not be acceptable to your ESM, depending on the support it is providing. If the UD\_flag is 1, the user specified *userdata*. Your ESM can process the data in UD\_content.

If your ESM accepts user data, it must process the user data and set UD\_processed to 1 before returning the authorization call. On the return from the ESM, the authorization routine copies the UD\_processed value as the second flag of parameter 12 before returning to SFS. If no user data was supplied on the CSL routine (UD\_flag set to 0), it does not matter if UD\_processed is set to 0. In fact, this is always the case for ESMs that do not support user data.

## RACROUTE Request Formats

The following parameters are common to all RACROUTE requests (except REQUEST=VERIFYX, which does not use ECB1 and ECB2):

```
RACROUTE REQUEST=AUTH (or REQUEST=DEFINE)
,RELEASE=1.9
,ENTITYX=address of file/external object/alias/directory/command
name
,USERWRD=address of user data structure
,USERID=address of requesting user ID
,WORKA=address of 512-byte work area
,ECB1=address of event control block 1
,ECB2=address of event control block 2
```

If SFS sets the new security token information indicator to Y on the CSL call and passes an ESM security token and an ESM information buffer address (see [exit routine parameters 20, 21, and 22](#)), the authorization checking routine calls the following RACROUTE call to generate new ESM information for the requesting user ID and security token:

```
RACROUTE REQUEST=VERIFYX
,RELEASE=1.9
,ENTITYX=address of file/external object/alias/directory/command
name
,USERWRD=address of user data structure
,USERID=address of requesting user ID
,WORKA=address of 512-byte work area
,SECLABEL=security token
,TOKNOUT=address of ESM information buffer
```



After RACROUTE REQUEST=VERIFYX successfully fills in the ESM information buffer with the generated information, it resets the new security token information indicator to N.

The ESM information buffer is then passed as an additional parameter (UTOKEN) on subsequent RACROUTE REQUEST=AUTH calls:

```
RACROUTE REQUEST=AUTH
      ,RELEASE=1.9
      ,ENTITYX=address of file/external object/alias/directory/command
name
      ,USERWRD=address of user data structure
      ,USERID=address of requesting user ID
      ,WORKA=address of 512-byte work area
      ,ECB1=address of event control block 1
      ,ECB2=address of event control block 2
      ,UTOKEN=address of ESM information buffer
```

## Mapping Tables

The tables in the following sections illustrate the mapping of SFS calls through the CSL interface to the ESM. For brevity and clarity on the RACROUTE calls, the tables omit the common parameters, such as release level, and show only those fields unique to a given call.

### ESM Initialization and Termination

To perform ESM initialization processing, the SFS server calls the ESM initialization/termination CSL routine specified in the DMSESM PROFILE. The IBM-supplied DMSSECIT routine calls a CMSCALL to the RPIUCMS module.

Table 39. Mapping for ESM Initialization

DMSSECIT Parameters	ESM Call
<ul style="list-style-type: none"> <li>Server user ID</li> <li>File pool ID (VM APPC resource name)</li> <li>Maximum number of connected users</li> <li>Connections used by ESM</li> <li>Maximum number of active users</li> <li>Command keyword INIT</li> <li>Identity field the ESM can fill in with its name.</li> <li>Flag the ESM can set to 1 if it protects all resources and never defers decisions to SFS.</li> <li>Global storage field the ESM can fill in with an address or value that SFS will keep and pass to the ESM on subsequent calls.</li> </ul>	<ul style="list-style-type: none"> <li>CMSCALL with parameter list:           <pre>RPIUCMS INIT X'FFFFFFFFFFFFFFFF'</pre> </li> </ul>

To perform ESM termination processing, the SFS server calls the ESM initialization/termination CSL routine specified in the DMSESM PROFILE. The IBM-supplied DMSSECIT routine calls a CMSCALL to the RPIUCMS module.

Table 40. Mapping for ESM Termination

DMSSECIT Parameters	ESM Call
<ul style="list-style-type: none"> <li>Server user ID</li> <li>Command keyword TERM</li> </ul>	<ul style="list-style-type: none"> <li>CMSCALL with parameter list:           <pre>RPIUCMS TERM X'FFFFFFFFFFFFFFFF'</pre> </li> </ul>

## ESM Program Check

To inform the ESM a program check occurred, the SFS server calls the ESM program check CSL routine specified in the DMSESM PROFILE. The IBM-supplied DMSSECIT routine calls a CMSCALL to the RPIUCMS module.

Table 41. Mapping for ESM Program Check

DMSSECIT Parameters	ESM Call
<ul style="list-style-type: none"> <li>Command keyword TERM</li> </ul>	<ul style="list-style-type: none"> <li>CMSCALL with parameter list:                     <pre>RPIUCMS TERM X'FFFFFFFFFFFFFFFF'</pre> </li> </ul>

## SFS Operator Commands

To check a user's authority for an SFS operator command, the SFS server calls the SFS operator command authorization checking CSL routine. The IBM-supplied DMSOAUTH routine calls a single RACROUTE call to the ESM, as shown in Table 42 on page 698.

Table 42. RACROUTE Mapping for SFS Operator Commands

DMSOAUTH Parameters	RACROUTE Call
<ul style="list-style-type: none"> <li>Server user ID</li> <li>Special file pool request code (see Table 14 on page 168)</li> <li>File pool ID</li> <li>Command string</li> </ul>	<ul style="list-style-type: none"> <li>Authorization call for CMS command:                     <pre>RACROUTE REQUEST=AUTH ,ATTR=UPDATE ,CLASS=SFSCMD</pre> </li> </ul>

## SFS Non-Operator Commands and CSL Routines

To check a user's authority for an SFS non-operator command or CSL routine, the SFS server calls the SFS command authorization checking CSL routine. The IBM-supplied DMSAAUTH routine calls a single RACROUTE call to the ESM, as shown in Table 43 on page 698.

Table 43. RACROUTE Mapping for SFS Non-Operator Commands and CSL Routines

DMSAAUTH Parameters	RACROUTE Call
<ul style="list-style-type: none"> <li>User ID</li> <li>File pool request code (see Table 37 on page 692)</li> <li>Processing code</li> <li>Transaction ID</li> <li>File pool ID</li> <li>Rebuilt command string (see Table 37 on page 692)</li> <li>User data and flags (optional on certain CSL routines; see Table 20 on page 190)</li> <li>User ID list (if group name specified)</li> </ul>	<ul style="list-style-type: none"> <li>Authorization call for CMS command:                     <pre>RACROUTE REQUEST=AUTH ,ATTR=UPDATE ,CLASS=SFSCMD</pre> </li> </ul>

## SFS Objects

The following tables show the mapping for requests requiring authority for an SFS object.

**Note:** Because the RACROUTE interface does not have a CLASS=EO (External Object) or CLASS=ALIAS, all external objects and aliases will be mapped to CLASS=FILE.

To assist the security administrator with reviewing and auditing, the LOGSTR='text' option is used on the RACROUTE macros for SFS objects. The text consists of 'SFS' followed by the name of the file pool request code requiring the authority for the SFS object. If the object is an alias or external object, the string will also contain 'Alias' or 'External Object'. The LOGSTR string contents and order may change if other new information is to be added. It is not intended to be a programming interface.

### COPYFILE NEWFILE/REPLACE

To check a user's authority to copy a file, the SFS server calls the SFS object authorization checking CSL routine. A new file is indicated by FN in the object description in the object list on the authorization call. The IBM-supplied DMSUAUTH routine calls a series of RACROUTE calls to the ESM.

Table 44. Mapping for COPYFILE Command

DMSUAUTH Parameters	RACROUTE Calls
<ul style="list-style-type: none"> <li>• User ID</li> <li>• File pool request code: X'25'</li> <li>• Processing code</li> <li>• Transaction ID</li> <li>• File pool ID</li> <li>• Name of file to be copied</li> <li>• Name of target file (existing) or directory (if new file)</li> <li>• New file name (if requested in DMSESM PROFILE file)</li> <li>• User data and flags (optional)</li> </ul>	<p>Separate calls for:</p> <ul style="list-style-type: none"> <li>• READ access to source file</li> <li>• WRITE access to target file (existing) or directory (if new file)</li> <li>• ALTER access to new file (if requested)</li> </ul>

### CREATE ALIAS

To check a user's authority to create an alias, the SFS server calls the SFS object authorization checking CSL routine. The IBM-supplied DMSUAUTH routine calls a series of RACROUTE calls to the ESM.

Table 45. Mapping for CREATE ALIAS Command

DMSUAUTH Parameters	RACROUTE Calls
<ul style="list-style-type: none"> <li>• User ID</li> <li>• File pool request code: X'22'</li> <li>• Processing code</li> <li>• Transaction ID</li> <li>• File pool ID</li> <li>• Name of base file</li> <li>• Name of target directory</li> <li>• Alias name (if DMSESM PROFILE file requested alias information)</li> <li>• User data and flags (optional)</li> </ul>	<p>Separate calls for:</p> <ul style="list-style-type: none"> <li>• READ access to base file</li> <li>• READ access to base file for the owner of the directory in which the alias is being created</li> <li>• WRITE access to directory</li> <li>• ALTER access to alias name (if requested)</li> </ul>

## **CREATE DIRECTORY**

To check a user's authority to create a directory, the SFS server calls the SFS object authorization checking CSL routine. The IBM-supplied DMSUAUTH routine calls a series of RACROUTE calls to the ESM.

*Table 46. Mapping for CREATE DIRECTORY Command*

<b>DMSUAUTH Parameters</b>	<b>RACROUTE Calls</b>
<ul style="list-style-type: none"> <li>• User ID</li> <li>• File pool request code: X'21'</li> <li>• Processing code</li> <li>• Transaction ID</li> <li>• File pool ID</li> <li>• Name of parent directory</li> <li>• Name of new directory (if requested in DMSESM PROFILE file)</li> <li>• User data and flags (optional)</li> </ul>	<p>Separate calls for:</p> <ul style="list-style-type: none"> <li>• WRITE access to parent directory</li> <li>• ALTER access to new directory (if requested)</li> </ul>

## **DMSCROB (Create External Object)**

To check a user's authority to create a new external object, the SFS server calls the SFS object authorization checking CSL routine. A new external object is indicated by EN in the object description in the object list on the authorization call. The IBM-supplied DMSUAUTH routine calls a series of RACROUTE calls to the ESM.

*Table 47. Mapping for DMSCROB CSL Routine*

<b>DMSUAUTH Parameters</b>	<b>RACROUTE Calls</b>
<ul style="list-style-type: none"> <li>• User ID</li> <li>• File pool request code: X'61'</li> <li>• Processing code</li> <li>• Transaction ID</li> <li>• File pool ID</li> <li>• Name of directory</li> <li>• New external object name (if new file names are requested in DMSESM PROFILE file)</li> <li>• User data and flags (optional)</li> </ul>	<p>Separate calls for:</p> <ul style="list-style-type: none"> <li>• WRITE access to directory</li> <li>• ALTER access to new file (External Object) (If requested.)</li> </ul>

## **CREATE LOCK**

To check a user's authority to create a lock, the SFS server calls the SFS object authorization checking CSL routine. The IBM-supplied DMSUAUTH routine calls a single RACROUTE call to the ESM.

Table 48. Mapping for CREATE LOCK Command

DMSUAUTH Parameters	RACROUTE Call
<ul style="list-style-type: none"> <li>• User ID</li> <li>• File pool request code: X'16'</li> <li>• Processing code</li> <li>• Transaction ID</li> <li>• File pool ID</li> <li>• Name of directory or file to be locked</li> <li>• User data and flags (optional)</li> </ul>	<ul style="list-style-type: none"> <li>• Authority to directory or file to be locked (READ/WRITE; READ is for share, WRITE is for exclusive.)</li> </ul>

## DIRLIST/LISTDIR

Checking a user's authority to call the requests generated by the DIRLIST or LISTDIR command results in a series of calls to the ESM. The open directory (X'18') request causes the SFS server to call do-you-know and what-authority queries. See [“Do You Know?” Query](#) on page 702 and [“What Authority?” Query](#) on page 705.

## DISCARD/ERASE

To check a user's authority to discard a file/directory/external object, the SFS server calls the SFS object authorization checking CSL routine. The IBM-supplied DMSUAUTH routine calls a series of RACROUTE calls to the ESM.

Table 49. Mapping for DISCARD and ERASE Commands

DMSUAUTH Parameters	RACROUTE Calls
<ul style="list-style-type: none"> <li>• User ID</li> <li>• File pool request code: X'23' for file, external object, or alias X'24' for directory</li> <li>• Processing code</li> <li>• Transaction ID</li> <li>• File pool ID</li> <li>• Name of parent directory</li> <li>• Name of file, external object, or directory (base file if alias and the user ID does not match user ID of top directory, unless this is an SFS revoked or dropped alias)</li> <li>• Name of alias (if alias requested in DMSESM PROFILE file)</li> <li>• User data and flags (optional)</li> </ul>	<p>Separate calls for:</p> <ul style="list-style-type: none"> <li>• WRITE access to parent directory</li> <li>• ALTER access to file, external object, directory, or alias (if requested)</li> <li>• READ access to base file (if specified)</li> </ul>

*Table 50. Mapping for DISCARD and ERASE Commands (Commit or Rollback)*

<b>DMSUAUTH Parameters</b>	<b>RACROUTE Calls</b>
<ul style="list-style-type: none"> <li>• File pool request code:                             <ul style="list-style-type: none"> <li>X'23' for file, external object, or alias</li> <li>X'24' for directory</li> </ul> </li> <li>• Processing code</li> <li>• Transaction ID</li> <li>• File pool ID</li> <li>• Name of file or directory</li> <li>• User data and flags (optional)</li> </ul>	Separate calls for: <ul style="list-style-type: none"> <li>• Delete discrete profile if request is committed</li> <li>• RACROUTE VERIFY</li> </ul>

### **DMSOPEN (New or Existing File)**

To check a user's authority to create a new file or open an existing file, the SFS server calls the SFS object authorization checking CSL routine. A new file is indicated by FN in the object description in the object list on the authorization call. The IBM-supplied DMSUAUTH routine calls a series of RACROUTE calls to the ESM.

*Table 51. Mapping for DMSOPEN CSL Routine*

<b>DMSUAUTH Parameters</b>	<b>RACROUTE Calls</b>
<ul style="list-style-type: none"> <li>• User ID</li> <li>• File pool request code: X'17'</li> <li>• Processing code</li> <li>• Transaction ID</li> <li>• File pool ID</li> <li>• Name of file (for existing) or name of directory (if new file)</li> <li>• New file name (if requested in DMSESM PROFILE file)</li> <li>• User data and flags (optional)</li> </ul>	Separate calls for: <ul style="list-style-type: none"> <li>• READ/WRITE access to file (existing)</li> <li>• WRITE access to directory (if new file)</li> <li>• ALTER access to new file (if requested)</li> </ul>

### **'Do You Know?' Query**

To obtain information about whether an SFS object is known to the ESM, the SFS server calls the SFS object authorization checking CSL routine. The IBM-supplied DMSUAUTH routine calls a single RACROUTE call to the ESM. This type of checking is usually done to update the response data to CMS with the R/W/P flags for the file/external object or directory.

Table 52. Mapping for 'Do You Know?' Query

DMSUAUTH Parameters	RACROUTE Call
<ul style="list-style-type: none"> <li>User ID</li> <li>File pool request code:                             <ul style="list-style-type: none"> <li>X'11' for CLOSE</li> <li>X'15' for GET DIRECTORY ENTRY</li> <li>X'18' for OPEN DIRECTORY</li> <li>X'1A' for REFRESH DIRECTORY</li> <li>X'22' for CREATE ALIAS</li> <li>X'25' for COPYFILE</li> <li>X'26' for GRANT AUTHORITY</li> <li>X'28' for RENAME</li> <li>X'29' for REVOKE AUTHORITY</li> <li>X'61' for DMSCROB</li> </ul> </li> <li>Processing code</li> <li>Transaction ID</li> <li>File pool ID</li> <li>Name of file, external object, directory, or alias</li> <li>User data and flags (optional)</li> </ul>	<ul style="list-style-type: none"> <li>"Do you know?" query (READ check for file, external object, directory, or alias)</li> </ul> <pre>RACROUTE REQUEST=AUTH ,ATTR=READ ,CLASS=FILE/DIRECTRY ,LOG=NOFAIL</pre>

### Generic Profile Query

To check for the existence of a generic profile for the files, external objects and subdirectories in a directory, the SFS server calls the SFS object authorization checking CSL routine. External objects are covered under the file object test. The IBM-supplied DMSUAUTH routine calls a single RACROUTE call to the ESM.

Table 53. Mapping for Generic Profile Query

DMSUAUTH Parameters	RACROUTE Call
<ul style="list-style-type: none"> <li>User ID</li> <li>File pool request code:                             <ul style="list-style-type: none"> <li>X'14' for GET DIRECTORY</li> <li>X'1A' for REFRESH DIRECTORY</li> <li>X'27' for RELOCATE</li> <li>X'28' for RENAME</li> </ul> </li> <li>Processing code</li> <li>Transaction ID</li> <li>File pool ID</li> <li>Name of directory</li> <li>Object types of either file or directory</li> <li>User data and flags (optional)</li> </ul>	<ul style="list-style-type: none"> <li>Generic profile query:</li> </ul> <pre>RACROUTE REQUEST=AUTH ,ATTR=READ ,LOG=NOFAIL ,CLASS=FILE/DIRECTRY ,ENTITYX='dirname.**'       (for CLASS=DIRECTRY) or='dirname.**.*'       (for CLASS=FILE)</pre>

### Miscellaneous Authority Check

To check whether a user has read or write authority on a file, external object, or directory, the SFS server calls the SFS object authorization checking CSL routine. The IBM-supplied DMSUAUTH routine calls a single RACROUTE call to the ESM.

*Table 54. Mapping for Miscellaneous Authority Check*

DMSUAUTH Parameters	RACROUTE Call
<ul style="list-style-type: none"> <li>User ID</li> <li>File pool request code:                             <ul style="list-style-type: none"> <li>X'14' for GET DIRECTORY</li> <li>X'1A' for REFRESH DIRECTORY</li> <li>X'1C' for UNLOCK</li> <li>X'23' for CREATE ALIAS</li> <li>X'2C' for DMSCATTR</li> <li>X'41' for CLOSE CATALOG</li> </ul> </li> <li>Processing code</li> <li>Transaction ID</li> <li>File pool ID</li> <li>Name of file, external object, or directory</li> <li>User data and flags (optional)</li> </ul>	<ul style="list-style-type: none"> <li>Miscellaneous authority check (READ or WRITE check for file, external object, or directory):</li> </ul> <pre>RACROUTE REQUEST=AUTH           ,ATTR=READ/UPDATE           ,CLASS=FILE/DIRECTRY</pre>

### RELOCATE (File, Alias, External Object, or Directory)

To check a user's authority to move a file or directory, the SFS server calls the SFS object authorization checking CSL routine. The IBM-supplied DMSUAUTH routine calls a series of RACROUTE calls to the ESM.

*Table 55. Mapping for RELOCATE (File, Alias, External Object, or Directory) Command*

DMSUAUTH Parameters	RACROUTE Calls
<ul style="list-style-type: none"> <li>User ID</li> <li>File pool request code: X'27'</li> <li>Processing code</li> <li>Transaction ID</li> <li>File pool ID</li> <li>Source file, external object, or directory (if an alias, this is ignored)</li> <li>Parent directory</li> <li>Target directory</li> <li>Old alias name (if requested in DMSESM PROFILE file)</li> <li>User data and flags (optional)</li> </ul>	<p>Separate calls for:</p> <ul style="list-style-type: none"> <li>Authority for CMS command:</li> </ul> <pre>RACROUTE REQUEST=AUTH           ,ATTR=UPDATE           ,CLASS=SFSCMD</pre> <p><b>Note:</b> This administrator check occurs only if the administrator check flag s is on.</p> <ul style="list-style-type: none"> <li>READ authority to base file (if an alias)</li> <li>ALTER authority to the old name</li> <li>WRITE authority to parent directory</li> <li>WRITE authority to target directory</li> <li>If a new file/external object, directory, or alias name is specified, then:</li> </ul> <pre>RACROUTE REQUEST=VERIFY RACROUTE REQUEST=DEFINE           ,ENVIR=VERIFY           ,ENTITYX=<i>old name</i>           ,NEWNAMX=<i>new name</i>           ,CHKAUTH=YES           ,SPECIAL=YES           ,TYPE=DEFINE RACROUTE REQUEST=VERIFY</pre>



Table 56. Mapping for RELOCATE (Commit or Rollback)

DMSUAUTH Parameters	RACROUTE Calls
<ul style="list-style-type: none"> <li>Processing code</li> <li>Transaction ID</li> <li>File pool ID</li> </ul>	Separate calls for: <ul style="list-style-type: none"> <li>RACROUTE REQUEST=VERIFY</li> <li>Redefine profiles if request is committed</li> <li>RACROUTE REQUEST=VERIFY</li> </ul>

## RENAME (File, Alias, External Object, or Directory)

To check a user's authority to rename a file, external object, directory, or alias the SFS server calls the SFS object authorization checking CSL routine. The IBM-supplied DMSUAUTH routine calls a series of RACROUTE calls to the ESM.

Table 57. Mapping for RENAME(File, Alias, External Object, or Directory) Command

DMSUAUTH Parameters	RACROUTE Calls
<ul style="list-style-type: none"> <li>User ID</li> <li>File pool request code: X'28'</li> <li>Internal codes</li> <li>Transaction ID</li> <li>File pool ID</li> <li>Old file or directory name</li> <li>Parent directory</li> <li>New file, external object, or directory name (if requested in DMSESM PROFILE file)</li> <li>Old alias name (if requested in DMSESM PROFILE file)</li> <li>New alias name (if requested in DMSESM PROFILE file)</li> <li>User data and flags (optional)</li> </ul>	Separate calls for: <ul style="list-style-type: none"> <li>Authority for CMS command:                             <pre>RACROUTE REQUEST=AUTH ,ATTR=UPDATE ,CLASS=SFSCMD</pre> <p><b>Note:</b> This administrator check occurs only if the administrator check flag <code>x</code> is on.</p> </li> <li>READ authority to base file (if an alias)</li> <li>ALTER authority to the old name</li> <li>WRITE authority to parent directory</li> <li>If a new file, external object, directory, or alias name is specified, then:                             <pre>RACROUTE REQUEST=VERIFY RACROUTE REQUEST=DEFINE ,ENVIR=VERIFY ,ENTITYX=old name ,NEWNAMX=new name ,CHKAUTH=YES ,SPECIAL=YES ,TYPE=DEFINE RACROUTE REQUEST=VERIFY</pre> </li> </ul>

Table 58. Mapping for RENAME (Commit or Rollback)

DMSUAUTH Parameters	RACROUTE Calls
<ul style="list-style-type: none"> <li>Processing code</li> <li>Transaction ID</li> <li>File pool ID</li> </ul>	Separate calls for: <ul style="list-style-type: none"> <li>RACROUTE REQUEST=VERIFY</li> <li>Redefine profiles if request is committed</li> <li>RACROUTE REQUEST=VERIFY</li> </ul>

## 'What Authority?' Query

To obtain information about all the authorities a user has on an object, the SFS server calls the SFS object authorization checking CSL routine. The IBM-supplied DMSUAUTH routine calls a single RACROUTE call to the ESM.

Table 59. Mapping for 'What Authority?' Query

DMSUAUTH Parameters	RACROUTE Call
<ul style="list-style-type: none"><li>• User ID</li><li>• File pool request code: X'15' for GET DIRECTORY ENTRY X'18' for OPEN DIRECTORY</li><li>• Processing code</li><li>• Transaction ID</li><li>• File pool ID</li><li>• Name of file, external object, or directory</li><li>• User data and flags (optional)</li></ul>	<ul style="list-style-type: none"><li>• "What authority?" query for the file, external object, or directory.</li></ul> <pre data-bbox="878 321 1464 422">RACROUTE REQUEST=AUTH           ,STATUS=ACCESS           ,CLASS=FILE/DIRECTRY</pre>

**PI end**

## Appendix E. Sample NetView CLISTS for Distributed CRR Management

If you are managing many distributed CRR recovery servers (or any other distributed virtual machines) from a central site, you need to use NetView CLISTS like the sample CLISTS described in this appendix. See [“CRR Management in the Distributed Environment” on page 308](#). These CLISTS are associated with NetView at the central site. The sample CLIST files can be found on the MAINTvrm 3C2 minidisk.

**Note:** It is assumed NetView release 2 (or later) is already installed on your system. For information on installing NetView, see the NetView documentation.

### Process Explanation

The CRR recovery server virtual machine creates terminal output sequences with no beginning or ending sequences.

Therefore, special NetView CLISTS, message automation entries, and operator definitions are needed to:

- Capture and reformat terminal output
- Forward the reformatted terminal output by a NetView-to-NetView session created by the CRR recovery server operator at the NetView Focal Point™
- Display the terminal output on the CRR recovery server Focal Point operator console in full-screen mode
- Store the terminal output on the Focal Point NetView log and the Distributed Node NetView log.

The CLISTS identify the CRR recovery server's terminal output through message prefixing provided by the programmable operator command SET OUTID. The operator defines a message prefix to be added to terminal output from the CRR recovery server by using the following command, which can be placed in the operator's initial CLIST:

```
route dnid, prop set outid on csid msgprefix
```

where:

**dnid**

is the distributed node user ID

**csid**

is the CRR recover server user ID

**msgprefix**

is the message prefix, which in the sample CLISTS is 'CRRMSG:'. In the sample CLISTS, the colon is necessary to delimit the prefix from the rest of the terminal output.

Error message handling must be set on, so both the error code and text are included in the terminal output, in order for the CRR recovery server terminal output to be formatted correctly by the sample CLISTS. The following command should be processed when logging on to the CRR recovery server:

```
cp set emsg on
```

An example process sequence follows:

1. The PROP SET OUTID command is processed at the NetView Focal Point.
2. CRR recovery server terminal output is captured by NetView message automation at the Distributed Node, and is logged in the NetView log.
3. CRR recovery server terminal output is reformatted for forwarding to the NetView Focal Point node.

4. CRR recovery server terminal output is forwarded. Forwarding occurs either because of a change in the CRR recovery server terminal output identifier, or because the time-out sequence expires (assumes that after one minute, the last terminal output sequence has completed).
5. The Focal Point captures the forwarded CRR recovery server text using the NetView message automation facility and presents the terminal output in full-screen mode at the CRR recovery server operator console.
6. CRR recovery server Focal Point terminal output is captured on the NetView Focal Point log for later retrieval and viewing using the NetView browse function.

## Using the Sample CLIST Files

---

The following section describes the sample CLIST files, operation and testing of the sample CLIST files, potential problems and solutions when using the sample CLIST files, and the use of available automation functions.

### File Descriptions

The sample CLIST files are meant to reside at the Distributed Node or the Focal Point, depending on their function. The sample CLIST files are briefly described below. Refer to the sample CLIST file prolog blocks for more information.

### Naming Conventions

File types for the sample CLIST files follow these conventions:

- APPEND - The contents of these files are to be appended to other existing NetView files
- DNCLST - Identifies a Distributed Node CLIST
- FPCLST - Identifies a Focal Point CLIST
- FPPANEL - NetView Focal Point screen definition files

### Distributed Node CLISTS

- CRROPF APPEND - This file contains the NetView Focal Point and Distributed Node CRR recovery server operator definition. Append the contents of this file to the active NetView DSIOPF file from the appropriate NetView RUN disk.
- MATDNCRR APPEND - This NetView Message Automation Table entry captures CRR recovery server terminal output at the Distributed Node, and calls CRRMSGR DNCLST to reformat the terminal output and forward it to the Focal Point. Append the contents of this file to the active NetView Message Automation Table from the appropriate NetView RUN disk.
- CRRMSGR DNCLST - This CLIST reformats captured CRR recovery server terminal output at the Distributed Node, and forwards it to the Focal Point. Copy this file to the NetView RUN disk, and rename it to CRRMSGR NCCFLST.

### Focal Point CLISTS

- CRROPF APPEND - This is the same file as the Distributed Node CRROPF APPEND file. Append the contents of this file to the active NetView DSIOPF file from the appropriate NetView RUN disk.
- MATFPCRR APPEND - This NetView Message Automation Table entry captures reformatted CRR recovery server terminal output at the Focal Point, and calls AOPMCRRM FPCLST to display the terminal output at the CRR recovery server operator console. For z/VM, append the contents of this file to the active NetView Message Automation Table from the appropriate NetView RUN disk. For MVS™, append the contents of this file to the active NetView Message Automation Table PDS member.
- AOPMCRRM FPCLST - This CLIST displays reformatted CRR recovery server terminal output forwarded to the Focal Point from the Distributed Node. For z/VM, copy this file to the NetView RUN disk, and rename it to AOPMCRRM NCCFLST. For MVS, copy this file to the production CLIST PDS.

- CRRM001-CRRM004 FPPANEL - These files are used for NetView VIEW function screen definition at the Focal Point. For z/VM, copy these files to the NetView RUN disk, and change the file extensions from FPPANEL to NCCFLST. For MVS, copy these files to the NetView production screen PDS.

## Test Program

- CRRMSGR EXEC - This REXX EXEC can be used to test the function of the CLIST files, if a CRR recovery server is not available. It can reside on any valid z/VM user ID (except MAINT).

## Operation and Testing

After the sample CLIST files have been installed, you may want to follow these steps to operate and test them using the CRRMSGR EXEC:

1. Log on to the NetView Focal Point using the CRR recovery server operator user ID (sample CLISTS use CRROPER).
2. Using the NetView *start domain* function, create a NetView-to-NetView session to a NetView Distributed Node. Use the CRR recovery server operator user ID as the operator user ID.
3. Enter the PROP SET OUTID command.
4. Using another terminal, log on a z/VM user ID that has access to the CRRMSGR EXEC.
5. Process CRRMSGR EXEC, which sends terminal output, similar to what you would get from a CRR recovery server, to the operator where the programmable operator to NetView interface is active.
6. Verify the NetView Focal Point operator console receives and displays the CRR-type terminal output in full-screen mode.

## Potential Problems and Solutions

The following is a list of problems you might encounter when you use the sample files:

- Newly installed files do not function properly under NetView:
  - When the sample CLIST files are edited or appended during installation, they must be saved in fixed record format (RECFM=F) with 80-byte records (LRECL=80).
- Substring index error messages appear at the Focal Point during execution of the CRRMSGR DNCLST:
  - Make sure error message handling is on at the CRR recovery server (CP SET EMSG ON).
- The panel display at the Focal Point is not formatted correctly:
  - Make sure the *msgprefix* used in the PROP SET OUTID command ends with a colon.
- CRR recovery server terminal output is not being forwarded by the Single Console Image Facility (SCIF) to the programmable operator virtual machine:
  - Verify the installation of CRR recovery server and operator identification.
- The programmable operator interface is not running in the appropriate user ID:
  - Ensure the programmable operator facility is active in the appropriate virtual machine.
  - Change CRRMSGR EXEC to reflect the user ID where the programmable operator interface is running.
- The programmable operator to NetView interface is not active:
  - Ensure the installation procedure is complete and this interface has been activated.
- NetView is not the logical operator for the programmable operator:
  - Enter the following command from a NetView Distributed Node console:
 

```
prop lglopr rpl
```
  - You will receive a message on the NetView operator console stating NetView is now the programmable operator logical operator. Retry the CRRMSGR EXEC. If this sequence is successful,

ensure NetView initial automation issues the PROP LGLOPR RPL command to become the logical operator.

- CRR recovery server terminal output is in the NetView Distributed Node log, but is not displayed in full-screen mode at the NetView Focal Point:
  - Verify the message automation entries have been included in both the NetView Distributed Node and Focal Point Message Automation Tables.
  - Verify the Message Automation Table is the active table by issuing the following NetView command from a NetView operator console:

```
automsg status
```

- Verify the CRRMSGR CLIST is accessible by NetView. If the installation was done without recycling NetView, ensure the NetView disks have been re-accessed since installation. Ensure the session from the NetView Focal Point to the NetView Distributed Node is active. Rerun the test when all items have been verified.
- Reformatted CRR recovery server terminal output does not appear in the NetView Focal Point log:
  - Verify the message automation entries have been included in the active NetView Focal Point Message Automation Table. Verify AOPMCRRM CLIST is available. Verify CRRM001 FPPANEL through CRRM004 FPPANEL are accessible from the production panel library. Ensure the session from the NetView Focal Point to the NetView Distributed Node is active. Rerun the CRRMSGR EXEC when all items have been verified.

## Use of Available Automation Functions

If your initial installation included the Auto-Logon function provided as part of NetView release 1 automated operations, change the MATDNCRR operator user ID to reflect the operator user ID specified in the AOPIOC00 CLIST for the distributed node to be monitored. A change to the MATDNCRR will be required to select the DNAUTOOP message forwarding operator user ID.

## Listings of Sample Files

Prolog blocks are placed at end of file for executable files, to speed execution. The listings are in alphabetic order.

### AOPMCRRM FPCLST

```
CLIST
&CONTROL ERR
&CNAME = AOPMCRRM
MSG LOG &CNAME : &PARMSTR
&BGNMSG = CRR.BGN
&TXTMSG = CRR.TXT
&ENDMSG = CRR.END
&TGLOBAL CRRMTI
&TGLOBAL CRRMSGID CRRMUSER CRRROUTID
&TGLOBAL CRRVMNODE CRRNVNODE CRRDATE CRRTIME
PARSEL2R PARMSTR CRRTYPE CRRWRK
&IF &CRRTYPE EQ &ENDMSG &THEN &GOTO -CRRVIEW
&IF &CRRTYPE EQ &BGNMSG &THEN &GOTO -CRRINIT
&IF &CRRTYPE EQ &TXTMSG &THEN &GOTO -CRRTEXT
&EXIT
*
-CRRINIT
PARSEL2R CRRWRK CRRROUTID CRRMSGID CRRVMNODE CRRNVNODE
&TGLOBAL CRRMTI
&CRRMTI = 1
&EXIT
*
-CRRTEXT
&TGLOBAL CRRMT&CRRMTI
PARSEL2R CRRWRK Z CRRMT&CRRMTI
&CRRMTI = &CRRMTI + 1
&EXIT
*
```

```

-CRRVIEW
&WHITE = '%'
&WHITER = '}'
&BLUE = '+'
&BLUER = '~'
&GREEN = '>'
&YELLOW = ']'
&PINK = '|-'
&RED = '@'
&TURQ = '<'
&CRRVMNODE = &CONCAT &WHITE &CRRVMNODE
&CRRVMUSER = &CONCAT &WHITE &CRRVMUSER
&CRRNVNODE = &CONCAT &WHITE &CRRNVNODE
&CRRDATE = &CONCAT &WHITE &DATE
&CRRTIME = &CONCAT &WHITE &TIME
&CRRMSGID = &CONCAT DMS &CRRMSGID
&CRRMSGID = &CONCAT &WHITE &CRRMSGID
&OI = 1
-VIEWNEXT
&TGLOBAL CRRMT&OI
&SI = &OI
&IF &LENGTH &SI EQ 1 &THEN &SI = &CONCAT 0 &SI
&SN = &CONCAT CRR &SI
&SN = &CONCAT &SN MSG
&&SN = &CONCAT &GREEN &CRRMT&OI
&OI = &OI + 1
&IF &OI LT &CRRMTI &THEN &GOTO -VIEWNEXT
VIEW 4 CRRM001
&EXIT

```

```

*****
*
* File Name: AOPMCRRM FPCLST
*
* Function: This CLIST is called by NetView Message Automation to display
* reformatted CRR messages forwarded to the NetView Focal Point
* from a Distributed Node and captured by Focal Point message
* automation, in full-screen mode using the NetView VIEW command.
* The VIEW command panels referenced are contained in files
* CRRM001-CRRM004 FPPANEL (NCCFLST). The reformatted messages are
* prefix as follows: 'CRR.BGN' (first line), 'CRR.TXT.' (middle
* lines), 'CRR.END' (last line).
*
* Usage: This is a Focal Point CLIST.
*
* VM - Copy this file to the NetView production CLIST minidisk and
* rename it to AOPMCRRM NCCFLST.
*
* MVS - Copy this file to the NetView production CLIST PDS.
*
*****

```

## CRRMSGR DNCLST

```

CLIST
&CONTROL ERR
&CNAME = CRRMSGR
* MSG LOG &CNAME : &PARMSTR
&BGNMSG = CRR.BGN
&TXTMSG = CRR.TXT
&ENDMSG = CRR.END
&EOMTYPE = ''
&EOMTEXT = 'End of message'
&CGLOBAL CRROP SYSNAME NVWNAME
&TGLOBAL CRRINIT CRRMI
&TGLOBAL CRRMSGID CRRUSERID CRRTIMER CRROUTID
*
&IF .&CRRINIT EQ .YES &THEN &GOTO -CRRCPARM
&CRRINIT = YES
&CRRMI = 0
-CRRCPARM
PARSEL2R PARMSTR CRRCMD ZZZZ
&IF .&CRRCMD EQ .INIT &THEN &GOTO -CRRINITIAL
&IF .&CRRCMD EQ .TIMER &THEN &GOTO -CRROPTMR
*-----*
PARSEL2R PARMSTR IUSERID CRRVMNODE /:/ CRROUTID /:/ IMSGID IMSGTXT
-PROPO0
&IF .&CRRMSGID EQ . &THEN &CRRMSGID = &IMSGID

```

## NetView CLISTS

```
&IF .&CRRUSERID EQ . &THEN &CRRUSERID = &IUSERID
&IF .&CRRMSGID NE .&IMSGID &THEN &GOTO -CRRINTOUT
&IF .&CRRUSERID NE .&IUSERID &THEN &GOTO -CRRINTOUT
&IF .&EOMTEXT EQ .&IMSGTXT &THEN &GOTO -CRRPEOM
&CRRMSGID = &IMSGID
&CRRUSERID = &IUSERID
-CRRROUTMSG
&CRRMI = &CRRMI + 1
&TGLOBAL CRRMSG&CRRMI
&CRRMSG&CRRMI = &IMSGTXT
&IF .&CRRTIMER EQ .YES &THEN &EXIT
&CRRTIMER = YES
EVERY 1, ID=CRREND, CRRMSGR TIMER
&EXIT
*
-CRRINTOUT
PURGE TIMER=CRREND
&CRRTIMER = NO
&IF .&CRRMSGID = .&CNAME &THEN &GOTO -CRRIO002
&CRRMSGLN = &LENGTH &CRRMSGID
&CRRMSGLN = &CRRMSGLN - 3
&CRRMSGID = &SUBSTR &CRRMSGID 4 &CRRMSGLN
&WRITE &BGNMSG &CRRROUTID &CRRMSGID &CRRVMNODE &CRRUSERID &MSGORIGIN
&OI = 1
-CRRIO001
&TGLOBAL CRRMSG&OI
&WRITE &TXTMSG &CRRMSGID &CRRMSG&OI
&OI = &OI + 1
&IF &OI LE &CRRMI &THEN &GOTO -CRRIO001
&EOMTYPE = 'MsgID change'
&WRITE &ENDMSG &EOMTEXT '(' &EOMTYPE ')'
-CRRIO002
&CRRMI = 0
&CRRMSGID = &IMSGID
&CRRUSERID = &IUSERID
&GOTO -CRRROUTMSG
*
-CRRPEOM
&EOMTYPE = 'Text'
&GOTO -CRRROUTPUT
-CRRPTMR
&EOMTYPE = 'Timer'
*&GOTO -CRRROUTPUT
-CRRROUTPUT
PURGE TIMER=CRREND
&CRRTIMER = NO
&CRRMSGLN = &LENGTH &CRRMSGID
&CRRMSGLN = &CRRMSGLN - 3
&CRRMSGID = &SUBSTR &CRRMSGID 4 &CRRMSGLN
&WRITE &BGNMSG &CRRROUTID &CRRMSGID &CRRVMNODE &CRRUSERID &MSGORIGIN
&OI = 1
-CRRP001
&TGLOBAL CRRMSG&OI
&WRITE &TXTMSG &CRRMSGID &CRRMSG&OI
&OI = &OI + 1
&IF &OI LE &CRRMI &THEN &GOTO -CRRP001
&WRITE &ENDMSG &EOMTEXT '(' &EOMTYPE ')'
&CRRMI = 0
&CRRMSGID = &CNAME
&CRRUSERID = &CNAME
&EXIT
*
-CRRINITIAL
&CRRMSGID = &CNAME
&CRRUSERID = &CNAME
&CRRMSGTXT = &CNAME
&CRRINIT = NO
&CRRMI = 0
&WRITE &CNAME - Re-Initialized
&EXIT
*****
*
* File Name: CRRMSGR DNCLST
*
* Function: This CLIST is called by NetView Message Automation to reformat
*           captured CRR messages from the PROP interface, and forward the
*           reformatted messages to the Focal Point. Messages from CRR are
*           recognized by the message prefix assigned by the PROP SET OUTID
*           command. Since CRR does not use an end-of-message indicator,
*           this CLIST processes CRR message lines until either the message
*           indicator changes or a time-out occurs. Captured messages are
*           reformatted by prefixing each line with 'CRR.BGN' (first line),
```



```

*          'CRR.TXT' (middle lines), or 'CRR.END' (last line). Reformatted *
*          messages are forwarded to the NetView Focal Point by the Net- *
*          View-to-NetView session established by the CRR Focal Point oper- *
*          ator. *
* *
* Usage: This is a Distributed Node CLIST. *
* *
*          VM - Copy this file to the NetView production CLIST minidisk and *
*          rename it to CRRMSGR NCCFLST. *
* *
*****

```

## CRRMSGR EXEC

```

/* */
Trace 0
TELL OP 'DMS6SH3303E LU USIBMNY.SFSTEST6, previously reported to be exposed to state'
TELL OP 'DMS6SH3303E inconsistency with respect to other resources for'
TELL OP 'DMS6SH3303E LUWID USIBMNY.NEOWONE.A23424B9FA00.0002 with token 000006A6 has'
TELL OP 'DMS6SH3303E been found to be out of synchronization.'
/* .newmsg */
TELL OP 'DMS6S03301W A decision of commit has been made for LU USIBMNY.SFSTEST6'
TELL OP 'DMS6S03301W participating in LUWID USIBMNY.NEOWONE.A23424B9FA00.0002'
TELL OP 'DMS6S03301W with token 000006A6. As a result, the following resources may be'
TELL OP 'DMS6S03301W in states that are inconsistent:'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST0 with index 1'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST1 with index 2'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST2 with index 3'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST3 with index 4'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST4 with index 5'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST5 with index 6'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST6 with index 7'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST7 with index 8'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST8 with index 9'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST9 with index 10'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST10 with index 11'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST11 with index 12'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST12 with index 13'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST13 with index 14'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST14 with index 15'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST15 with index 16'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST16 with index 17'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST17 with index 18'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST18 with index 19'
TELL OP 'DMS6S03301W LU USIBMNY.SFSTST19 with index 20'
/* .newmsg */
TELL OP 'DMS6SH3302I LU USIBMNY.SFSTEST6, previously reported to be exposed to'
TELL OP 'DMS6SH3302I state inconsistency with respect to other resources or'
TELL OP 'DMS6SH3302I LUWID USIBMNY.NEOWONE.A23424B9FA00.0002 with token 000006A6 has'
TELL OP 'DMS6SH3302I been found to be synchronized'
Exit 1
/*****
/*
/* File Name: CRRMSGR EXEC */
/*
/* Function: This is a REXX EXEC that can be used to verify that the CLIST */
/* files have been installed correctly, if a CRR recovery server */
/* is not readily available. The EXEC generates simulated CRR */
/* messages to PROP. The messages are then captured by NetView */
/* and reflected to the NetView Focal Point operator (CRROPER) as */
/* if CRR were active. */
/*
/* The following conditions are represented in this EXEC: */
/*
/* 1. The message identifier change from DMS6SH3303E to */
/* DMS6S03301W causes full-screen display to occur for message */
/* DMS6SH3303E. */
/*
/* 2. Message DMS6S03301W uses multiple screens at the Focal */
/* Point. */
/*
/* 3. Message DMS6SH3302I is displayed after the message time-out */
/* occurs. */
/*
/* Usage: Copy this file to a valid VM user logon. */
/*
*****/

```

## CRRM001 FPPANEL

```

/*****/
/*
/* File Name: CRRM001 FPPANEL
/*
/* Function: Define a screen for the displaying of reformatted CRR messages
/* at the NetView Focal Point by AOPMCRRM FPCLST (NCCFLST).
/*
/* Usage: This is a Focal Point file.
/*
/* VM - Copy this file to the NetView production CLIST minidisk and
/* rename it to CRRM001 NCCFLST.
/*
/* MVS - Copy this file to the NetView production panel library PDS.
/*
/*****/
CRRM002 Unsolicited CRR Message Processor (Cont'd)
***
+CRRM001R Unsolicited CRR Message Processor
+ Outid : &CRRROUTID + Msgid : &CRRMSGID
+ Received from : &CRRVMNODE + VM userid : &CRRVMUSER + NetView : &CRRNVNODE
+ date : &CRRDATE + time : &CRRTIME
+Message
+Text
¢
&CRR01MSG
&CRR02MSG
&CRR03MSG
&CRR04MSG
&CRR05MSG
&CRR06MSG
&CRR07MSG
&CRR08MSG
&CRR09MSG
&CRR10MSG
¢
%Action===>&similar.&amp;.CUR
$ PF1= Help PF2= End PF3= Return
$ PF6= Roll %PF8= Forward $PF11= Entry Point

```

## CRRM002 FPPANEL

```

/*****/
/*
/* File Name: CRRM002 FPPANEL
/*
/* Function: Define a screen for the displaying of reformatted CRR messages
/* at the NetView Focal Point by AOPMCRRM FPCLST (NCCFLST).
/*
/* Usage: This is a Focal Point file.
/*
/* VM - Copy this file to the NetView production CLIST minidisk and
/* rename it to CRRM002 NCCFLST.
/*
/* MVS - Copy this file to the NetView production panel library PDS.
/*
/*****/
CRRM003 Unsolicited CRR Message Processor (Cont'd)
***
+CRRM002R Unsolicited CRR Message Processor (Cont'd)
+ Outid : &CRRROUTID + Msgid : &CRRMSGID
+ Received from : &CRRVMNODE + VM userid : &CRRVMUSER + NetView : &CRRNVNODE
+ date : &CRRDATE + time : &CRRTIME
+Message
+Text
¢
&CRR11MSG
&CRR12MSG
&CRR13MSG
&CRR14MSG
&CRR15MSG
&CRR16MSG

```

```

&CRR17MSG
&CRR18MSG
&CRR19MSG
&CRR20MSG

¢
%Action===>&similar.&amp;.CUR
$   PF1= Help   PF2= End   PF3= Return
$   PF6= Roll           %PF8= Forward $PF11= Entry Point

```

## CRRM003 FPPANEL

```

/*****
/*
/* File Name: CRRM003 FPPANEL
/*
/* Function: Define a screen for the displaying of reformatted CRR messages
/*           at the NetView Focal Point by AOPMCRRM FPCLST (NCCFLST).
/*
/* Usage: This is a Focal Point file.
/*
/*           VM - Copy this file to the NetView production CLIST minidisk and
/*           rename it to CRRM003 NCCFLST.
/*
/*           MVS - Copy this file to the NetView production panel library PDS.
/*
/*****
CRRM004 Unsolicited CRR Message Processor (Cont'd)
***
+CRRM003R           Unsolicited CRR Message Processor (Cont'd)

+ Outid : &CRRROUTID +           Msgid : &CRRMSGID
+ Received from : &CRRVMNODE +   VM userid : &CRRVMUSER +   NetView : &CRRNVNODE
+           date : &CRRDATE   +   time : &CRRTIME

+Message
+Text
¢
&CRR21MSG
&CRR22MSG
&CRR23MSG
&CRR24MSG
&CRR25MSG
&CRR26MSG
&CRR27MSG
&CRR28MSG
&CRR29MSG
&CRR30MSG

¢
%Action===>&similar.&amp;.CUR
$   PF1= Help   PF2= End   PF3= Return
$   PF6= Roll           %PF8= Forward $PF11= Entry Point

```

## CRRM004 FPPANEL

```

/*****
/*
/* File Name: CRRM004 FPPANEL
/*
/* Function: Define a screen for the displaying of reformatted CRR messages
/*           at the NetView Focal Point by AOPMCRRM FPCLST (NCCFLST).
/*
/* Usage: This is a Focal Point file.
/*
/*           VM - Copy this file to the NetView production CLIST minidisk and
/*           rename it to CRRM004 NCCFLST.
/*
/*           MVS - Copy this file to the NetView production panel library PDS.
/*
/*****
CRRM004R           Unsolicited CRR Message Processor (Cont'd)
***
+CRRM004R           Unsolicited CRR Message Processor (Cont'd)

+ Outid : &CRRROUTID +           Msgid : &CRRMSGID
+ Received from : &CRRVMNODE +   VM userid : &CRRVMUSER +   NetView : &CRRNVNODE

```

## NetView CLISTS

```
+          date : &CRRDATE  +  time : &CRRTIME

+Message
+Text
¢
&CRR31MSG
&CRR32MSG
&CRR33MSG
&CRR34MSG
&CRR35MSG
&CRR36MSG
&CRR37MSG
&CRR38MSG
&CRR39MSG
&CRR40MSG

¢
%Action===>&similar.&amp.CUR
$  PF1= Help  PF2= End  PF3= Return
$  PF6= Roll  %PF8= Forward $PF11= Entry Point
```

## CRROPF APPEND

```
*****
*
* File Name: CRROPF APPEND
*
* Function: CRR operator definition for NetView Distributed Node and Focal
*           Point. This definition is used for NetView-to-NetView session
*           and Message Automation Table entries.
*
* Usage: This file is for use by both Distributed Nodes and Focal Point.
*        It is assumed that the operator is CRROPER, with a profile named
*        DSIPROFB.
*
*        VM - Append the contents of this file to the active NetView DSIOPF
*        file from the appropriate RUN minidisk.
*
*        MVS - Append the contents of this file to the active NetView DSIOPF
*        PDS member.
*
*****
CRROPER      OPERATOR      PASSWORD=CRROPER
             PROFILEN     DSIPROFB
```

## MATDNCRR APPEND

```
*****
*
* File Name: MATDNCRR APPEND
*
* Function: NetView Message Automation Table entry for the Distributed Node.
*           This entry captures CRR messages and calls CRRMSGR DNCLST
*           (NCCFLST) to reformat the messages and forward them to the Focal
*           Point.
*
* Usage: This is a Distributed Node file. This file assumes the operator is
*        CRROPER, and assumes that CRR messages are prefix with 'CRRMSG:'
*        by the PROP SET OUTID command.
*
*        VM - Append the contents of this file to the active NetView Message
*        Automation Table from the appropriate RUN minidisk.
*
*****
IF TEXT='CRRMSG:'.
  & TEXT=MESSAGE
  THEN EXEC(CMD('CRRMSGR ' MESSAGE)
            ROUTE(ONE CRROPER))
  DISPLAY(N) NETLOG(Y);
*
```

## MATFPCRR APPEND

```

*****
*
* File Name: MATFPCRR APPEND
*
* Function: NetView Message Automation Table entry for the Focal Point.
*           This entry captures reformatted CRR messages and calls AOPMCRRM
*           FPCLST (NCCFLST) to display the messages at the Focal Point.
*
* Usage: This is a Focal Point file. This file assumes the operator is
*        CRROPER, and that the reformatted messages are prefix by 'CRR.'
*        (as in 'CRR.BGN', 'CRR.TXT', 'CRR.END').
*
*        VM - Append the contents of this file to the active NetView Message
*             Automation Table from the appropriate RUN minidisk.
*
*        MVS - Append the contents of this file to the active NetView Mes-
*             sage Automation Table PDS member.
*
*****
IF MSGID='CRR.' .
  & TEXT = MESSAGE
  THEN EXEC(CMD('AOPMCRRM ' MESSAGE)
            ROUTE(ONE CRROPER))
  DISPLAY(N) NETLOG(Y);
*

```



## Notices

---

This information was developed for products and services offered in the US. This material might be available from IBM® in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
US*

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

*Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
19-21, Nihonbashi-Hakozakicho, Chuo-ku  
Tokyo 103-8510, Japan*

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

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

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
US*

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

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

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

#### COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

## Programming Interface Information

---

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

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

**PI**

<...Programming Interface information...>

**PI end**

## Trademarks

---

IBM, the IBM logo, and [ibm.com](https://www.ibm.com/legal/us/en/copytrade.shtml)<sup>®</sup> are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at [IBM copyright and trademark information - United States](https://www.ibm.com/legal/us/en/copytrade.shtml) (<https://www.ibm.com/legal/us/en/copytrade.shtml>).

The registered trademark Linux<sup>®</sup> is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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



## Terms and Conditions for Product Documentation

---

Permissions for the use of these publications are granted subject to the following terms and conditions.

### Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

### Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

### Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

### Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

## IBM Online Privacy Statement

---

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at <https://www.ibm.com/privacy>
- [https://www.ibm.com/privacy#Cookies\\_and\\_Similar\\_Technologies](https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)



# Bibliography

---

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see [z/VM: General Information](#).

## Where to Get z/VM Information

---

The current z/VM product documentation is available in [IBM Documentation - z/VM \(https://www.ibm.com/docs/en/zvm\)](https://www.ibm.com/docs/en/zvm).

## z/VM Base Library

---

### Overview

- [z/VM: License Information](#), GI13-4377
- [z/VM: General Information](#), GC24-6286

### Installation, Migration, and Service

- [z/VM: Installation Guide](#), GC24-6292
- [z/VM: Migration Guide](#), GC24-6294
- [z/VM: Service Guide](#), GC24-6325
- [z/VM: VMSES/E Introduction and Reference](#), GC24-6336

### Planning and Administration

- [z/VM: CMS File Pool Planning, Administration, and Operation](#), SC24-6261
- [z/VM: CMS Planning and Administration](#), SC24-6264
- [z/VM: Connectivity](#), SC24-6267
- [z/VM: CP Planning and Administration](#), SC24-6271
- [z/VM: Getting Started with Linux on IBM Z](#), SC24-6287
- [z/VM: Group Control System](#), SC24-6289
- [z/VM: I/O Configuration](#), SC24-6291
- [z/VM: Running Guest Operating Systems](#), SC24-6321
- [z/VM: Saved Segments Planning and Administration](#), SC24-6322
- [z/VM: Secure Configuration Guide](#), SC24-6323
- [z/VM: TCP/IP LDAP Administration Guide](#), SC24-6329
- [z/VM: TCP/IP Planning and Customization](#), SC24-6331
- [z/OS and z/VM: Hardware Configuration Manager User's Guide \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sc342670/\\$file/eequ100\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sc342670/$file/eequ100_v2r4.pdf), SC34-2670

### Customization and Tuning

- [z/VM: CP Exit Customization](#), SC24-6269
- [z/VM: Performance](#), SC24-6301

### Operation and Use

- [z/VM: CMS Commands and Utilities Reference](#), SC24-6260

- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268
- *z/VM: System Operation*, SC24-6326
- *z/VM: TCP/IP User's Guide*, SC24-6333
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

## Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940
- *CPI Communications Reference*, SC26-4399
- *Common Programming Interface Resource Recovery Reference*, SC31-6821
- *z/OS: IBM Tivoli Directory Server Plug-in Reference for z/OS* ([https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa760169/\\$file/glpa300\\_v2r4.pdf](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa760169/$file/glpa300_v2r4.pdf)), SA76-0169
- *z/OS: Language Environment Concepts Guide* ([https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380687/\\$file/ceea800\\_v2r4.pdf](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380687/$file/ceea800_v2r4.pdf)), SA38-0687
- *z/OS: Language Environment Debugging Guide* ([https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4ga320908/\\$file/ceea100\\_v2r4.pdf](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4ga320908/$file/ceea100_v2r4.pdf)), GA32-0908
- *z/OS: Language Environment Programming Guide* ([https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380682/\\$file/ceea200\\_v2r4.pdf](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380682/$file/ceea200_v2r4.pdf)), SA38-0682
- *z/OS: Language Environment Programming Reference* ([https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380683/\\$file/ceea300\\_v2r4.pdf](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380683/$file/ceea300_v2r4.pdf)), SA38-0683

- [z/OS: Language Environment Runtime Messages \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380686/\\$file/ceea900\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380686/$file/ceea900_v2r4.pdf), SA38-0686
- [z/OS: Language Environment Writing Interlanguage Communication Applications \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380684/\\$file/ceea400\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa380684/$file/ceea400_v2r4.pdf), SA38-0684
- [z/OS: MVS Program Management Advanced Facilities \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa231392/\\$file/ieab200\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa231392/$file/ieab200_v2r4.pdf), SA23-1392
- [z/OS: MVS Program Management User's Guide and Reference \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa231393/\\$file/ieab100\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa231393/$file/ieab100_v2r4.pdf), SA23-1393

## Diagnosis

- [z/VM: CMS and REXX/VM Messages and Codes](#), GC24-6255
- [z/VM: CP Messages and Codes](#), GC24-6270
- [z/VM: Diagnosis Guide](#), GC24-6280
- [z/VM: Dump Viewing Facility](#), GC24-6284
- [z/VM: Other Components Messages and Codes](#), GC24-6300
- [z/VM: TCP/IP Diagnosis Guide](#), GC24-6328
- [z/VM: TCP/IP Messages and Codes](#), GC24-6330
- [z/VM: VM Dump Tool](#), GC24-6335
- [z/OS and z/VM: Hardware Configuration Definition Messages \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sc342668/\\$file/cbdm100\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sc342668/$file/cbdm100_v2r4.pdf), SC34-2668

## z/VM Facilities and Features

---

### Data Facility Storage Management Subsystem for VM

- [z/VM: DFSMS/VM Customization](#), SC24-6274
- [z/VM: DFSMS/VM Diagnosis Guide](#), GC24-6275
- [z/VM: DFSMS/VM Messages and Codes](#), GC24-6276
- [z/VM: DFSMS/VM Planning Guide](#), SC24-6277
- [z/VM: DFSMS/VM Removable Media Services](#), SC24-6278
- [z/VM: DFSMS/VM Storage Administration](#), SC24-6279

### Directory Maintenance Facility for z/VM

- [z/VM: Directory Maintenance Facility Commands Reference](#), SC24-6281
- [z/VM: Directory Maintenance Facility Messages](#), GC24-6282
- [z/VM: Directory Maintenance Facility Tailoring and Administration Guide](#), SC24-6283

### Open Systems Adapter

- [Open Systems Adapter-Express Customer's Guide and Reference \(https://www.ibm.com/support/pages/node/6019492\)](https://www.ibm.com/support/pages/node/6019492), SA22-7935
- [Open Systems Adapter-Express Integrated Console Controller User's Guide \(https://www.ibm.com/support/pages/node/6019810\)](https://www.ibm.com/support/pages/node/6019810), SC27-9003
- [Open Systems Adapter-Express Integrated Console Controller 3215 Support \(https://www.ibm.com/docs/en/SSLTBW\\_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm\)](https://www.ibm.com/docs/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SA23-2247
- [Open Systems Adapter/Support Facility on the Hardware Management Console \(https://www.ibm.com/docs/en/SSLTBW\\_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm\)](https://www.ibm.com/docs/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SC14-7580

## Performance Toolkit for z/VM

- [z/VM: Performance Toolkit Guide](#), SC24-6302
- [z/VM: Performance Toolkit Reference](#), SC24-6303

## RACF® Security Server for z/VM®

- [z/VM: RACF Security Server Auditor's Guide](#), SC24-6305
- [z/VM: RACF Security Server Command Language Reference](#), SC24-6306
- [z/VM: RACF Security Server Diagnosis Guide](#), GC24-6307
- [z/VM: RACF Security Server General User's Guide](#), SC24-6308
- [z/VM: RACF Security Server Macros and Interfaces](#), SC24-6309
- [z/VM: RACF Security Server Messages and Codes](#), GC24-6310
- [z/VM: RACF Security Server Security Administrator's Guide](#), SC24-6311
- [z/VM: RACF Security Server System Programmer's Guide](#), SC24-6312
- [z/VM: Security Server RACROUTE Macro Reference](#), SC24-6324

## Remote Spooling Communications Subsystem Networking for z/VM

- [z/VM: RSCS Networking Diagnosis](#), GC24-6316
- [z/VM: RSCS Networking Exit Customization](#), SC24-6317
- [z/VM: RSCS Networking Messages and Codes](#), GC24-6318
- [z/VM: RSCS Networking Operation and Use](#), SC24-6319
- [z/VM: RSCS Networking Planning and Configuration](#), SC24-6320
- [z/OS: Network Job Entry \(NJE\) Formats and Protocols \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa320988/\\$file/hasa600\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4sa320988/$file/hasa600_v2r4.pdf), SA32-0988

## Prerequisite Products

---

### Device Support Facilities

- [Device Support Facilities \(ICKDSF\): User's Guide and Reference \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4gc350033/\\$file/ickug00\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4gc350033/$file/ickug00_v2r4.pdf), GC35-0033

### Environmental Record Editing and Printing Program

- [Environmental Record Editing and Printing Program \(EREP\): Reference \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4gc350152/\\$file/ifc2000\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4gc350152/$file/ifc2000_v2r4.pdf), GC35-0152
- [Environmental Record Editing and Printing Program \(EREP\): User's Guide \(https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4gc350151/\\$file/ifc1000\\_v2r4.pdf\)](https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r4gc350151/$file/ifc1000_v2r4.pdf), GC35-0151

# Index

## Special Characters

\$\$SAVED \$POOLDEF temporary file [111](#)  
\$STEMP \$INPUT temporary file [201](#), [220](#), [520](#), [526](#)  
\$STEMP \$BACKUP temporary file [106](#), [367](#), [501](#), [507](#)  
\$STEMP \$POOLDEF temporary file [199](#), [201](#), [220](#), [267](#), [505](#), [507](#), [520](#), [523](#), [526](#)

## Numerics

191 minidisk  
  for administration machines [142](#), [262](#)  
  for server machines [254](#)  
  for server, replacing [129](#)  
4KB blocks  
  changing a user's allocation of [91](#)  
  deleting file space allocation of [93](#)  
  giving file spaces allocations of [72](#)  
  giving users allocations of [72](#)  
  maximum number per file pool [685](#)  
  maximum number per file space [685](#)  
  maximum number per storage group minidisk [685](#)

## A

A-disk  
  considerations for [73](#)  
  defining defaults for users [74](#)  
  for administration machines [142](#), [262](#)  
  for server machines [254](#)  
  for server, replacing [129](#)  
  in a remote file pool [236](#)  
  moving contents to an SFS directory [78](#)  
access BFS request [156](#)  
access mode required for file pool minidisks  
  when adding space to file pool [198](#)  
  when generating a file pool [260](#)  
  when regenerating a file pool [218](#)  
access security information [234](#)  
accessing another user's directory [163](#)  
ACCESSORS option of the CMS QUERY command [547](#)  
ACCOUNT startup parameter [224](#), [229](#), [347](#)  
accounting  
  exit  
    customizing accounting records for alternate user  
    IDs [231](#)  
    file pool [229](#)  
  facility  
    brief overview of [29](#)  
    customizing records created by [229](#)  
    in dedicated maintenance mode [224](#)  
    records generated by [224](#)  
    starting [223](#)  
    startup parameters related to [347](#)  
    stopping [224](#)  
  records [224](#)  
ACCT operand of the OPTION control statement [223](#), [258](#)

adding minidisks  
  in dedicated maintenance mode [201](#)  
  in multiple user mode [199](#)  
  required tasks [196](#)  
  using FILEPOOL MINIDISK command [199](#)  
  using FILESERV MINIDISK command [201](#)  
ADMIN startup parameter  
  description of [342](#)  
  specified for VMSYS, VMPSFS, VMSYSU, VMSYSR [42](#)  
administration (CRR) [315](#)  
administration authority  
  DELETE ADMINISTRATOR command [400](#)  
  deleting [142](#)  
  description of [22](#), [25](#), [139](#)  
  ENROLL ADMIN command [417](#)  
  for VMSYS, VMPSFS, VMSYSU, and VMSYSR [42](#)  
  GRANT ADMIN operator command [541](#)  
  granting [141](#)  
  granting by ADMIN startup parameter [342](#)  
  granting permanently [141](#)  
  internal recording of [138](#)  
  REVOKE ADMIN operator command [663](#)  
  when ESECURITY is in effect [162](#)  
administration commands  
  checking during administration requests [162](#)  
  DATASPACE [396](#)  
  DELETE ADMINISTRATOR [400](#)  
  DELETE LOCK [403](#)  
  DELETE PUBLIC [406](#)  
  DELETE USER [408](#)  
  description of [357](#)  
  ENROLL ADMINISTRATOR [417](#)  
  ENROLL PUBLIC [420](#)  
  ENROLL USER [422](#)  
  FILEPOOL BACKUP [432](#)  
  FILEPOOL CLEANUP [437](#)  
  FILEPOOL DISABLE [443](#)  
  FILEPOOL ENABLE [447](#)  
  FILEPOOL FILELOAD [451](#)  
  FILEPOOL FORMAT AUDIT [455](#)  
  FILEPOOL LIST BACKUP [459](#)  
  FILEPOOL LIST MINIDISK [467](#)  
  FILEPOOL MINIDISK [470](#)  
  FILEPOOL RELOAD [476](#)  
  FILEPOOL RENAME [484](#)  
  FILEPOOL RESTORE [488](#)  
  FILEPOOL UNLOAD [495](#)  
  listing of each [26–28](#)  
  MODIFY USER [544](#)  
  QUERY ACCESSORS [547](#)  
  QUERY DATASPACE [550](#)  
  QUERY FILEPOOL AGENT [557](#)  
  QUERY FILEPOOL CATALOG [560](#)  
  QUERY FILEPOOL CONFLICT [563](#)  
  QUERY FILEPOOL COUNTER [569](#)  
  QUERY FILEPOOL CRR [574](#)  
  QUERY FILEPOOL DISABLE [577](#)



- administration commands (*continued*)
  - QUERY FILEPOOL LOG [583](#)
  - QUERY FILEPOOL MINIDISK [586](#)
  - QUERY FILEPOOL OVERVIEW [589](#)
  - QUERY FILEPOOL REPORT [592](#)
  - QUERY FILEPOOL STATUS [623](#)
  - QUERY FILEPOOL STORGRP [652](#)
  - QUERY LIMITS [655](#)
  - security [139](#)
  - SET THRESHOLD [664](#)
- administration machine
  - defining [261](#)
  - definition of [25](#)
  - MAXCONN setting for [142](#)
  - work minidisk for [142](#)
- Advanced Program-to-Program Communication/VM (APPC/VM)
  - paths [540](#)
  - using with TSAF, ISFC, and AVS [234](#)
- agents [629](#), [644](#)
- aliases
  - creating as an administrator [140](#)
  - effect of FILEPOOL FILELOAD on [127](#)
  - effect of FILEPOOL RELOAD FILES on [127](#)
  - erased [121](#)
  - revoked [121](#)
  - unresolved [122](#)
  - when transferring ownership [99](#)
- allocation and sync point tree [302](#)
- AOPMCRRM FPCLST [710](#)
- APPC/VM (Advanced Program-to-Program Communication/VM)
  - paths [540](#)
  - using with TSAF, ISFC, and AVS [234](#)
- APPC/VM VTAM Support (AVS)
  - used to set up file pool for remote use [233](#)
- APPCPASS z/VM system directory control statement [237](#)
- application
  - considerations when enrolling users [74](#)
- APPLMON operand of the OPTION control statement [258](#)
- architected limits [28](#), [685](#)
- audit
  - complete [144](#)
  - output file [144](#)
  - partial [143](#)
- AUDIT ddname [348](#), [364](#), [505](#)
- AUDIT operator command
  - instructions for using [145](#)
  - reference information for [362](#)
- audit output file [504](#)
- AUDIT startup parameter [145](#), [348](#)
- auditing security
  - AUDIT operator command [362](#)
  - brief overview of [29](#)
  - changing output file for [144](#)
  - defining output file for [144](#)
  - FILEPOOL FORMAT AUDIT command [455](#)
  - formatted output [160](#)
  - formatting the output of [146](#)
  - security audit facility [143](#)
  - starting and stopping [145](#)
  - startup parameters related to [348](#)
- AUDITPOINT field in FILEPOOL FORMAT AUDIT output [160](#)

- AUTHORITY CHECKING DONE BY field in FILEPOOL FORMAT AUDIT output [160](#)
- authorization
  - effect of DELETE PUBLIC on [94](#)
  - effect of DELETE USER on [93](#)
  - effect of ENROLL PUBLIC on [93](#)
  - effect of FILEPOOL FILELOAD on [127](#)
  - effect of FILEPOOL RELOAD FILES on [127](#)
  - effect of FILEPOOL RESTORE on [121](#)
  - implicit [140](#)
  - in a remote-use environment [235](#)
  - recording within catalog storage group [137](#)
  - using external security managers [161](#)
  - when renaming file spaces [99](#)
  - when transferring ownership [99](#)
- AUTHREQ control statement for FILEPOOL FORMAT AUDIT [149](#), [160](#)
- auto-logging a server [272](#)
- AUTOOCR initialization parameter for CMS [259](#), [273](#)
- AUTOLOG1 user ID [272](#)
- automatic
  - periodic retry of resynchronization
    - definition of [305](#)
    - restarting [324](#)
    - stopping [324](#)
- automatic periodic retry of resynchronization, CRR [388](#), [390](#), [394](#)
- automatically started backups [105](#)
- automatically starting a server [272](#)
- AVS (APPC/VM VTAM Support)
  - used to set up file pool for remote use [233](#)

## B

- backing up
  - control data [104](#)
  - user data [113](#)
- backout required exit [299](#)
- BACKUP ddname [111](#), [269](#), [501](#), [516](#)
- backup files
  - defining or changing [107](#), [506](#)
  - listing contents of [459](#)
  - obsolete [491](#)
  - specifying during generation data to disk. [269](#)
  - staging to tape [115](#)
- BACKUP operator command [105](#), [366](#)
- BACKUP startup parameter
  - description of [342](#)
  - switching from [107](#)
  - switching to [104](#)
- backups at shutdown [104](#)
- backups of the control data
  - at shutdown [66](#)
  - BACKUP operator command [366](#)
  - control minidisk verification during [109](#)
  - creating [107](#)
  - determine current and default file using QUERY DEFBACKUP command [553](#)
  - determining frequency of [106](#)
  - during shutdown [104](#)
  - FILESERV BACKUP command [500](#)
  - recovering from errors of [106](#)
  - special considerations for DASD [106](#)
  - special considerations for tape [106](#)



- backups of the control data (*continued*)
  - started automatically [56](#), [105](#)
  - startup parameters related to [342](#)
  - temporary file used during [367](#)
  - using BACKUP operator command [105](#)
  - using FILEPOOL CONTROL BACKUP administration command [105](#)
  - using FILESERV BACKUP [104](#)
- backups of user storage groups
  - considerations when managed by DFSMS/VM [115](#)
  - creating [113](#), [115](#)
  - FILEPOOL BACKUP command [432](#)
  - FILEPOOL CLEANUP command [437](#)
  - FILEPOOL FILELOAD command [451](#)
  - FILEPOOL LIST BACKUP command [459](#)
  - FILEPOOL RESTORE command [488](#)
  - running concurrently [115](#)
- backups started automatically [105](#)
- backups using FILESERV BACKUP [104](#)
- backups using the BACKUP operator command [105](#)
- backups using the FILEPOOL CONTROL BACKUP administration command [105](#)
- backups, concurrent [115](#)
- backups, remote [113](#)
- BFS (Byte File System)
  - changing amount of space allocated [91](#)
  - checking to see which file spaces exist [92](#)
  - creating [71](#)
  - deleting file spaces [93](#)
  - description of [11](#)
  - determining space assigned [92](#)
  - enrolling [71](#)
  - enrolling public [92](#)
  - establishing clients [80](#)
  - location of, VMSYS [37](#)
  - managing [71](#)
  - MAXCONN value [258](#)
  - moving file spaces [95](#)
  - POSIXOPT SETIDS ALLOW [256](#)
  - renaming file spaces [99](#)
  - restoring user data
    - for a file space [117](#)
    - for user storage group [116](#)
  - startup parameters [339](#)
  - storage use exits [209](#)
  - used with BFS files
    - DELETE USER [408](#)
    - ENABLE [415](#)
    - ENROLL USER [422](#)
    - FILEPOOL DISABLE [443](#)
    - FILEPOOL ENABLE [447](#)
    - FILEPOOL FILELOAD [451](#)
    - FILEPOOL LIST BACKUP [459](#)
    - FILEPOOL LIST MINIDISK [467](#)
    - FILEPOOL RELOAD [476](#)
    - FILEPOOL RENAME [484](#)
    - FILEPOOL RESTORE [488](#)
    - FILEPOOL UNLOAD [495](#)
    - FILESERV REORG [535](#)
    - FORCE [539](#)
    - MODIFY USER [544](#)
    - QUERY DISABLE [555](#)
    - QUERY FILEPOOL AGENT [557](#)
    - QUERY FILEPOOL CONFLICT [563](#), [566](#)
- BFS (Byte File System) (*continued*)
  - used with BFS files (*continued*)
    - QUERY FILEPOOL DISABLE [577](#)
    - QUERY FILEPOOL REPORT [592](#)
    - QUERY LIMITS [655](#)
  - VMSYS [37](#)
- BFS file pools
  - creating BFS file spaces [71](#)
  - deleting file space from [93](#)
  - deleting PUBLIC from [94](#)
  - enrolling PUBLIC in [92](#)
  - enrolling users in [71](#)
  - listing existing file spaces [92](#)
  - listing users enrolled in [92](#)
  - moving a user from [96](#)
  - moving users in [95](#)
- BFS file space
  - listing those that exist [92](#)
- BFS file spaces
  - allocating to users [72](#)
  - changing space allocations of [91](#)
  - deleting [93](#)
  - moving to another storage group [95](#), [96](#)
  - multiple user mode [95](#)
- block I/O files [131](#), [519](#)
- block size
  - of control data backup files [111](#)
  - of log minidisks [515](#)
  - of storage group minidisks [201](#), [517](#)
- blocks, 4KB
  - changing a user's allocation of [91](#)
  - deleting file space allocation of [93](#)
  - giving file spaces allocations of [72](#)
  - giving users allocations of [72](#)
  - maximum number per file pool [685](#)
  - maximum number per file space [685](#)
  - maximum number per storage group minidisk [685](#)
- break tree processing [303](#)
- BUFFSIZ parameter of the DEFNUC macro [49](#), [275](#)
- bypass timed wait interval [325](#)
- Byte File System
  - deleting [93](#)
  - renaming file spaces [99](#)
  - transferring ownership among [99](#)
  - unenrolled
    - renaming [99](#)
- Byte File System (BFS)
  - changing amount of space allocated [91](#)
  - checking to see which file spaces exist [92](#)
  - creating [71](#)
  - deleting file spaces [93](#)
  - description of [11](#)
  - determining space assigned [92](#)
  - enrolling [71](#)
  - enrolling public [92](#)
  - establishing clients [80](#)
  - location of, VMSYS [37](#)
  - managing [71](#)
  - MAXCONN value [258](#)
  - moving file spaces [95](#)
  - POSIXOPT SETIDS ALLOW [256](#)
  - renaming file spaces [99](#)
  - restoring user data
    - for a file space [117](#)

Byte File System (BFS) (*continued*)

- restoring user data (*continued*)
  - for user storage group [116](#)
- startup parameters [339](#)
- storage use exits [209](#)
- used with BFS files
  - DELETE USER [408](#)
  - ENABLE [415](#)
  - ENROLL USER [422](#)
  - FILEPOOL DISABLE [443](#)
  - FILEPOOL ENABLE [447](#)
  - FILEPOOL FILELOAD [451](#)
  - FILEPOOL LIST BACKUP [459](#)
  - FILEPOOL LIST MINIDISK [467](#)
  - FILEPOOL RELOAD [476](#)
  - FILEPOOL RENAME [484](#)
  - FILEPOOL RESTORE [488](#)
  - FILEPOOL UNLOAD [495](#)
  - FILESERV REORG [535](#)
  - FORCE [539](#)
  - MODIFY USER [544](#)
  - QUERY DISABLE [555](#)
  - QUERY FILEPOOL AGENT [557](#)
  - QUERY FILEPOOL CONFLICT [563](#), [566](#)
  - QUERY FILEPOOL DISABLE [577](#)
  - QUERY FILEPOOL REPORT [592](#)
  - QUERY LIMITS [655](#)
- VMSYS [37](#)

**C**

- cache [151](#)
- cache release request [151](#)
- callable services library (CSL)
  - routines
    - DMS2AB [229](#)
    - DMSAAUTH [177](#)
    - DMSOAUTH [177](#)
    - DMSPERM [182](#)
    - DMSSECIT [176](#)
    - DMSSFSEX [209](#)
    - DMSUAUTH [177](#)
    - DMSUDATA [191](#)
  - template files
    - DMS2AB [229](#)
    - DMS5XTP [210](#)
    - DMSJBATP [177](#)
    - DMSJBITP [176](#)
    - DMSJBPTP [182](#)
- catalog
  - indexes [213](#)
  - minidisk
    - determining allocations for [254](#)
    - minimum recommended size of [255](#)
    - placement of [255](#)
  - reorganizing [213](#)
  - storage group
    - adding space to [195](#)
    - description of [14](#)
    - determining when to reorganize [213](#)
    - monitoring logical space in [63](#)
    - monitoring physical storage in [61](#)
    - recording of authorizations within [137](#)
- catalog reorganization, migration considerations [52](#)

- catalog storage group
  - displaying information about [623](#)
  - listing the contents of [521](#)
  - reorganizing [535](#)
  - setting logical size of [514](#)
- catalog utility BFS request [156](#)
- CATBUFFERS startup parameter
  - description of [343](#)
  - recommend setting for, during file pool regeneration [219](#)
  - recommended setting for FILESERV REORG [214](#)
  - setting for, when restoring control data [110](#), [354](#)
  - setting for, when restoring storage groups [118](#)
- change timed wait interval [325](#)
- changing the file pool ID [67](#)
- changing the output file for security auditing [144](#)
- chaudit BFS request [156](#)
- checkpoint accounting record [226](#)
- checkpoint start data
  - accounting records generated for [224](#)
  - definition of [62](#)
- checkpoints [614](#), [629](#)
- chmod BFS request [156](#)
- chown BFS request [157](#)
- CLISTs [308](#)
- CLISTs, NetView [707](#)
- CMSFILES physical saved segment [43](#), [355](#)
- commands
  - administration
    - DATASPACE [396](#)
    - DELETE ADMINISTRATOR [400](#)
    - DELETE LOCK [403](#)
    - DELETE PUBLIC [406](#)
    - DELETE USER [408](#)
    - ENROLL ADMINISTRATOR [417](#)
    - ENROLL PUBLIC [420](#)
    - ENROLL USER [422](#)
    - FILEPOOL BACKUP [432](#)
    - FILEPOOL CLEANUP [437](#)
    - FILEPOOL DISABLE [443](#)
    - FILEPOOL ENABLE [447](#)
    - FILEPOOL FILELOAD [451](#)
    - FILEPOOL FORMAT AUDIT [455](#)
    - FILEPOOL LIST BACKUP [459](#)
    - FILEPOOL LIST MINIDISK [467](#)
    - FILEPOOL MINIDISK [470](#)
    - FILEPOOL RELOAD [476](#)
    - FILEPOOL RENAME [484](#)
    - FILEPOOL RESTORE [488](#)
    - FILEPOOL UNLOAD [495](#)
    - MODIFY USER [544](#)
    - QUERY ACCESSORS [547](#)
    - QUERY DATASPACE [550](#)
    - QUERY FILEPOOL AGENT [557](#)
    - QUERY FILEPOOL CATALOG [560](#)
    - QUERY FILEPOOL CONFLICT [563](#)
    - QUERY FILEPOOL COUNTER [569](#)
    - QUERY FILEPOOL CRR [574](#)
    - QUERY FILEPOOL DISABLE [577](#)
    - QUERY FILEPOOL LOG [583](#)
    - QUERY FILEPOOL MINIDISK [586](#)
    - QUERY FILEPOOL OVERVIEW [589](#)
    - QUERY FILEPOOL REPORT [592](#)
    - QUERY FILEPOOL STATUS [623](#)
    - QUERY FILEPOOL STORGRP [652](#)

commands (*continued*)
 

- administration (*continued*)
  - QUERY LIMITS [655](#)
  - SET THRESHOLD [664](#)
- file pool server
  - FILESERV BACKUP [500](#)
  - FILESERV CRRLOG [502](#)
  - FILESERV DEFAUDIT [504](#)
  - FILESERV DEFBACKUP [506](#)
  - FILESERV DEFCRRLOG [509](#)
  - FILESERV FIXCENT [511](#)
  - FILESERV GENERATE [513](#)
  - FILESERV LIST [521](#)
  - FILESERV LOG [523](#)
  - FILESERV MINIDISK [525](#)
  - FILESERV MOVEUSER [528](#)
  - FILESERV REGENERATE [531](#)
  - FILESERV REORG [535](#)
  - FILESERV START [537](#)
- operator
  - AUDIT [362](#)
  - BACKUP [366](#)
  - CRR ERASE LU [368](#)
  - CRR ERASE LUWID [370](#)
  - CRR QUERY LOG [371](#)
  - CRR QUERY LOGTABLE [373](#)
  - CRR QUERY LU [376](#)
  - CRR QUERY LUWID [382](#)
  - CRR RESUME [388](#)
  - CRR RESYNC [390](#)
  - CRR SUSPEND [394](#)
  - DEFBACKUP [398](#)
  - DISABLE [412](#)
  - ENABLE [415](#)
  - ERASE LUNAME [427](#)
  - ETRACE [429](#)
  - FORCE [539](#)
  - GRANT ADMIN [541](#)
  - ITRACE [542](#)
  - QUERY DEFBACKUP [553](#)
  - QUERY DISABLE [555](#)
  - QUERY LOGTABLE [658](#)
  - QUERY PREPARED [660](#)
  - REVOKE ADMIN [663](#)
  - STOP [666](#)

commands used in file pool administration [22](#), [113](#), [115](#), [119](#)

comments
 

- specifying in control statement files [514](#)
- specifying in DMSPARMS files [340](#)
- specifying in POOLDEF files [514](#)

common denominator minidisks [197](#)

communication
 

- IUCV [258](#)

communications directory [234](#)

complete audit [144](#)

concurrent backups [115](#)

configuring SFS server to use data spaces [242](#)

connections to file pools
 

- allowed by ENROLL PUBLIC [92](#)
- authorizing users for [71](#)
- description of [8](#)
- effect of DELETE PUBLIC on [94](#)
- effect of FORCE command on [540](#)

connections to file pools (*continued*)
 

- estimating number required [258](#)
- limiting scope with startup parameters [347](#)
- severing at shutdown [667](#)
- when established [75](#)

console, operator [19](#)

control data
 

- backing up [107](#)
- damage of [57](#)
- description of [13](#)
- restoring [110](#)
- verification of [668](#)

control data backups
 

- at shutdown [66](#)
- BACKUP operator command [366](#)
- control minidisk verification during [109](#)
- creating [107](#)
- determine current and default file using QUERY DEFBACKUP command [553](#)
- determining frequency of [106](#)
- during shutdown [104](#)
- FILESERV BACKUP command [500](#)
- recovering from errors of [106](#)
- special considerations for DASD [106](#)
- special considerations for tape [106](#)
- started automatically [56](#), [105](#)
- startup parameters related to [342](#)
- temporary file used during [367](#)
- using BACKUP operator command [105](#)
- using FILEPOOL CONTROL BACKUP administration command [105](#)
- using FILESERV BACKUP [104](#)

CONTROL ddname [131](#), [268](#), [515](#)

control minidisk
 

- defining virtual device number of [515](#)
- description of [14](#)
- determining allocation for [250](#)
- effect of FILESERV MINIDISK on [206](#)
- increasing the size of [217](#)
- maximum size of [251](#), [685](#)
- minimum recommended size of [251](#)
- placement of [251](#)
- recovery from inconsistencies on [109](#)
- replacing [129](#)
- verification of [668](#)

control statements
 

- for FILEPOOL FORMAT AUDIT [149](#)
- for FILESERV GENERATE [514](#)
- for FILESERV MINIDISK [525](#)
- for FILESERV MINIDISK or FILEPOOL MINIDISK [202](#)

controlling access to a server machine [139](#)

Conversational Monitor System (CMS)
 

- accounting facility
  - customizing file pool server user accounting records [229](#)
  - DMS2AB exit routine [229](#)
- event control block
  - error handling [173](#)
  - format [173](#)
  - posting by security adapter [173](#)
- exits
  - accounting records, file pool server [229](#)
  - DMS2AB routine [229](#)
  - DMSJNE routine [238](#)

## Conversational Monitor System (CMS) *(continued)*

- exits *(continued)*
  - local IDs for remote SFS users [238](#)
- macros
  - DMSJNEPL [239](#)
  - EXITBUFF [209](#)
- nickname resolution for SFS commands [239](#)
- Shared File System (SFS)
  - definition of [10](#)
- work unit [14](#)

## Coordinated Resource Recovery (CRR)

- accounting facility [223](#)
- ACCT operand [258](#)
- administration
  - automatic periodic retry of resynchronization, restart [324](#)
  - automatic periodic retry of resynchronization, stopping [324](#)
  - bypass timed wait interval [325](#)
  - change timed wait interval [325](#)
  - CRR file pool minidisks, replacing [327](#)
  - CRR log minidisk [306](#)
  - CRR log minidisk, replace both [328](#)
  - CRR log minidisk, replace one [328](#)
  - CRR log name table, freeing space [323](#)
  - CRR recovery server [317](#), [318](#), [320](#)
  - free space in CRR log name table [323](#)
  - managing performance [327](#)
  - performance managing [327](#)
  - remote administration [326](#)
  - replace both CRR log minidisks [328](#)
  - replace minidisks in CRR file pool [327](#)
  - replace one CRR log minidisk [328](#)
  - restart automatic periodic retry of resynchronization [324](#)
  - stop automatic periodic retry of resynchronization [324](#)
  - tasks [315](#)
  - timed wait interval, bypass [325](#)
  - timed wait interval, change [325](#)
- administration and operation [288](#)
- administration commands
  - ENROLL ADMINISTRATOR [417](#)
  - FILEPOOL FORMAT AUDIT [455](#)
  - QUERY FILEPOOL AGENT [557](#)
  - QUERY FILEPOOL CATALOG [560](#)
  - QUERY FILEPOOL CRR [574](#)
  - QUERY FILEPOOL OVERVIEW [589](#)
  - QUERY FILEPOOL REPORT [592](#)
  - QUERY FILEPOOL STATUS [623](#)
- allocation and sync point tree [302](#)
- application development [288](#)
- auditing security [143](#)
- automatic periodic retry of resynchronization [305](#), [388](#), [390](#), [394](#)
- backout required exit [299](#)
- banking example [287](#)
- break tree processing [303](#)
- CLISTs [308](#)
- coordinated transaction [355](#)
- coordination function [296](#)
- data restoration considerations [308](#)
- distributed applications [292](#)
- distributed environment, management [308](#)

## Coordinated Resource Recovery (CRR) *(continued)*

- distributed resource [289](#)
- end of work unit exit [299](#)
- exchange log names [352](#), [395](#)
- exit processing [312](#)
- exits and registration [297](#)
- file pool server machine commands
  - FILESERV CRRLOG [502](#)
  - FILESERV DEFAUDIT [504](#)
  - FILESERV DEFCRRLOG [509](#)
  - FILESERV FIXCENT [511](#)
  - FILESERV GENERATE [513](#)
  - FILESERV LOG [523](#)
  - FILESERV START [537](#)
- file pools
  - committing or rolling back [15](#)
  - generating [247](#)
  - server machine [12](#), [16](#)
- forcing prepared work [57](#)
- forward recovery [308](#)
- function [294](#)
- functional flow [311](#)
- heuristic damage [379](#), [380](#), [385](#), [386](#), [390](#), [393](#)
- heuristic decision [390–392](#), [539](#), [540](#)
- heuristic response [390](#), [392](#)
- index [377](#), [380](#), [383](#), [386](#), [390](#)
- last agent optimization [304](#)
- limp mode [296](#)
- log name table [306](#)
- log ring [307](#)
- log size [254](#), [685](#)
- logging function [306](#)
- logs [306](#)
- LU 6.2 [353](#)
- LUID [310](#)
- LUID instance [355](#)
- management in distributed environment [308](#)
- MAXDISKS value [249](#)
- MAXUSERS value [249](#)
- MDISK statement [260](#)
- Netview Clists
  - AOPMCRRM FPCLST [710](#)
  - CRRM001 FPPANEL [714](#)
  - CRRM002 FPPANEL [714](#)
  - CRRM003 FPPANEL [715](#)
  - CRRM004 FPPANEL [715](#)
  - CRRMSGR DNCLST [711](#)
  - CRRMSGR EXEC [713](#)
  - CRROPF APPEND [716](#)
  - MATDNCRR APPEND [716](#)
  - MATFPCRR APPEND [717](#)
- operator commands
  - AUDIT [362](#)
  - CRR ERASE LU [368](#)
  - CRR ERASE LUID [370](#)
  - CRR QUERY LOG [371](#)
  - CRR QUERY LOGTABLE [373](#)
  - CRR QUERY LU [376](#)
  - CRR QUERY LUID [382](#)
  - CRR RESUME [388](#)
  - CRR RESYNC [390](#)
  - CRR SUSPEND [394](#)
  - ERASE LUNAME [427](#)
  - ETRACE [429](#)

Coordinated Resource Recovery (CRR) (*continued*)

- operator commands (*continued*)
  - FORCE [539](#)
  - GRANT ADMIN [541](#)
  - ITRACE [542](#)
  - QUERY LOGTABLE [658](#)
  - QUERY PREPARED [660](#)
  - REVOKE ADMIN [663](#)
  - STOP [666](#)
- overview [287](#)
- post- installation [37](#)
- problem management [330](#)
- product participation [288](#), [307](#)
- protected conversations [289](#)
- protected resource [289–291](#)
- recovery server [296](#)
- recovery token [377](#), [380](#), [382](#), [383](#), [386](#), [660](#), [661](#)
- recovery TPN [377](#), [380](#), [383](#), [386](#)
- registration and exits [297](#)
- resource adapter [296](#)
- resynchronization
  - function [305](#)
  - recovery [305](#)
- resynchronization function [305](#), [390](#)
- resynchronization roles [380](#), [386](#)
- resynchronization states [380](#), [386](#)
- security, auditing [143](#)
- server machine (define) [257](#)
- SFS example [288](#)
- SFS participation
  - administration and operator intervention [279](#)
  - commands [279](#)
  - overview [279](#)
- SFS participation in CRR
  - ERASE LUNAME operator command [427](#)
  - FORCE PREPARED operator command [539](#)
  - QUERY PREPARED operator command [660](#)
- SNA LU 6.2 sync point architecture [289](#)
- startup parameters
  - CRR [348](#)
  - LUNAME [352](#)
  - NOCCR [348](#)
  - NOLUNAME [352](#)
  - RESYNCINTERVAL [354](#)
- startup parameters related to [348](#)
- sync point exits
  - coordination [298](#)
  - postcoordination [298](#)
  - precoordination [298](#)
- sync point processing [298](#)
- sync point roles [378](#), [384](#)
- sync point states [378](#), [384](#)
- sync point tree [299](#), [378](#), [384](#), [390](#), [391](#), [393](#)
- timed wait [354](#), [378](#), [379](#), [384](#), [385](#), [388](#), [389](#), [394](#), [395](#)
- TPN X'06F2' [376](#), [378](#), [384](#), [395](#)
- transaction program [309](#)
- transaction tag [377](#), [379](#), [383](#), [385](#), [457](#), [661](#)
- two-phase commit [355](#), [378–380](#), [384–386](#), [521](#), [529](#), [533](#), [536](#), [537](#), [661](#)
- two-phase commit protocol [299](#)
- upstream shoulder tap [395](#)
- user data minidisks [255](#)

coordinated transaction [355](#)

coordination

coordination (*continued*)

- exit, CRR [298](#)
- function, CRR [296](#)
- copying sample files [273](#)
- counter data
  - in the QUERY FILEPOOL COUNTER command [570](#)
  - in the QUERY FILEPOOL REPORT command [607](#)
  - in the QUERY FILEPOOL STATUS command [628](#)
- counter information, CRR [628](#)
- CP SEND command [56](#)
- create authority check, SFS [174](#)
- creating a file pool [247](#)
- creating a startup parameter file [263](#)
- CRR file pool minidisks, replacing [327](#)
- CRR log minidisk, replace one [328](#)
- CRR log minidisks, replace both [328](#)
- CRR log name table, freeing space [323](#)
- CRR log name table, management [321](#)
- CRR logical unit of work [310](#)
- CRR QUERY LOGTABLE [373](#)
- CRR recovery server
  - designate an alternate [318](#)
  - generating [317](#)
  - MAXDISKS value [249](#)
  - MAXUSERS value [249](#)
  - remove designation of alternate [320](#)
  - user data minidisks [255](#)
- CRR remote administration [326](#)
- CRR startup parameter [348](#)
- CRR startup parameter, as specified for VMYSR [43](#)
- CRR1 ddname [518](#)
- CRRM001 FPPANEL [714](#)
- CRRM002 FPPANEL [714](#)
- CRRM003 FPPANEL [715](#)
- CRRM004 FPPANEL [715](#)
- CRRMSGR DNCLST [711](#)
- CRRMSGR EXEC [713](#)
- CRROPF APPEND [716](#)
- CSL (callable services library) routines [28](#), [151](#)
- CTLBUFFERS startup parameter [343](#)

**D**

- DAC component of server [351](#), [429](#)
- damaged
  - control data [57](#)
- DASD Dump Restore (DDR) Service Program, using with file pools [135](#)
- DASD Dump/Restore (DDR) program [135](#)
- DASD failures
  - recovering from
    - with no good log [111](#)
    - with one good log [111](#)
- DASD storage
  - accounting for use of [227](#), [229](#)
  - adding to the file pool [195](#)
  - allocating to file spaces [72](#)
  - changing amounts allocated to users [91](#)
  - directing control data backups to [269](#)
  - increasing file pool limit of [217](#)
  - managing [195](#), [212](#)
  - monitoring consumption of [60](#)
  - querying a user's allocation [92](#)
  - removing from file pool [206](#)

DASD storage (*continued*)  
 setting warning threshold for [208](#)

data  
 restoration considerations, CRR [308](#)

data space  
 configuring SFS server to use [242](#)  
 definition of [10](#)  
 determining status of [550](#)  
 dumping [344](#)  
 ensuring optimal server performance [245](#)  
 exploitation, migration [52](#)  
 extended address volume minidisk [244](#)  
 FBA minidisk restrictions [244](#)  
 finding candidates for [241](#)  
 making existing directories eligible for [245](#)  
 making existing minidisks eligible for [244](#)  
 monitoring usage [245](#)  
 moving system minidisks into [244](#)  
 selecting a file pool [242](#)  
 using with SFS repository servers [241](#)

DATASPACE command  
 instructions for using with directories [245](#)  
 instructions for using with minidisks [244](#)  
 overview of [26](#)  
 reference information for [396](#)

DATE control statement for FILEPOOL FORMAT AUDIT [150](#)

DATE field in FILEPOOL FORMAT AUDIT output [160](#)

DDNAME=AUDIT control statement [505](#)

DDNAME=BACKUP control statement [269](#), [507](#), [516](#)

DDNAME=CONTROL control statement [131](#), [515](#)

DDNAME=CRR1 control statement [518](#)

DDNAME=CRR2 control statement [518](#)

DDNAME=LOG1 control statement [131](#), [515](#)

DDNAME=LOG2 control statement [131](#), [515](#)

DDNAME=MDKnnnnn control statement [202](#), [517](#)

ddnames used by FILEPOOL FORMAT AUDIT  
 INPUT [149](#)  
 INPUTCTL [149](#)  
 OUTPUT [149](#)

DDR (DASD Dump Restore) Service Program, using with file pools [135](#)

DDR (DASD Dump/Restore program) [135](#)

debugging aids  
 brief overview of [29](#)  
 ETRACE operator command [429](#)  
 external tracing [351](#)  
 FILESERV LIST command [521](#)  
 internal tracing [351](#)  
 ITRACE operator command [542](#)  
 mini-dumps [29](#)

dedicated maintenance mode  
 accounting facility [224](#)  
 description of [19](#)  
 specifying startup parameters for [339](#)  
 tracing server execution during [351](#)

DEFBACKUP operator command [398](#)

defining a file pool [247](#)

DEFNUC macro [49](#), [275](#)

DELETE ADMINISTRATOR command  
 instructions for using [142](#)  
 overview of [26](#)  
 reference information for [400](#)

DELETE LOCK command [403](#)

DELETE PUBLIC command (*continued*)  
 instructions for using [94](#), [406](#)  
 overview of [26](#)

DELETE USER command  
 instructions for using [93](#)  
 overview of [26](#)  
 reference information for [408](#)

deleting  
 administrator [142](#)  
 file pool [277](#)  
 file spaces [93](#)

deleting an administrator [400](#)

designating an alternate CRR recovery server [318](#)

device errors  
 recovering from  
 with no good log [111](#)  
 with one good log [111](#)

DFSMS COPY command [134](#)

DFSMS startup parameter [349](#)

DFSMS/VM  
 considerations for  
 back up of storage group [115](#)  
 restore of individual files in migrated status [127](#)  
 restore of storage group [123](#)  
 using for storage management [208](#)

diagnostic aids  
 brief overview of [29](#)  
 ETRACE operator command [429](#)  
 external tracing [351](#)  
 FILESERV LIST command [521](#)  
 internal tracing [351](#)  
 ITRACE operator command [542](#)  
 mini-dumps [29](#)

directory  
 definition of [10](#)  
 determining who is accessing [547](#)  
 effect of FILEPOOL RESTORE on [120](#)  
 making data space eligible [245](#)  
 querying data space eligibility of [550](#)  
 renaming user's file spaces [99](#)  
 transferring ownership of [99](#)  
 used with data spaces [241](#)

disable locks  
 checking status of [555](#)  
 creating [412](#)  
 deleting [415](#)  
 during FILEPOOL RESTORE [120](#)

DISABLE operator command  
 reference information for [412](#)  
 use in removing file pool minidisks [206](#)

disconnected mode [21](#)

disk failures  
 recovering from  
 with no good log [111](#)  
 with one good log [111](#)

disk files, using for backups [108](#)

disk labels for file pool minidisks [131](#), [219](#)

disk space  
 accounting for use of [227](#), [229](#)  
 adding to the file pool [195](#)  
 allocating to file spaces [72](#)  
 changing amounts allocated to users [91](#)  
 directing control data backups to [269](#)  
 increasing file pool limit of [217](#)



- disk space (*continued*)
  - managing [195](#), [212](#)
  - monitoring consumption of [60](#)
  - querying a user's allocation [92](#)
  - removing from file pool [206](#)
  - setting warning threshold for [208](#)
- distributed
  - application [292](#)
  - environment, CRR management [308](#)
  - resource [289](#)
- DMS1151E message [59](#)
- DMS1174E message [75](#)
- DMS2AB exit routine
  - CSL template file [229](#)
  - function [229](#)
  - making your exit routine available [230](#)
  - return code [230](#)
  - when called by CMS [229](#)
  - writing your own routine [229](#)
- DMS2AB TEMPLATE file [229](#)
- DMS3029I message [66](#)
- DMS3216E message [109](#)
- DMS3217E message [109](#)
- DMS3294I message [106](#)
- DMS3500I message [120](#)
- DMS3620I message [125](#), [127](#)
- DMS3701I message [221](#)
- DMS3913E message [217](#)
- DMS3922I message [200](#), [201](#), [527](#)
- DMS446R message [270](#)
- DMS5XX module (DMSSFSEX exit routine)
  - functions [210](#)
  - modifying or replacing [210](#)
- DMS5XXTP TEMPLATE file [210](#)
- DMSAAUTH exit routine
  - CSL template file [177](#)
  - function [177](#)
  - making replacement routine available [188](#)
  - mapping to RACROUTE requests [698](#)
  - parameters [178](#)
  - processing overview [694](#)
  - return codes [178](#)
  - user data structure [695](#)
- DMSESM PROFILE
  - contents of [164](#)
  - customizing [164](#)
  - file pool requests to be reviewed [168](#)
  - tailoring [164](#)
  - types of calls to be reviewed [165](#)
- DMSIBM POOLDEF [513](#)
- DMSIBM POOLDEF file [268](#), [513](#)
- DMSJBATP TEMPLATE file [177](#)
- DMSJBITP TEMPLATE file [176](#)
- DMSJBPTP TEMPLATE file [182](#)
- DMSJNE exit routine
  - input registers [239](#)
  - installing [240](#)
  - return codes [240](#)
  - when called by CMS [239](#)
- DMSJNEPL macro [239](#)
- DMSNGP ASSEMBLE file [49](#), [275](#)
- DMSOAUTH exit routine
  - CSL template file [177](#)
  - function [177](#)
- DMSOAUTH exit routine (*continued*)
  - making replacement routine available [188](#)
  - mapping to RACROUTE requests [698](#)
  - parameters [178](#)
  - processing overview [691](#)
  - return codes [178](#)
- DMSPARMS file
  - creating during generation [265](#)
  - definition of [20](#)
  - for VMSYS, VMPSFS, VMSYSU, and VMSYSR [41](#)
  - specifying startup parameters within [339](#)
- DMSSECIT exit routine
  - CSL template file [176](#)
  - function [176](#)
  - mapping to RPIUCMS calls [697](#)
  - parameters [176](#)
  - return codes [176](#)
- DMSSFSEX exit routine
  - activating sample functions [210](#)
  - CSL template file [210](#)
  - default functions [210](#)
  - general data buffer, SFS [209](#)
  - making your exit routine available [211](#)
  - marking users for rollback [211](#)
  - modifying or replacing [210](#)
  - parameters [210](#)
  - return codes [210](#)
  - SFS file space usage exit
    - default action [210](#)
    - function [209](#)
    - sample (dormant) function [210](#)
    - when called by SFS server [209](#)
  - SFS user storage group full exit
    - default action [210](#)
    - function [209](#)
    - sample (dormant) function [210](#)
    - when called by SFS server [209](#)
  - special considerations for writing exit routine [211](#)
  - when called by SFS server [209](#)
- DMSUAUTH exit routine
  - CSL template file [177](#)
  - function [177](#)
  - making replacement routine available [188](#)
  - mapping to RACROUTE requests [699](#)
  - parameters [178](#)
  - processing overview [694](#)
  - return codes [178](#)
  - user data structure [695](#)
- DMSUDATA routine [191](#)
- DMSWRACC - SFS Write Accounting Record routine [225](#)
- DMSZNGP ASSEMBLE file [49](#), [275](#)
- DUMP startup parameter
  - controlling what a server dumps [29](#)
  - description of [344](#)
- DUMPALL control statement for FILEPOOL FORMAT AUDIT
  - [150](#), [344](#)
  - dumps of data spaces [344](#)
  - dumps of server storage [344](#)

## E

- ENABLE operator command [128](#), [415](#)
- enabling a file space [415](#)
- enabling a storage group [120](#), [206](#), [415](#)

- end of work unit exit [299](#)
- ENROLL ADMINISTRATOR command
  - instructions for using [141](#)
  - overview of [26](#)
  - reference information for [417](#)
- ENROLL PUBLIC command
  - instructions for using [92](#)
  - overview of [26](#)
  - reference information for [420](#)
- ENROLL USER command
  - instructions for using [76](#)
  - overview of [26](#)
  - reference information for [422](#)
- enrolling
  - file pool administrators [141](#)
  - PUBLIC
    - how to [92](#)
    - remote use considerations for [234](#)
  - remote users [236](#)
  - users [71](#)
- ERASE LUNAME operator command [427](#)
- erased alias [121](#)
- errors, restart recovery processing after [56](#)
- ESECURITY startup parameter
  - description of [350](#)
  - effect on administrator powers [162](#)
- ESM (external security manager)
  - administration considerations for [162](#)
  - auditing security with [143](#)
  - brief overview of [29](#)
  - connect considerations for [162](#)
  - SFS interface
    - administrator command authorization processing [692](#)
    - authorization processing [172](#)
    - command authorization checking routine [177](#)
    - create authority check [174](#)
    - DMSAAUTH exit routine [177](#)
    - DMSOAUTH exit routine [177](#)
    - DMSSECIT exit routine [176](#)
    - DMSUAUTH exit routine [177](#)
    - ESM initialization [172](#)
    - ESM program check routine [176](#)
    - ESM requirements [169](#)
    - event control blocks [173](#)
    - exit routines, IBM-supplied [171](#), [172](#)
    - exit routines, writing replacements [176](#)
    - function [172](#)
    - generic profile query [174](#)
    - initialization/termination routine [176](#)
    - maintaining list of authorizations [170](#)
    - mapping SFS authorization calls to RACROUTE requests [689](#)
    - object authorization checking routine [177](#)
    - object authorization processing [694](#)
    - operator command authorization checking routine [177](#)
    - operator command authorization processing [689](#)
    - overview [171](#)
    - password protection for objects [192](#)
    - profile rename [175](#)
    - providing user or administrator interface [170](#)
    - read authority check [174](#)
    - rebuilt commands and CSL routines [692](#)
- ESM (external security manager) (*continued*)
  - SFS interface (*continued*)
    - RPIUCMS module [172](#)
    - security adapter, communications with security machine [172](#)
    - security adapter, loading into server [172](#)
    - types of authorization checks [174](#)
    - user data structure [695](#)
    - user data support [189](#)
    - what authority query [174](#)
    - write authority check [174](#)
  - startup parameters related to [350](#)
  - user data support [163](#)
  - using the DMSESM PROFILE [164](#)
- ETRACE operator command [30](#), [429](#)
- ETRACE startup parameter [22](#), [29](#), [351](#)
- event control block (ECB)
  - error handling [173](#)
  - format [173](#)
  - posting by security adapter [173](#)
- exchange log names, CRR [352](#), [395](#)
- exclusive disable locks
  - checking status of [555](#)
  - creating [412](#)
  - deleting [415](#)
  - during FILEPOOL RESTORE [120](#)
- EXITBUFF macro [209](#)
- exits
  - accounting
    - file pool [229](#)
  - CMS
    - DMS2AB routine [229](#)
    - DMSJNE routine [238](#)
    - file pool accounting [229](#)
    - local IDs for remote SFS users [238](#)
  - DMS2AB routine
    - CSL template file [229](#)
    - function [229](#)
    - making your exit routine available [230](#)
    - return code [230](#)
    - when called by CMS [229](#)
    - writing your own routine [229](#)
  - DMSAAUTH routine
    - CSL template file [177](#)
    - function [177](#)
    - making replacement routine available [188](#)
    - mapping to RACROUTE requests [698](#)
    - parameters [178](#)
    - processing overview [694](#)
    - return codes [178](#)
    - user data structure [695](#)
  - DMSJNE routine
    - function [239](#)
    - input registers [239](#)
    - installing [240](#)
    - return codes [240](#)
    - when called by CMS [239](#)
  - DMSOAUTH routine
    - CSL template file [177](#)
    - function [177](#)
    - making replacement routine available [188](#)
    - mapping to RACROUTE requests [698](#)
    - parameters [178](#)
    - processing overview [691](#)



exits (continued)

- DMSOAUTH routine (continued)
  - return codes [178](#)
- DMSSECIT routine
  - CSL template file [176](#)
  - function [176](#)
  - parameters [176](#)
  - return codes [176](#)
- DMSSFSEX routine
  - activating sample functions [210](#)
  - CSL template file [210](#)
  - default functions [210](#)
  - general data buffer, SFS [209](#)
  - making your exit routine available [211](#)
  - marking users for rollback [211](#)
  - modifying or replacing [210](#)
  - parameters [210](#)
  - return codes [210](#)
  - SFS file space usage exit [209](#)
  - SFS user storage group full exit [209](#)
  - special considerations for writing exit routine [211](#)
- DMSUAUTH routine
  - CSL template file [177](#)
  - function [177](#)
  - making replacement routine available [188](#)
  - mapping to RACROUTE requests [699](#)
  - parameters [178](#)
  - processing overview [694](#)
  - return codes [178](#)
  - user data structure [695](#)
- external security manager
  - SFS [171](#), [172](#)
- installation-wide
  - accounting, file pool [229](#)
  - BFS authorization checking [182](#)
  - DMS2AB routine [229](#)
  - DMSAAUTH routine [177](#)
  - DMSJNE routine [238](#)
  - DMSOAUTH routine [177](#)
  - DMSSECIT routine [176](#)
  - DMSSFSEX routine [209](#)
  - DMSUAUTH routine [177](#)
  - ESM initialization/termination [176](#)
  - ESM program check [176](#)
  - file pool accounting records [229](#)
  - local IDs for remote SFS users [238](#)
  - SFS command authorization checking [177](#)
  - SFS file space usage [209](#)
  - SFS object authorization checking [177](#)
  - SFS operator command authorization checking [177](#)
  - SFS storage use [209](#)
  - SFS user storage group full [209](#)
- SFS server
  - command authorization checking [177](#)
  - DMSAAUTH routine [177](#)
  - DMSOAUTH routine [177](#)
  - DMSPERM routine [182](#)
  - DMSSECIT routine [176](#)
  - DMSSFSEX routine [209](#)
  - DMSUAUTH routine [177](#)
  - ESM initialization/termination [176](#)
  - ESM program check [176](#)
  - file space usage [209](#)
  - object authorization checking [177](#)

exits (continued)

- SFS server (continued)
  - operator command authorization checking [177](#)
  - special considerations [687](#)
  - storage use [209](#)
  - user storage group full [209](#)
- exits and registration, CRR [297](#)
- extended address volume minidisk [244](#)
- external security manager (ESM)
  - administration considerations for [162](#)
  - auditing security with [143](#)
  - brief overview of [29](#)
  - connect considerations for [162](#)
- SFS interface
  - administrator command authorization processing [692](#)
  - authorization processing [172](#)
  - command authorization checking routine [177](#)
  - create authority check [174](#)
  - DMSAAUTH exit routine [177](#)
  - DMSOAUTH exit routine [177](#)
  - DMSSECIT exit routine [176](#)
  - DMSUAUTH exit routine [177](#)
  - ESM initialization [172](#)
  - ESM program check routine [176](#)
  - ESM requirements [169](#)
  - event control blocks [173](#)
  - exit routines, IBM-supplied [171](#), [172](#)
  - exit routines, writing replacements [176](#)
  - function [172](#)
  - generic profile query [174](#)
  - initialization/termination routine [176](#)
  - maintaining list of authorizations [170](#)
  - mapping SFS authorization calls to RACROUTE requests [689](#)
  - object authorization checking routine [177](#)
  - object authorization processing [694](#)
  - operator command authorization checking routine [177](#)
  - operator command authorization processing [689](#)
  - overview [171](#)
  - password protection for objects [192](#)
  - profile rename [175](#)
  - providing user or administrator interface [170](#)
  - read authority check [174](#)
  - rebuilt commands and CSL routines [692](#)
  - RPIUCMS module [172](#)
  - security adapter, communications with security machine [172](#)
  - security adapter, loading into server [172](#)
  - types of authorization checks [174](#)
  - user data structure [695](#)
  - user data support [189](#)
  - what authority query [174](#)
  - write authority check [174](#)
- startup parameters related to [350](#)
- user data support [163](#)
- using the DMSESM PROFILE [164](#)

**F**

- FBA minidisk restrictions [244](#)
- FIFO startup parameter
  - description of [345](#)

- file cache size
  - changing [49, 275](#)
  - default setting of [49, 275](#)
- file mode A
  - considerations for [73](#)
  - defining defaults for users [74](#)
  - in a remote file pool [236](#)
  - moving contents to an SFS directory [78](#)
- file pool administration authority
  - DELETE ADMINISTRATOR command [400](#)
  - deleting [142](#)
  - description of [22, 25, 139](#)
  - ENROLL ADMIN command [417](#)
  - for VMSYS, VMPSFS, VMSYSU, and VMSYSR [42](#)
  - GRANT ADMIN operator command [541](#)
  - granting [141](#)
  - granting by ADMIN startup parameter [342](#)
  - granting permanently [141](#)
  - internal recording of [138](#)
  - REVOKE ADMIN operator command [663](#)
  - when ESECURITY is in effect [162](#)
- file pool administration commands [25, 26, 139](#)
- file pool administration machine
  - defining [261](#)
  - definition of [25](#)
  - MAXCONN setting for [142](#)
  - work minidisk for [142](#)
- file pool catalogs
  - description of [14](#)
- file pool identifier (file pool ID)
  - changing [67](#)
  - description of [6](#)
  - naming restrictions for [344](#)
  - of the system file pool [37](#)
  - setting defaults for users [79](#)
  - startup parameter for [344](#)
- file pool request codes [151, 156](#)
- file pool server machine
  - controlling access to [139](#)
  - description of [6, 18](#)
  - IBM-supplied [18](#)
  - VMSESRV [18](#)
  - VMSESRV [18](#)
  - VMSESRV [18](#)
  - VMSESRV [18](#)
- file pool server machine commands
  - FILESERV BACKUP [500](#)
  - FILESERV CRRLOG [502](#)
  - FILESERV DEFAUDIT [504](#)
  - FILESERV DEFBACKUP [506](#)
  - FILESERV DEFCCRLOG [509](#)
  - FILESERV FIXCENT [511](#)
  - FILESERV GENERATE [513](#)
  - FILESERV LIST [521](#)
  - FILESERV LOG [523](#)
  - FILESERV MINIDISK [525](#)
  - FILESERV MOVEUSER [528](#)
  - FILESERV REGENERATE [531](#)
  - FILESERV REORG [535](#)
  - FILESERV START [537](#)
- file pool server operator commands [22](#)
- file pool startup parameters, migration [51](#)
- file pools
  - adding space to [195](#)
- file pools (*continued*)
  - auditing security of [143](#)
  - backing up [115](#)
  - changing ID of [67](#)
  - deleting [277](#)
  - description of [6](#)
  - generating [247, 275](#)
  - IBM-supplied [18](#)
  - making available to other processors [233](#)
  - maximums [685](#)
  - monitoring physical space in [64](#)
  - monitoring status of [56](#)
  - moving [204](#)
  - regenerating [217](#)
  - remote use considerations for [233](#)
  - removing space from [206](#)
  - reorganizing the catalogs of [213](#)
  - replacing minidisks of [129](#)
  - restoring [129](#)
  - restoring by generating again [127](#)
  - restoring in increments [122](#)
  - restricting access from other processors [236](#)
  - restricting access to [69](#)
  - setting space warning threshold for [208](#)
  - starting multiple user access to [55](#)
  - stopping access to [66](#)
  - VMPSFS [18](#)
  - VMSYS [18](#)
  - VMSYSR [18](#)
  - VMSYSU [18](#)
- file space usage exit, SFS
  - default function [210](#)
  - sample (dormant) function [210](#)
  - when called by SFS server [209](#)
- file spaces
  - disabling [412](#)
  - enabling [415](#)
  - maximum number of [685](#)
  - maximum number of 4KB blocks [685](#)
  - requesting a threshold warning message [664](#)
- FILEDEF command
  - as used for FILEPOOL BACKUP [113](#)
  - as used for FILEPOOL FILELOAD [124](#)
  - as used for FILEPOOL RELOAD [116, 126](#)
  - as used for FILEPOOL RESTORE [119](#)
  - as used for FILEPOOL UNLOAD [112](#)
  - for FILEPOOL FORMAT AUDIT control statement file [149](#)
  - for FILEPOOL FORMAT AUDIT input file [146](#)
  - for FILEPOOL FORMAT AUDIT output file [146](#)
  - for restoring control data [110](#)
  - for the FILESERV REORG temporary file [214](#)
  - required during FILESERV REGENERATE processing [214, 220](#)
- FILEPOOL BACKUP command
  - instructions for using [113](#)
  - overview of [26](#)
  - reference information for [432](#)
- FILEPOOL CLEANUP command
  - overview of [26](#)
  - reference information for [437](#)
  - using during filepool restore [120](#)
- FILEPOOL CONTROL BACKUP command
  - instructions for using [105](#)
  - overview of [26](#)

FILEPOOL CONTROL BACKUP command (*continued*)  
     reference information for [439](#)  
 FILEPOOL DISABLE command  
     overview of [26](#)  
     reference information for [443](#), [495](#)  
 FILEPOOL ENABLE command  
     overview of [26](#)  
     reference information for [447](#)  
 FILEPOOL FILELOAD command  
     instructions for using [123](#)  
     listing files on corrupted minidisk for input to [123](#)  
     overview of [26](#)  
     reference information for [451](#)  
     restoring aliases and authorizations [127](#)  
 FILEPOOL FORMAT AUDIT command  
     instructions for using [146](#)  
     keyword control statements for [149](#)  
     output format of [160](#)  
     overview of [27](#)  
     reference information for [455](#)  
     selectivity prompts for [146](#)  
     using control file for [149](#)  
 FILEPOOL LIST BACKUP command  
     overview of [27](#)  
     reference information for [459](#)  
     using with FILEPOOL BACKUP command [436](#)  
 FILEPOOL LIST MINIDISK command  
     instructions for using [123](#)  
     overview of [27](#)  
     reference information for [467](#)  
 FILEPOOL MINIDISK command  
     instructions for using [196](#)  
     overview of [27](#)  
     recovering from errors of [200](#)  
     reference information for [470](#)  
 FILEPOOL operand of IPL control statement  
     generating [513](#)  
     using to define default file mode A [74](#)  
 FILEPOOL RELOAD command  
     reference information for [476](#)  
 FILEPOOL RELOAD FILES command  
     restoring SFS aliases and authorizations [127](#)  
 FILEPOOL RENAME command  
     instructions for using [99](#)  
     overview of [27](#)  
     reference information for [484](#)  
 FILEPOOL RESTORE command  
     instructions for using [119](#)  
     overview of [27](#)  
     reference information for [488](#)  
 FILEPOOL UNLOAD command  
     instructions for using [112](#)  
 FILEPOOLID startup parameter  
     description of [344](#)  
     using to restrict user access [69](#)  
 FILEREQ control statement for FILEPOOL FORMAT AUDIT  
[151](#)  
 files  
     effect of FILEPOOL RESTORE on [120](#)  
     maximums related to [685](#)  
     renaming file spaces [99](#)  
     shadow copies of [685](#)  
     specifying for control data backups [108](#)  
     transferring ownership of [99](#)  
 FILESERV BACKUP command  
     instructions for using [104](#)  
     reference information for [500](#)  
 FILESERV commands  
     descriptions of [19](#)  
     specifying startup parameters for [339](#)  
     used when generating a file pool [263](#)  
 FILESERV DEFAUDIT command  
     instructions for using [144](#)  
     reference information for [504](#)  
 FILESERV DEFBACKUP command  
     instructions for using [108](#)  
     reference information for [506](#)  
 FILESERV DEFERRLOG command  
     reference information for [509](#)  
     using when replacing CRR log minidisks [328](#)  
 FILESERV FIXCENT command  
     reference information for [511](#)  
 FILESERV GENERATE command  
     instructions for using [267](#)  
     reference information for [513](#)  
     using to restore a file pool [128](#)  
 FILESERV LIST command [19](#), [30](#), [521](#)  
 FILESERV LOG command  
     instructions for using [133](#)  
     reference information for [523](#)  
 FILESERV MINIDISK command  
     instructions for using [196](#)  
     recovering from errors of [201](#)  
     reference information for [525](#)  
 FILESERV MOVEUSER command  
     instructions for using [98](#)  
     reference information for [528](#)  
 FILESERV REGENERATE command  
     instructions for using [217](#)  
     recovering from errors of [221](#)  
     reference information for [531](#)  
 FILESERV REORG command  
     instructions for using [213](#)  
     reference information for [535](#)  
 FILESERV START command [46](#), [55](#), [68](#), [206](#), [328](#), [330](#), [537](#)  
 FORCE operator command  
     instructions for using [57](#)  
     reference information for [539](#)  
 forcing a file pool server user [539](#)  
 forcing a server user [57](#)  
 FORMAT  
     as used by FILESERV MINIDISK processing [201](#)  
     as used in replacing log minidisks [133](#)  
     command (CMS)  
         as used during file pool generation [519](#)  
         as used during file pool regeneration [218](#)  
     startup parameter [107](#), [133](#), [201](#)  
 FORMAT startup parameter  
     as used by FILESERV GENERATE processing [513](#)  
     description of [346](#)  
 formatting the output of an audit [146](#)  
 forward recovery [308](#)  
 FP FUNCTION CODE field in FILEPOOL FORMAT AUDIT  
 output [160](#)  
 fragmentation of catalog indexes [213](#)  
 free space in CRR log name table [323](#)  
 FULLDUMP startup parameter  
     controlling what a server dumps [29](#)

FULLDUMP startup parameter (*continued*)  
description of [344](#)  
functional flow, CRR [311](#)  
functions, CRR [294](#)

## G

GB, definition of [250](#)  
GCS (Group Control System)  
storage dump [29](#)  
GCS (Group Control System) directories [38](#)  
general data buffer, SFS [209](#)  
generate a CRR recovery server [317](#)  
generating a file pool to restore it [127](#)  
generic profile query, SFS [174](#)  
generic profile, SFS [175](#)  
get directory entry request [151](#)  
gigabyte, definition of [250](#)  
GRANT ADMIN operator command  
instructions for using [142](#)  
reference information for [541](#)  
Group Control System (GCS)  
storage dump [29](#)  
Group Control System (GCS) directories [38](#)  
GROUPTHRESH startup parameter  
changing threshold percentage [208](#)  
description of [347](#)

## H

HELP, online  
operator [362](#)  
heuristic damage [379](#), [380](#), [385](#), [386](#), [390](#), [393](#)  
heuristic decision [390–392](#), [539](#), [540](#)  
heuristic response [390](#), [392](#)  
HX command, stopping server with [67](#)

## I

implicit read authority [140](#)  
implicit write authority [140](#)  
incomplete work, resolution of [56](#)  
inconsistencies of control minidisk, recovery from [109](#)  
incremental availability [122](#)  
index, CRR [377](#), [380](#), [383](#), [386](#), [390](#)  
indexes on file pool catalogs [213](#)  
Initial Program Load (IPL)  
FILEPOOL operand of [74](#)  
using to define default file mode A [74](#)  
initialization accounting record [225](#)  
initialize  
file pool [247](#)  
file pool minidisks [267](#)  
INPUT ddname [146](#), [149](#)  
INPUTCTL ddname [149](#)  
installing  
file pool [247](#)  
Inter-User Communications Vehicle (IUCV)  
estimating connections for [258](#)  
maximum connections for [61](#)  
maximum number of links per server machine [685](#)  
IPL (Initial Program Load)  
FILEPOOL operand of [74](#)

IPL (Initial Program Load) (*continued*)  
using to define default file mode A [74](#)  
IPL command, stopping server with [67](#)  
ITRACE operator command [22](#), [30](#), [542](#)  
ITRACE startup parameter [29](#), [351](#)  
IUCV (Inter-User Communications Vehicle)  
estimating connections for [258](#)  
maximum connections for [61](#)  
maximum number of links per server machine [685](#)  
IUCV control statement for server machines [259](#)

## K

keyword control statements  
for FILEPOOL FORMAT AUDIT [149](#)  
for FILESERV GENERATE [514](#)  
for FILESERV MINIDISK [525](#)  
for FILESERV MINIDISK or FILEPOOL MINIDISK [202](#)

## L

LABELDEF command  
as used for FILEPOOL BACKUP [114](#)  
as used for FILEPOOL FILELOAD [124](#)  
as used for FILEPOOL RELOAD [116](#), [126](#)  
as used for FILEPOOL RESTORE [119](#)  
as used for FILEPOOL UNLOAD [112](#)  
last agent optimization [304](#)  
limit monitoring [58](#)  
limits, architected [28](#), [685](#)  
limp mode [296](#), [317](#)  
LINK  
restriction regarding control minidisk [218](#)  
restriction regarding storage group minidisks [198](#)  
LINK command  
as used during file pool regeneration [218](#)  
as used during minidisk replacement [130](#)  
LINK control statements [198](#), [218](#)  
LIST ddname [521](#)  
local IDs [347](#)  
local IDs, using with remote users [237](#)  
LOCAL startup parameter  
description of [347](#)  
making file pool available to other processors [236](#)  
local, definition of [233](#)  
LOCALID tag [238](#)  
lock request [152](#)  
locks  
deleting [403](#)  
during a restore [118](#)  
during FILEPOOL RESTORE [120](#)  
effect of DELETE PUBLIC on [94](#)  
QUERY FILEPOOL CONFLICT command [563](#)  
status of, after DELETE ADMINISTRATOR [401](#)  
status of, after REVOKE ADMIN [663](#)  
log data [14](#)  
log minidisks  
block size for [515](#)  
changing the location of [133](#), [523](#)  
changing the size of [133](#), [523](#)  
considerations for, when switching to NOBACKUP  
processing [107](#)  
defining virtual device number for [515](#), [518](#)

- log minidisks (*continued*)
  - description of [14](#)
  - determining allocation for [251](#)
  - freeing space on [57](#)
  - maximum size of [685](#)
  - minimum recommended size of [252](#), [254](#)
  - moving to another physical device [133](#)
  - placement of [253](#)
  - recommended size for NOBACKUP [254](#)
  - reformatting [133](#), [523](#)
  - relationship to control data backups [105](#)
  - replacing both [133](#)
  - replacing one [129](#)
  - tuning the size of [62](#)
  - use of, when restoring control data [111](#)
- log name table, CRR [306](#)
- log ring, CRR [307](#)
- LOG1 ddname [268](#), [515](#)
- LOG2 ddname [268](#), [515](#)
- logging function, CRR [306](#)
- logical space [58](#)
- logical unit of work (LUW) [15](#)
- logs, CRR [306](#)
- lookup BFS request [157](#)
- LU 6.2 sync point architecture, SNA [289](#)
- LU 6.2, SNA [353](#), [643](#)
- LU name, migration, changing [51](#)
- LUNAME startup parameter
  - as specified for VMSYSR [44](#)
  - description of [352](#)
- LUW (logical unit of work) [15](#)
- LUWID [310](#)
- LUWID instance [355](#)

## M

- MACHINE directory statement [242](#)
- macros
  - CMS
    - DMSJNEPL [239](#)
    - EXITBUFF [209](#)
- MAINT user ID
  - as a secondary user [21](#)
  - directories owned by [39](#)
  - role of, in VMPSFS file pool [40](#)
  - role of, in VMSYS file pool [37](#)
  - role of, in VMSYSR file pool [40](#)
  - role of, in VMSYSU file pool [39](#)
- making a file pool available to a TSAF collection [233](#)
- manage
  - Byte File System [71](#)
  - changing amount of space allocated [91](#)
  - changing threshold values [92](#)
  - checking to see who is enrolled [92](#)
  - creating BFS file spaces [71](#)
  - CRR log name table [321](#)
  - CRR performance [327](#)
  - deleting enrolled users and file spaces [93](#)
  - deleting everyone [94](#)
  - determining space assigned [92](#)
  - enrolling public [92](#)
  - enrolling SFS users [71](#)
  - establishing clients for BFS file spaces [80](#)
  - moving users and file spaces [95](#)

- manage (*continued*)
  - renaming file spaces [99](#)
  - users [71](#)
- MATDNCRR APPEND [716](#)
- MATFPCRR APPEND [717](#)
- MAXCONN values
  - for administration machines [142](#)
  - for file pool users [74](#)
  - for server machines [258](#)
  - monitoring [61](#)
- MAXDISKS control statement [249](#), [268](#), [515](#), [685](#)
- MAXDISKS value
  - estimating [249](#)
  - increasing [217](#)
  - meaning of [249](#)
  - monitoring [61](#)
  - recommended value for [249](#), [268](#)
  - specifying during generation [268](#)
  - specifying in control statement file [515](#)
- maximums
  - 4KB blocks per file space [72](#)
  - architected limits [28](#)
  - increasing by regeneration [217](#)
  - listing for file pool [685](#)
- MAXUSERS control statement [220](#), [268](#), [514](#), [685](#)
- MAXUSERS value
  - estimating [249](#)
  - increasing [217](#)
  - meaning of [249](#)
  - recommended value for [249](#)
  - specifying during generation [268](#)
  - specifying in control statement file [514](#)
- MDISK
  - adjustment to, when removing from file pool [207](#)
  - for a server work minidisk [260](#)
  - for file pool control minidisk [218](#)
  - for file pool minidisks [198](#), [260](#)
- MDISK control statements [130](#), [198](#), [207](#), [218](#), [221](#), [250](#), [260](#), [329](#)
- MDKnnnnn ddname [203](#), [517](#)
- media failures
  - recovering from
    - with no good log [111](#)
    - with one good log [111](#)
- MESSAGE command
  - at secondary user console [56](#)
  - importance of [56](#)
- message examples, notation used in [xxiii](#)
- messages
  - during start of multiple user mode [55](#)
  - from a file pool server machine [56](#)
  - suppressing [353](#)
  - when limit is reached on server [61](#)
- migrated files
  - finding [529](#)
- migration considerations
  - catalog reorganization [52](#)
  - data space exploitation [52](#)
  - LU name of CRR recovery server [51](#)
  - SFS and CRR file pool startup parameters [51](#)
  - storage use exits [52](#)
- mini-dumps [29](#)
- minidisk
  - access mode required [198](#), [218](#)

minidisk (*continued*)  
adding to the file pool [195](#)  
block size of [201](#)  
defining maximum number of [515](#)  
FBA restrictions for data spaces [244](#)  
initializing for file pools [267](#)  
maximum number per storage group [685](#)  
MDISK control statements for [260](#)  
moving contents to an SFS directory [78](#)  
password requirements for [198](#)  
removing from a file pool [206](#)  
replacing [129](#)  
virtual device number considerations for [198](#)  
volume labels for [131](#), [219](#)

mkcat BFS request [157](#)

MODIFY USER command

instructions for using [91](#)

overview of [27](#)

reference information for [544](#)

monitoring file pool server operation [59](#)

monitoring performance [65](#)

monitoring server operation [56](#)

moving

users [95](#)

MSGs startup parameter [353](#)

multiple user mode

backups at shutdown of [104](#)

backups during [105](#)

backups during, using BACKUP command [105](#)

backups during, using FILEPOOL CONTROL BACKUP command [105](#)

definition of [20](#)

how to automatically start [272](#)

monitoring [56](#)

restricting user access during [69](#)

specifying startup parameters for [339](#)

starting [55](#)

stopping [66](#)

tracing server execution during [351](#)

## N

naming file pools [264](#), [344](#)

NetView CLISTS

AOPMRRM FPCLST [710](#)

CRRM001 FPPANEL [714](#)

CRRM002 FPPANEL [714](#)

CRRM003 FPPANEL [715](#)

CRRM004 FPPANEL [715](#)

CRRMSGR DNCLST [711](#)

CRRMSGR EXEC [713](#)

CRROPF APPEND [716](#)

MATDNCRR APPEND [716](#)

MATFPCRR APPEND [717](#)

nickname resolution for SFS commands [239](#)

nicknames, using with remote users [237](#)

NOACCOUNT startup parameter [224](#), [347](#)

NOAUDIT startup parameter [348](#)

NOBACKUP startup parameter

as specified for VMSYS, VMPSFS, VMSYSU, and VMSYSR [42](#)

description of [342](#)

switching from [104](#)

switching to [107](#)

NOCRR startup parameter [348](#)

NODFSMS startup parameter [349](#)

NODUMP startup parameter

controlling what a server dumps [29](#)

description of [344](#)

NOSECURITY startup parameter [350](#)

NOETRACE startup parameter [351](#)

NOFORMAT startup parameter

as used by FILESERV GENERATE processing [513](#)

description of [346](#)

NOITRACE startup parameter [351](#)

NOLUNAME startup parameter [352](#)

NOMSGS startup parameter [353](#)

non-file pool backup facilities [134](#)

NORESTORE startup parameter [353](#)

NOSAVESEGID startup parameter [355](#)

NOSHUTDOWN SIGNAL startup parameter [355](#)

notation used in message and response examples [xxiii](#)

## O

OID OF RESOURCE field in FILEPOOL FORMAT AUDIT output [161](#)

online HELP facility, using [360](#)

open BFS request [157](#)

opendir BFS request [157](#)

operation of a file pool server [15](#), [104](#), [105](#), [218](#)

operation of a server [55](#)

operator

accounting record [226](#)

commands

accounting records generated for [225](#)

overview of [22](#)

console [19](#)

operator commands

AUDIT [362](#)

BACKUP [366](#)

CRR ERASE LU [368](#)

CRR ERASE LUWID [370](#)

CRR QUERY LOG [371](#)

CRR QUERY LOGTABLE [373](#)

CRR QUERY LU [376](#)

CRR QUERY LUWID [382](#)

CRR RESUME [388](#)

CRR RESYNC [390](#)

CRR SUSPEND [394](#)

DEFBACKUP [398](#)

DISABLE [412](#)

ENABLE [415](#)

ERASE LUNAME [427](#)

ETRACE [429](#)

FORCE [539](#)

GRANT ADMIN [541](#)

ITRACE [542](#)

QUERY DEFBACKUP [553](#)

QUERY DISABLE [555](#)

QUERY LOGTABLE [658](#)

QUERY PREPARED [660](#)

REVOKE ADMIN [663](#)

STOP [666](#)

OPTION control statement

for server machines [258](#)

for user machines [75](#)

for using the accounting facility [223](#)



OUTPUT ddname [149](#)  
output file for security audit [144](#), [504](#)  
OWNERID control statement for FILEPOOL FORMAT AUDIT [159](#)

## P

parameter list  
    DMSJNE exit routine [239](#)  
parameters, startup, file pool, migration [51](#)  
partial audit [143](#)  
passwords needed for file pool minidisks [198](#), [218](#)  
performance  
    management, CRR [327](#)  
    monitoring [65](#)  
    reorganizing file pool catalogs [213](#)  
physical  
    saved segment CMSFILES [43](#)  
    space [58](#)  
physical saved segment CMSFILES [355](#)  
planning  
    for file pools [31](#)  
point of error, restoring to [102](#)  
POOLDEF file  
    effect of FILESERV MINIDISK on [206](#)  
    general description of [13](#)  
    generation of [513](#)  
    manually restoring [111](#)  
    using FILESERV DEFAULT to modify [144](#)  
POSIXOPT SETIDS ALLOW  
    specifying [259](#)  
post-installation activities [37](#)  
postcoordination exit, CRR [298](#)  
potential addressable space, increasing [217](#)  
practical maximums [28](#)  
precoordination exit, CRR [298](#)  
preventing  
    multiple user mode access [69](#)  
primary user [21](#)  
privilege classes needed for servers [257](#)  
problem management [330](#)  
product participation in CRR [307](#)  
PROFILE EXEC  
    for an administration machine [272](#)  
    for server machines [262](#), [273](#)  
    for SFS users [79](#)  
    for the AUTOLOG1 machine [273](#)  
profile rename, ESM [175](#)  
protected conversations [289](#)  
protected resource [289](#)–[291](#)  
PUBLIC enrollment  
    at generation time [272](#)  
    DELETE PUBLIC command [406](#)  
    deleting [94](#)  
    ENROLL PUBLIC command [420](#)  
    enrolling [92](#)  
    remote use considerations for [234](#)  
    verifying [92](#)

## Q

QUALIFIED OBJECT NAME field in FILEPOOL FORMAT AUDIT output [161](#)

quasi ownership of files [139](#)  
QUERY ACCESSORS command  
    overview of [27](#)  
    reference information for [547](#)  
    using with dataspace option [245](#)  
QUERY DATASPACE command  
    overview of [27](#)  
    reference information for [550](#)  
QUERY DEFBACKUP operator command [553](#)  
QUERY DISABLE operator command [22](#), [555](#)  
QUERY FILEPOOL AGENT command  
    monitoring limits on USERS startup parameter [59](#)  
    overview of [27](#)  
    reference information for [557](#)  
QUERY FILEPOOL CATALOG command  
    monitoring logical catalog space [64](#)  
    overview of [27](#)  
    reference information for [560](#)  
    using to determine when reorganization is needed [213](#)  
QUERY FILEPOOL CONFLICT command  
    overview of [27](#)  
    reference information for [563](#)  
    using when forcing a server user [57](#)  
QUERY FILEPOOL COUNTER command  
    monitoring rollbacks because of deadlock [65](#)  
    overview of [27](#)  
    reference information for [569](#)  
QUERY FILEPOOL CRR command  
    overview of [27](#)  
    reference information for [574](#)  
QUERY FILEPOOL DISABLE command  
    overview of [27](#)  
    reference information for [577](#)  
QUERY FILEPOOL LOG command  
    monitoring SFS log minidisk usage [62](#)  
    overview of [27](#)  
    reference information for [583](#)  
    using during control data backup [107](#)  
QUERY FILEPOOL MINIDISK command  
    monitoring physical storage in user storage group [60](#)  
    overview of [27](#)  
    reference information for [586](#)  
    using when enrolling users [76](#)  
QUERY FILEPOOL OVERVIEW command  
    monitoring MAXCONN value [61](#)  
    monitoring MAXDISKS value [61](#)  
    monitoring total physical file pool space [64](#)  
    monitoring virtual storage consumption in server [63](#)  
    overview of [27](#)  
    reference information for [589](#)  
    using when adding minidisks [196](#)  
QUERY FILEPOOL REPORT command  
    monitoring logical catalog space [64](#)  
    overview of [27](#)  
    reference information for [592](#)  
    using to monitor limits [58](#)  
QUERY FILEPOOL STATUS command  
    overview of [28](#)  
    reference information for [623](#)  
QUERY FILEPOOL STORGRP command  
    overview of [28](#)  
    reference information for [652](#)  
    using during control data backups [106](#)  
    using during user data backup [115](#)

QUERY FILEPOOL STORGRP command (*continued*)  
  using when adding minidisks [196](#)  
  using when changing the amount of user's file space [91](#)  
  using when enrolling users [77](#)  
QUERY LIMITS command  
  determining committed blocks in accounting record [225](#)  
  monitoring physical space in user storage groups [60](#)  
  overview of [28](#)  
  reference information for [655](#)  
  using during control data backups [106](#)  
QUERY LOGTABLE command [658](#)  
QUERY PREPARED operator command [660](#)  
QUICKDSP operand of OPTION control statement [259](#)

## R

RACROUTE macro  
  called by SFS authorization checking routines [172](#)  
  mapping to SFS calls  
    non-operator commands and CSL routines [698](#)  
    objects [699](#)  
    operator commands [698](#)  
  request formats [696](#)  
read authority check, SFS [174](#)  
read authority, implicit [140](#)  
readlink BFS request [158](#)  
recording of authorizations [137](#)  
records, accounting [224](#)  
recovery  
  from control minidisk verification errors [109](#)  
  related procedures [101](#)  
  server, CRR [296](#)  
  token [161](#)  
recovery server operator command (CRR)  
  CRR ERASE LU [368](#)  
  CRR ERASE LUWID [370](#)  
  CRR QUERY LOG [371](#)  
  CRR QUERY LOGTABLE [373](#)  
  CRR QUERY LU [376](#)  
  CRR QUERY LUWID [382](#)  
  CRR RESUME [388](#)  
  CRR RESYNC [390](#)  
  CRR SUSPEND [394](#)  
  QUERY LOGTABLE [658](#)  
recovery token [377](#), [380](#), [382](#), [383](#), [386](#), [660](#), [661](#)  
recovery TPN [377](#), [380](#), [383](#), [386](#)  
regenerating a file pool [217](#)  
registration and exits, CRR [297](#)  
REGRANT EXEC [674](#)  
remote  
  definition of [233](#)  
  use, topics related to [233](#)  
  users enrolling [236](#)  
REMOTE startup parameter  
  description of [347](#)  
  making file pool available to other processors [235](#)  
remote use  
  startup parameters related to [347](#)  
remove designation of alternate CRR recovery server [320](#)  
removing a file pool from a TSAF collection [236](#)  
rename BFS request [158](#)  
renaming SFS file spaces [99](#)  
reorganizing file pool catalogs [213](#)  
replace

  replace (*continued*)  
    both CRR log minidisks [328](#)  
    file pool minidisk [129](#)  
    minidisks in CRR file pool [327](#)  
    one CRR log minidisk [328](#)  
repository  
  data  
    overview [17](#)  
RESERVE command  
  as used during file pool generation [519](#)  
  as used during file pool regeneration [219](#)  
  as used during minidisk replacement [131](#)  
resource adapter [296](#)  
response examples, notation used in [xxiii](#)  
restart automatic periodic retry of resynchronization [324](#)  
restart recovery [56](#)  
restore  
  control data [110](#)  
RESTORE ddname [110](#)  
RESTORE startup parameter [110](#), [353](#)  
restoring lost data  
  startup parameters related to [353](#)  
restrict  
  user access [69](#)  
RESULTS control statement for FILEPOOL FORMAT AUDIT [159](#)  
RESULTS OF AUTHORITY CHECK field in FILEPOOL FORMAT AUDIT output [160](#)  
resynchronization function [305](#), [390](#)  
resynchronization roles [380](#), [386](#)  
resynchronization states [380](#), [386](#)  
RESYNCINTERVAL startup parameter [354](#)  
REVOKE ADMIN operator command  
  instructions for using [143](#)  
  reference information for [663](#)  
revoked alias [121](#)  
rmdir BFS request [158](#)  
routines [28](#), [151](#)  
RPIUCMS module  
  ESM program check call [698](#)  
  initialization call [697](#)  
  termination call [697](#)  
run-away write [605](#), [649](#)

## S

SAC component of server [351](#), [429](#)  
sample  
  execs [669](#)  
  files  
    copying [273](#)  
    installed in VMSYS [37](#)  
saved  
  segment  
    CMSFILES [43](#)  
saved segment CMSFILES [355](#)  
SAVESEGID startup parameter  
  as specified for VMSYS, VMPSFS, VMSYSU, and VMSYSR [43](#)  
  description of [355](#)  
  usage recommendations for VMSYS, VMPSFS, VMSYSU, and VMSYSR [43](#)  
SCIF (Single Console Image Facility) [21](#)  
secondary user



- secondary user (*continued*)
  - changing the setting of, for VMSYS, VMSYSU, and VMSYSR [47](#)
  - console, server spooling [47](#)
  - defining in SCIF [21](#)
  - specifying [260](#)
  - spooling server console output [47](#)
- security
  - external security manager
    - SFS interface [169](#)
  - manager [162](#)
  - overview [137](#)
  - remote use considerations for [234](#)
  - SFS interface to external security manager
    - administrator command authorization processing [692](#)
    - authorization processing [172](#)
    - command authorization checking routine [177](#)
    - create authority check [174](#)
    - DMSAAUTH exit routine [177](#)
    - DMSOAUTH exit routine [177](#)
    - DMSSECIT exit routine [176](#)
    - DMSUAUTH exit routine [177](#)
    - ESM initialization [172](#)
    - ESM program check routine [176](#)
    - ESM requirements [169](#)
    - event control blocks [173](#)
    - exit routines, IBM-supplied [171](#), [172](#)
    - exit routines, writing replacements [176](#)
    - function [172](#)
    - generic profile query [174](#)
    - initialization/termination routine [176](#)
    - maintaining list of authorizations [170](#)
    - mapping SFS authorization calls to RACROUTE requests [689](#)
    - object authorization checking routine [177](#)
    - object authorization processing [694](#)
    - operator command authorization checking routine [177](#)
    - operator command authorization processing [689](#)
    - overview [171](#)
    - password protection for objects [192](#)
    - profile rename, ESM [175](#)
    - providing user or administrator interface [170](#)
    - read authority check [174](#)
    - rebuild commands and CSL routines [692](#)
    - RPIUCMS module [172](#)
    - security adapter, communications with security machine [172](#)
    - security adapter, loading into server [172](#)
    - types of authorization checks [174](#)
    - user data structure [695](#)
    - user data support [189](#)
    - what authority query [174](#)
    - write authority check [174](#)
  - security adapter
    - communications with security machine [172](#)
    - function [172](#)
    - loading into SFS server [172](#)
    - posting event control blocks [173](#)
  - security audit output file [504](#)
  - security auditing
    - AUDIT operator command [362](#)
    - brief overview of [29](#)
  - security auditing (*continued*)
    - changing output file for [144](#)
    - defining output file for [144](#)
    - FILEPOOL FORMAT AUDIT command [455](#)
    - formatted output [160](#)
    - formatting the output of [146](#)
    - security audit facility [143](#)
    - starting and stopping [145](#)
    - startup parameters related to [348](#)
    - SECURITY tag setting for file pools [234](#)
    - SEND command, CP [56](#)
    - SERVER DMSPARMS file [265](#), [339](#)
    - server exit considerations, file pool [687](#)
    - server machine
      - defining [256](#)
      - how to automatically start [272](#)
      - monitoring operation of [56](#)
      - specifying startup parameters for [339](#)
      - starting multiple user mode processing for [55](#)
      - stopping multiple user processing for [66](#)
    - server operator command, (CRR) recovery
      - CRR ERASE LU [368](#)
      - CRR ERASE LUWID [370](#)
      - CRR QUERY LOG [371](#)
      - CRR QUERY LOGTABLE [373](#)
      - CRR QUERY LU [376](#)
      - CRR QUERY LUWID [382](#)
      - CRR RESUME [388](#)
      - CRR RESYNC [390](#)
      - CRR SUSPEND [394](#)
      - QUERY LOGTABLE [658](#)
    - service aids
      - brief overview of [29](#)
      - ETTRACE operator command [429](#)
      - external tracing [351](#)
      - FILESERV LIST command [521](#)
      - internal tracing [351](#)
      - ITRACE operator command [542](#)
      - mini-dumps [29](#)
    - session duration [225](#)
    - SET FILEPOOL command [79](#)
    - SET THRESHOLD command
      - overview of [28](#)
      - reference information for [664](#)
    - setting up a file pool for remote use [233](#)
    - SETUP EXEC
      - creating for VMSYS, VMPSFS, VMSYSU, and VMSYSR [41](#)
      - used for automatic server starting [273](#)
  - SFS
    - aliases
      - unresolved [122](#)
  - SFS aliases
    - effect of FILEPOOL RESTORE on [121](#)
  - SFS users
    - moving to another storage group [95](#), [96](#)
    - multiple user mode [95](#)
  - SFSTRANS EXEC [670](#)
  - shadow copies of files [685](#)
  - SHARE control statement [259](#)
  - share disable locks
    - checking status of [555](#)
    - creating [412](#)
    - deleting [415](#)
  - Shared File System

## Shared File System (*continued*)

- aliases

  - unresolved [122](#)

## Shared File System (SFS)

- accounting facility

  - server user accounting records, customizing [229](#)

- authorizations [137](#)

- create authority check [174](#)

- data spaces, using with [241](#)

- directory access [73](#), [198](#)

- ESM user data support

  - creation of data structure [695](#)

  - CSL routines that allow user data [190](#)

  - format of data structure [695](#)

  - function [189](#)

  - parts [189](#)

  - password protection for objects [192](#)

  - processing DMSUDATA CSL routine [191](#)

  - processing user data [189](#)

  - special considerations for GRANT and REVOKE [191](#)

- event control block

  - error handling [173](#)

  - format [173](#)

  - posting by security adapter [173](#)

- exits

  - command authorization checking [177](#)

  - DMSAAUTH routine [177](#)

  - DMSOAUTH routine [177](#)

  - DMSSECIT routine [176](#)

  - DMSSFSEX routine [209](#)

  - DMSUAUTH routine [177](#)

  - ESM initialization/termination [176](#)

  - ESM program check [176](#)

  - file space usage [209](#)

  - object authorization checking [177](#)

  - operator command authorization checking [177](#)

  - storage use [209](#)

  - user storage group full [209](#)

- external security manager, interface to

  - administrator command authorization processing [692](#)

  - authorization processing [172](#)

  - command authorization checking routine [177](#)

  - create authority check [174](#)

  - DMSAAUTH exit routine [177](#)

  - DMSOAUTH exit routine [177](#)

  - DMSSECIT exit routine [176](#)

  - DMSUAUTH exit routine [177](#)

  - ESM initialization [172](#)

  - ESM program check routine [176](#)

  - event control blocks [173](#)

  - exit routines, IBM-supplied [171](#), [172](#)

  - exit routines, writing replacements [176](#)

  - generic profile query [174](#)

  - initialization/termination routine [176](#)

  - mapping SFS authorization calls to RACROUTE

  - requests [689](#)

  - object authorization checking routine [177](#)

  - object authorization processing [694](#)

  - operator command authorization checking routine [177](#)

  - operator command authorization processing [689](#)

  - overview [171](#)

  - profile rename [175](#)

## Shared File System (SFS) (*continued*)

- external security manager, interface to (*continued*)

  - read authority check [174](#)

  - rebuilt commands and CSL routines [692](#)

  - requirements [169](#)

  - RPIUCMS module [172](#)

  - types of authorization checks [174](#)

  - user data support [189](#)

  - what authority query [174](#)

  - write authority check [174](#)

- external security managers [161](#)

- file pools [99](#), [143](#), [223](#), [250](#)

- fpcont [13](#)

- general data buffer [209](#)

- generic profile query [174](#)

- local IDs, coding program to supply for remote users [238](#)

- log [16](#), [105](#), [130](#)

- nickname resolution for SFS commands [239](#)

- overview [10](#)

- post-installation [37](#)

- read authority check [174](#)

- remote users, coding program to supply local IDs [238](#)

- security

  - auditing [143](#)

  - authorization processing [172](#)

  - create authority check [174](#)

  - ESM initialization [172](#)

  - ESM requirements [169](#)

  - generic profile query [174](#)

  - profile rename, ESM [175](#)

  - read authority check [174](#)

  - types of authorization checks [174](#)

  - what authority query [174](#)

  - write authority check [174](#)

  - security manager, augmenting or replacing [169](#)

  - server user accounting records, customizing [229](#)

  - startup parameters [339](#)

  - storage use exits [209](#)

  - using data spaces [241](#)

  - what authority query [174](#)

  - with CRR [143](#), [223](#), [247](#), [279](#), [315](#)

  - write authority check [174](#)

- sharing files using administration authority [139](#)

- shoulder tap, upstream [395](#)

- shutdown signal from CP, enabling for [355](#)

- SHUTDOWN SIGNAL startup parameter [355](#)

- Single Console Image Facility (SCIF) [21](#)

- SNA (System Network Architecture) [233](#)

- SNA LU 6.2 [353](#), [643](#)

- SNA LU 6.2 sync point architecture [289](#)

- softlink BFS request [158](#)

- space

  - accounting for use of [227](#), [229](#)

  - adding to the file pool [195](#)

  - allocating to file spaces [72](#)

  - changing amounts allocated to users [91](#)

  - directing control data backups to [269](#)

  - increasing file pool limit of [217](#)

  - managing [195](#), [212](#)

  - monitoring consumption of [60](#)

  - querying a user's allocation [92](#)

  - removing from file pool [206](#)

  - setting warning threshold for [208](#)

- SSI startup parameter [347](#)
- staging backups to tape [115](#)
- starting
  - accounting facility [223](#)
  - and stopping audits [145](#)
  - multiple user mode processing [55](#)
  - the server [55](#)
- startup parameters
  - ACCOUNT [347](#)
  - ADMIN [342](#)
  - AUDIT [348](#)
  - BACKUP [342](#)
  - CATBUFFERS [343](#)
  - CRR [348](#)
  - CTLBUFFERS [343](#)
  - descriptions of [342](#)
  - DFSMS [349](#)
  - DUMP [344](#)
  - ESECURITY [350](#)
  - ETRACE [351](#)
  - FIFO [345](#)
  - file pool, migration [51](#)
  - FILEPOOLID [344](#)
  - FORMAT [346](#)
  - FULLDUMP [344](#)
  - GROUPTHRESH [347](#)
  - ITRACE [351](#)
  - LOCAL [347](#)
  - LUNAME [352](#)
  - MSGS [353](#)
  - NOACCOUNT [347](#)
  - NOAUDIT [348](#)
  - NOBACKUP [342](#)
  - NOCRR [348](#)
  - NODFSMS [349](#)
  - NODUMP [344](#)
  - NOSECURITY [350](#)
  - NOETRACE [351](#)
  - NOFORMAT [346](#)
  - NOITRACE [351](#)
  - NOLUNAME [352](#)
  - NOMSGS [353](#)
  - NORESTORE [353](#)
  - NOSAVESEGID [355](#)
  - NOSHUTDOWN SIGNAL [355](#)
  - REMOTE [347](#)
  - RESTORE [353](#)
  - RESYNCINTERVAL [354](#)
  - SAVESEGID [355](#)
  - SHUTDOWN SIGNAL [355](#)
  - specifying during generation [263](#)
  - SSI [347](#)
  - USERS [355](#)
- status
  - of a data space [550](#)
  - of file pool
    - determining if additional DASD is needed [195](#)
    - information about physical space allocated [204](#)
    - using QUERY FILEPOOL REPORT command [592](#)
    - using QUERY FILEPOOL STATUS command [623](#)
- stop
  - accounting facility [224](#)
  - stop automatic periodic retry of resynchronization [324](#)
  - STOP operator command [66](#), [666](#)
- stopping
  - multiple user mode processing [66](#)
- storage group backup
  - considerations when managed by DFSMS/VM [115](#)
  - creating [113](#), [115](#)
  - FILEPOOL BACKUP command [432](#)
  - FILEPOOL CLEANUP command [437](#)
  - FILEPOOL FILELOAD command [451](#)
  - FILEPOOL LIST BACKUP command [459](#)
  - FILEPOOL RESTORE command [488](#)
  - running concurrently [115](#)
- storage groups
  - activating storage use exits [209](#)
  - adding space to [195](#)
  - alias considerations during restore [120](#)
  - allocating space to a file space within [72](#)
  - balancing use of [73](#)
  - block size of minidisks within [201](#)
  - changing user space allocations within [91](#)
  - concurrency during a restore [118](#)
  - defining [204](#)
  - defining maximum number of [515](#)
  - deleting file space from [93](#)
  - description of [17](#)
  - determining when space is needed [196](#)
  - disabling [412](#)
  - enabling [415](#)
  - enrolling users in [71](#)
  - FILEPOOL BACKUP command [432](#)
  - FILEPOOL CLEANUP command [437](#)
  - FILEPOOL FILELOAD command [451](#)
  - FILEPOOL LIST BACKUP command [459](#)
  - FILEPOOL RESTORE command [488](#)
  - in VMSYS [37](#)
  - in VMSYSU [39](#)
  - listing files on corrupted minidisk [123](#)
  - maximum number of [685](#)
  - maximum number of 4KB blocks [685](#)
  - maximum number of minidisks per [685](#)
  - maximum size of minidisk within [685](#)
  - monitoring physical storage in [60](#)
  - moving BFS file spaces to or from [95](#)
  - moving SFS users to or from [95](#), [96](#)
  - moving users in [95](#)
  - numbering consideration for [204](#)
  - restoring [115](#)
  - restoring individual files from [123](#)
  - restoring individual files in migrated status [127](#)
  - restoring multiple [122](#)
  - restoring when managed by DFSMS/VM [123](#)
  - setting space warning threshold for [208](#)
  - when to define additional [195](#)
- storage use exits, SFS
  - file space usage exit
    - default action [210](#)
    - function [209](#)
    - sample (dormant) function [210](#)
    - when called by SFS server [209](#)
  - general data buffer [209](#)
  - migration considerations [52](#)
  - modifying or replacing [210](#)
  - user storage group full exit
    - default action [210](#)
    - function [209](#)

- storage use exits, SFS (*continued*)
  - user storage group full exit (*continued*)
    - sample (dormant) function [210](#)
    - when called by SFS server [209](#)
- storage, DASD
  - accounting for use of [227](#), [229](#)
  - adding to the file pool [195](#)
  - allocating to file spaces [72](#)
  - changing amounts allocated to users [91](#)
  - directing control data backups to [269](#)
  - increasing file pool limit of [217](#)
  - managing [195](#), [212](#)
  - monitoring consumption of [60](#)
  - querying a user's allocation [92](#)
  - removing from file pool [206](#)
  - setting warning threshold for [208](#)
- storage, virtual
  - defining [257](#)
  - minimum requirement [18](#)
  - monitoring consumption of [63](#)
  - recommended amount of [18](#)
  - recommended setting for [257](#)
- subtasks
  - accounting record for [227](#)
  - description of [225](#)
- SVMSTAT operand of the OPTION control statement [259](#)
- sync point
  - exits, CRR [298](#)
  - processing, CRR [298](#)
  - tree [299](#)
- sync point roles [378](#), [384](#)
- sync point states [378](#), [384](#)
- sync point tree [378](#), [384](#), [390](#), [391](#), [393](#)
- syntax diagrams, how to read [xxi](#)
- system
  - data [31](#)
- System Network Architecture (SNA) [233](#)

**T**

- TALLY EXEC [677](#)
- tape files
  - directing control data backups to [269](#)
  - specifying for control data backups [108](#)
- TB, definition of [251](#)
- TEMP ddname [214](#), [221](#), [533](#), [535](#)
- temporary files
  - used in FILEPOOL MINIDISK processing [199](#)
  - used in FILESERV MINIDISK processing [201](#)
  - used in FILESERV REGENERATE processing [220](#)
  - used when backing up control data [367](#)
  - used while restoring control data [111](#)
- terabyte, definition of [251](#)
- termination accounting record [227](#)
- threshold for space warning, setting
  - description of GROUPTHRESH startup parameter [347](#)
  - for physical DASD in storage group [208](#)
  - for user file space using SET THRESHOLD command [664](#)
  - using GROUPTHRESH startup parameter [208](#)
- TIME control statement for FILEPOOL FORMAT AUDIT [159](#)
- TIME field in FILEPOOL FORMAT AUDIT output [159](#)
- timed wait interval
  - bypass [325](#)
  - change [325](#)
- timed wait interval, change [325](#)
- timed wait, CRR [354](#), [378](#), [379](#), [384](#), [385](#), [388](#), [389](#), [394](#), [395](#)
- timer management [160](#)
- top directory
  - considerations for [73](#)
  - creating [71](#)
  - establishing as file mode A [74](#)
  - moving minidisk contents to [78](#)
  - SFS or BFS [71](#)
- TPN tag setting for file pools [234](#)
- TPN X'06F2' [376](#), [378](#), [384](#), [395](#)
- trace level pairs [429](#)
- tracing
  - file pool access [143](#)
  - server execution [29](#)
- tracing server execution
  - ETRACE operator command [429](#)
  - ITRACE operator command [542](#)
  - startup parameters related to [351](#)
- transaction program [234](#), [309](#)
- transaction tag [377](#), [379](#), [383](#), [385](#), [457](#), [661](#)
- transaction, coordinated [355](#)
- transferring ownership of file pool objects [99](#)
- TSAF collection
  - making a file pool available to [233](#)
  - removing a file pool from [236](#)
- two-phase commit [355](#), [378–380](#), [384–386](#), [521](#), [529](#), [533](#), [536](#), [537](#), [637](#), [647](#), [648](#), [661](#)
- two-phase commit protocol [299](#)

**U**

- unenrolled users
  - renaming [99](#)
- unlink BFS request [158](#)
- unresolved aliases (SFS)
  - erasing [123](#)
- upstream shoulder tap [395](#)
- user
  - accounting record [228](#)
  - data
    - minidisks, minimum recommended size of [255](#)
    - minidisks, placement of [255](#)
    - restoring entire storage group [115](#)
    - restoring individual files [123](#)
  - minidisks [255](#)
  - minidisks, determining allocation of [255](#)
- user data
  - FILEPOOL BACKUP command for [432](#)
  - FILEPOOL CLEANUP command for [437](#)
  - FILEPOOL FILELOAD command for [451](#)
  - FILEPOOL RESTORE command for [488](#)
- user data, ESM
  - creation of data structure [695](#)
  - CSL routines that allow user data [190](#)
  - format of data structure [695](#)
  - function [189](#)
  - password protection for objects [192](#)
  - processing [189](#)
  - processing DMSUDATA CSL routine [191](#)
  - special considerations for GRANT and REVOKE [191](#)
  - support in SFS [189](#)
- user storage group backup

- user storage group backup (*continued*)
  - considerations when managed by DFSMS/VM [115](#)
  - creating [113](#), [115](#)
  - FILEPOOL BACKUP command [432](#)
  - FILEPOOL CLEANUP command [437](#)
  - FILEPOOL FILELOAD command [451](#)
  - FILEPOOL LIST BACKUP command [459](#)
  - FILEPOOL RESTORE command [488](#)
  - running concurrently [115](#)
- user storage group full exit, SFS
  - default function [210](#)
  - sample (dormant) function [210](#)
  - when called by SFS server [209](#)
- USERID control statement for FILEPOOL FORMAT AUDIT [160](#)
- USERID OF REQUESTOR field in FILEPOOL FORMAT AUDIT output [161](#)
- users
  - accounting records generated for [225](#)
  - DELETE USER command [408](#)
  - displaying current space allocations of [92](#)
  - ENROLL USER command [422](#)
  - enrolling [71](#)
  - enrolling remote [236](#)
  - forcing of [57](#)
  - listing those enrolled [92](#)
  - managing [71](#), [99](#)
  - MODIFY USER command [544](#)
  - moving [95](#)
  - moving to another file pool [96](#)
  - PROFILE EXEC for [79](#)
  - QUERY LIMITS command [655](#)
  - renaming file spaces [99](#)
  - transferring ownership among [99](#)
  - unenrolled
    - renaming [99](#)
- USERS startup parameter
  - as specified for VMSYS, VMSYSU, and VMSYSR [44](#)
  - description of [355](#)
  - monitoring [59](#)
  - moving minidisk files to [78](#)
- utime BFS request [158](#)

## V

- verification errors, recovery from [109](#)
- verification of the control minidisk [109](#)
- virtual device numbers
  - specifying during generation [268](#)
- virtual device numbers for file pool minidisks [130](#), [198](#), [261](#)
- virtual storage
  - defining [257](#)
  - minimum requirement [18](#)
  - monitoring consumption of [63](#)
  - recommended amount of [18](#)
  - recommended setting for [257](#)
- VMPSFS file pool
  - characteristics of [40](#)
  - DMSPARMS file for [41](#)
  - post-installation instructions for [37](#)
  - tailoring [40](#)
- VMSERVP server machine
  - secondary user for [40](#)
- VMSEVR server machine

- VMSEVR server machine (*continued*)
  - automatic logging of [40](#)
  - overview of [18](#)
  - secondary user for [40](#)
- VMSEVS server machine
  - automatic logging of [38](#)
  - overview of [18](#)
  - secondary user for [38](#)
- VMSEVU server machine
  - automatic logging of [39](#)
  - overview of [18](#)
  - secondary user for [39](#)
- VMSYS file pool
  - characteristics of [37](#)
  - DMSPARMS file for [41](#)
  - overview of [18](#)
  - post-installation instructions for [37](#), [49](#)
  - tailoring [40](#)
- VMSYSR file pool
  - characteristics of [40](#)
  - DMSPARMS file for [41](#)
  - overview of [18](#)
  - post-installation instructions for [37](#)
  - tailoring [40](#)
- VMSYSU file pool
  - characteristics of [39](#)
  - DMSPARMS file for [41](#)
  - overview of [18](#)
  - post-installation instructions for [37](#), [49](#)
  - tailoring [40](#)
- volume labels for file pool minidisks [131](#), [219](#)

## W

- warning threshold, setting
  - description of GROUPTHRESH startup parameter [347](#)
  - for physical DASD in storage group [208](#)
  - for user file space using SET THRESHOLD command [664](#)
  - using GROUPTHRESH startup parameter [208](#)
- what authority query, SFS [174](#)
- WHO EXEC [681](#)
- work minidisk
  - for administration machines [142](#), [262](#)
  - for server machines [254](#)
  - for server, replacing [129](#)
- work unit [14](#)
- write authority check, SFS [174](#)
- write authority, implicit [140](#)
- writing accounting records [224](#), [225](#)
- writing to the catalogs [28](#)

## X

- X'06F2' TPN [376](#), [378](#), [384](#), [395](#)
- XCONFIG ACCESSLIST directory statement [242](#)
- XCONFIG ADDRSPACE directory statement [242](#)
- XCONFIG control statement for server machines [259](#)
- XEDIT
  - locking consideration for [140](#)
  - option of QUERY, using [59](#)

## Z

z/VM HELP Facility, using [360](#)





Product Number: 5741-A09

Printed in USA

SC24-6261-02

