z/OS

Network File System Guide and Reference

SC26-7417-11

z/OS

Network File System Guide and Reference

SC26-7417-11

Note:

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 545.

This edition applies to Version 1 Release 12 of z/OS® (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC26-7417-09.

© Copyright IBM Corporation 1991, 2010.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures ix

Tables xi

About this document xiii

Required product knowledge xiii
Where to find more information xiii
Access to documents xiii
Access to softcopy documents on CD-ROM and DVD collections and the Internet xiv
How to look up message explanations with LookAt xv
How to read syntax diagrams xvi
Related protocol specifications xviii
The z/OS Basic Skills Information Center xix

How to send your comments to IBM xxi

If you have a technical problem xxi

Summary of changes. xxiii

Summary of changes for SC26-7417-10, z/OS Version 1 Release 12 xxiii
Summary of changes for SC26-7417-10, z/OS Version 1 Release 11 as updated April 2010 . . . xxiv
Summary of changes for SC26-7417-09, z/OS Version 1 Release 11 xxiv
Summary of changes for SC26-7417-08, z/OS Version 1 Release 10 xxvi
Summary of changes for SC26-7417-07, z/OS Version 1 Release 9 xxvii
Summary of changes for SC26-7417-06, z/OS Version 1 Release 8 xxix

Part 1. Using z/OS Network File System. 1

Chapter 1. Introduction 3

Overview 3
z/OS UNIX files 4
z/OS UNIX advantages. 4
NFS protocol compliance 5
Crossing between file systems–NFS server . . . 5
z/OS conventional MVS data sets 5
Mounting MVS data sets onto a client mount point 5
Creating z/OS conventional MVS data sets . . . 6
Serializing and sharing data sets. 6
NFS version 2 and version 3 statelessness. 7
NFS version 4 state 7
Name space and file system management. 8
z/OS NFS File System Type Selection 9
Specifying the path type prefix and the customer-configurable path resolution heuristic. . . 9
Implicit prefix support restrictions. 11
Server control files 13
Attributes data set 13
Exports data set 13
Mount handle data set. 14
Log data set 14
Tested clients for the z/OS NFS server 14
NFS protocol attributes for the z/OS NFS server 15
z/OS NFS server restrictions 15
Tested servers for the z/OS NFS client 16
z/OS NFS client restrictions. 17
WebNFS support 17
NFS version 3 and version 4 with TCP/IP protocols 18
Internet Protocol version 6 19
User-specified port range support 20
Dynamic addressing 20
Data transfer and conversion 21
Native ASCII support 21

Chapter 2. Creating z/OS conventional MVS data sets 23

Overriding data set creation attributes 23
Preparing to create an MVS file. 23
Naming MVS files 24
Creating physical sequential files 24
Creating direct access files 25
Creating PDSs and PDSEs 25
Creating a PDS or PDSE - mkdir dsntype(pds), dsntype(library) 25
Removing a PDS or PDSE - rm, rmdir 26
Accessing PDS or PDSE members 26
Updating or extending a PDS or PDSE member 27
Timing out while writing a PDS or PDSE member 27
Wildcard copy to a PDS or PDSE 27
Limitations of a PDS 27
Concurrent writes to a PDSE 27
Creating VSAM files 27
Exploiting SAM striped files. 28
Exploiting large format data sets 29
Exploiting data sets on extended address volumes 29

Chapter 3. Using z/OS conventional MVS data sets 31

Special MVS considerations 31
Selection of an MVS data storage format . . . 31
File size determination and time stamps. . . . 32
Ownership and permissions 32
File reading and writing 33
Case sensitivity–maplower, nomaplower. . . . 34
Selection of text or binary processing modes–text, binary 34
MVS prefix support 36
Number representation 36
Partial record identification 36

© Copyright IBM Corp. 1991, 2010

  - Access to migrated files–retrieve, noretrieve; wait, nowait 38
  - Access to migrated system-managed data sets 39
  - File handle refresh 39
- Mapping between the workstation and MVS file systems 40
  - File extension mapping 40
  - Mounting of MVS data sets onto a client mount point 41
  - Use of a PDS or PDSE as a directory 43
  - Use of multiple mount points 43
  - Data set serialization and sharing 44
  - NFS protocol 44
  - NFS file system attributes 44
- Delegating management of a file's resources to an NFS client 45

Chapter 4. Using z/OS UNIX System Services files 47

- z/OS UNIX file system 47
- POSIX compatibility 48
- NFS protocol 49
- Attributes specific to z/OS UNIX System Services 49
  - Synchronous write to a z/OS UNIX file for NFS version 2 protocol 49
  - Synchronous write to a z/OS UNIX file for NFS version 3 or 4 protocol 49
  - Authorization checking when writing to a z/OS UNIX file 50
  - HFSPREFIX site attribute 50
- Protecting your z/OS UNIX System Services files 50
- Accessing z/OS UNIX files from a client 51
  - Mount examples 51
- z/OS UNIX data transfer and conversion 52
  - Data transfer under the NFS version 4 protocol 52
  - Text or binary processing - NFS version 2 and 3 protocols 52
- Linking an MVS data set to a z/OS UNIX file system 53
  - Creating an external link 53
  - Displaying the contents on an external link 54
  - Deleting an external link 54
- Accessing symbolic links on z/OS NFS version 4 54
- UNIX look and feel 55
- NFS file system attributes 55
- Displaying and modifying remote file system access control lists 55
  - Remote ACL management restrictions: 56

Chapter 5. z/OS NFS file locking and access control 59

- Locking in NFS versions 2 and 3 (NFS server only) 59
  - Using Network Lock Manager (NLM) in NFS V2 and V3 59
  - Using Network Status Monitor (NSM) in NFS V2 and V3 60
- Locking in NFS version 4 61
  - Byte-range locking 61
  - Share reservations 61
  - Specifying a grace period for reclaiming locks 61
  - Listing locks held for a file 62
  - Releasing locks held for a file 62

Chapter 6. Commands and examples for AIX and UNIX clients 63

- Using commands on AIX 63
- Quick reference of AIX and UNIX commands 66
- Accessing z/OS UNIX file systems and z/OS conventional MVS files 67
  - Mvslogin command examples 68
  - Mount command examples using an MVS prefix 69
  - Mount command example: overriding server default attributes without an MVS prefix 70
- Displaying default and mount point attributes - showattr 72
- Unmounting and logging out of z/OS 75
  - Disconnecting your mount point - umount 75
  - Ending your z/OS session - mvslogout 76

Chapter 7. Commands and examples for z/OS NFS clients 77

- Using commands on the z/OS NFS client 77
  - Accessing z/OS 83
  - Mvslogin command examples 84
  - Mount command syntax and examples 85
  - Unmount command syntax and examples 92
  - Displaying client and server statistical information–nfsstat 94
  - Displaying server mount information–showmount 98
  - Displaying default and mount point attributes–showattr 99
  - Ending your z/OS session - mvslogout 101

Chapter 8. Initialization attributes for the z/OS NFS client 103

- Client attribute syntax 103
- Datacaching attribute 110
- Mount processing parameters and installation parameters 111
- NFS client translation support 112
- z/OS NFS client with z/OS NFS server 112

Chapter 9. Initialization attributes for the z/OS NFS server 115

- Attributes used for z/OS UNIX file access 115
- Multipliers 116
- Duplicate attributes 116
- Data set creation attributes syntax 116
- Processing attributes syntax 121
  - Timeout attributes 128
  - Retrieve attributes 129
  - Mapped keyword processing attribute 129
  - Native ASCII processing attributes 130
- Site attributes syntax 134

Part 2. Customization and Operations 151

Chapter 10. Customization 153

- Protecting your programs and files 153
  - Protecting the server control files 153
  - Setting up the z/OS NFS authorization 153
  - Protecting the file system on z/OS with the NFS V4 protocol 154
  - Protecting the file system on z/OS with the Security site attribute 158
  - Customizing installation security exits 163
  - Using UNIX style credentials for authentication 163
- Converting data 164
- Creating the conversion environment for Unicode Services 164
- Collecting NFS usage data 165
- Configuring the z/OS NFS client 167
  - Creating the PARMLIB statement for the client 167
  - Updating z/OS system data sets for the client 167
  - Allocating client log data sets 168
  - NFS Client with Multiple TCPIP stacks 168
  - Mounting remote file systems 168
  - Setting up reserved (privileged) ports 169
  - Configuring a secure z/OS NFS client 169
- Configuring the z/OS NFS server 171
  - Attributes data set 171
  - Exports data set 172
  - Checklist data set 181
  - Mount handle data sets 182
  - Lock data sets 182
  - Converting data between ASCII and EBCDIC - NFS V2 and V3 only 183
  - Updating z/OS system data sets for the server 185
  - Allocating the z/OS NFS server log data sets 185
  - Side file data set 185
  - Modifying tcpip.ETC.RPC and etc/rpc 185
  - Setting up a user-specified port range 186
  - Configuring a secure z/OS NFS server 188
  - Using dynamic client IP addressing 191
  - Terminal ID based restricted MVSLOGIN 191
  - SERVAUTH based restricted MVSLOGIN 192
  - Data Labeling 194
  - Using multiple TCP/IP stacks 195
- Installing the client enabling commands 198
  - Retrieving commands for AIX, Sun Solaris, and Linux 200
  - Porting the mvslogin, mvslogout, and showattr commands 203
  - Porting on different compilers and operating systems 204
- NFS v4 protocol name mapping 207
  - Inbound owner/owner_group processing 208
  - Outbound owner and owner_group attributes processing 209
  - Domain processing in NFSv4 Name Mapping 210

Chapter 11. Network File System operation 211

- Starting the z/OS NFS client 211
- Stopping the z/OS NFS client 212
- Starting component tracing for the z/OS NFS client 212
- Starting the z/OS NFS server 215
  - Starting multiple servers 217
- Stopping the z/OS NFS server 217
- Starting the z/OS NFS NSM and z/OS NFS NLM 218
- Starting component tracing for the z/OS NFS server 218
- Entering operands of the modify command for the z/OS NFS server 222
  - Addds operand 225
  - bufferusage operand 225
  - bufhigh operand 226
  - Cbsniff operand 226
  - DlyDTimeout operand 226
  - Exportfs operand 227
  - Freeds operand 227
  - Freeze operand 228
  - List operand 228
  - Mapfile operand 229
  - racf_cache_reset operand 229
  - Release operand 230
  - smf operand 231
  - Status operand 231
  - Swapldb operand 231
  - Swapmhdb operand 231
  - Unmount operand 232
  - Unmntall operand 232
  - Unmnthfs operand 232
  - Unmntmvs operand 233
  - v4delg operand 233
- Entering operands of the modify command for diagnosis 233
  - Debug operand 233
  - Flushlog operand 234
  - Listlock operand 234
  - Log operand 235
  - Smf operand 236
  - Sockhang operand 236
  - Switchlog operand 237
  - Version operand 237
- Displaying NFS trace information 238

Chapter 12. Installation-wide exit routines for the z/OS NFS server . . . 239

- Requirements for NFS 239
  - Sample link-edit JCL 240
  - Storage blocks of the server exits 240
- Login installation-wide exit 241
  - Requirements of the login exit 243
  - Options of the login exit 243
  - Structure of the login exit message 243
  - Contents of the login exit parameter list . . . 243
  - Login exit parameter list 244
  - Request codes to the login exit 244
  - Return codes from the login exit 245
  - System initialization routine of the login exit 245
  - Start of new user session routine of the login exit 245
  - User login request routine of the login exit . . 246
  - User logout request routine of the login exit . . 247
  - System termination routine of the login exit . . 247
- File security installation-wide exit 247
  - Requirements of the file security exit 250

Structure of the file security exit message . . . 250
Contents of the file security exit parameter list 250
File security exit parameter list 252
Request codes to the file security exit 252
Return codes from the file security exit 252
Validate allocate request routine of the file security exit 252
Validate write request routine of the file security exit 253
Validate read request routine of the file security exit 253
Return security permissions routine of the file security exit 254

Part 3. Performance Tuning 257

Chapter 13. Performance tuning in the NFS environment. 259
What is performance tuning? 259
How is performance characterized? 259
What is the NFS environment? 260
How to tune for performance 260
Impact of the NFS protocol on performance . . . 262

Chapter 14. Optimizing the NFS environment 265
Network performance tuning 265
NFS client system performance tuning 267
NFS server system performance tuning. 268
z/OS constraints 269

Chapter 15. Evaluating z/OS NFS performance 271
Evaluating throughput 271
Single process throughput 271
Multiple process throughput 272
Multiple client throughput 272
Evaluating NFS command response time 272
Evaluating CPU utilization 272
Collecting server usage data 273

Chapter 16. Tuning the z/OS NFS server 275
Data set creation attributes 275
Block size and record length 275
Record format 276
Data set organization and data set type. . . . 276
B37/D37/E37 ABEND handling 277
| NFS server cache monitoring and reporting . . 277
Processing attributes 277
Character translation 278
File size determination 278
Data set timeout specification 279
Accessing migrated files. 280
Asynchronous z/OS UNIX processing 280
Site attributes 281
Buffer usage and caching 281
Ordering out-of-sequence data. 282
Storage considerations 284
Subtasking 284

Chapter 17. Tuning the z/OS NFS client 287
Caching 287
Dynamicsizeadj. 287
Bufhigh 288
Biod 288
Readahead 288
Delaywrite 288
Vers 289
Wsize and rsize 289

Part 4. Diagnosis and Messages 291

Chapter 18. Diagnosis and reporting of problems 293
Correcting input errors 293
Using keywords to identify a problem 294
Component identification keyword 294
Release level keyword 295
Type-of-failure keyword 295
Service level keyword 297
Using z/OS component tracing 297
Component trace benefits 298
Using NFS server component trace PARMLIB member CTINFS*nn* 299
Using NFS client component trace PARMLIB member CTINFC*nn* 300
Capturing NFS Server component trace information in an SVC dump 304
Capturing NFS Client component trace information in an SVC dump 305
Using a z/OS component trace external writer 305
Setting up a dump data set for abnormal ends . . 311
Searching the IBM database for APARs and PTFs 312
Contacting the IBM Support Center 312
Diagnostic aids 313
First failure data capture 313
Errors and messages 314
Debug trace data capture 315
NFS client hang problem analysis 315
Environmental checklist 316

Chapter 19. Network File System messages 319
Server messages 319
Client messages 381
Client/Server Shared Messages 402
Messages from the client platform (AIX) 412

Chapter 20. Return codes 417

Chapter 21. Reason codes 427
Special reason codes (*xx* is 00-0F). 428
Reason codes from NFS Client or NFS Server modules (*xx* is 10-FF). 439
USS JR*cccc* reason codes (0000-0FFF). 439
Global reason codes (*yyyy* = 1000 - 3FFF) . . . 443

Module specific reason codes (*yyyy* = 4000 - 4FFF) . . . 447

Appendix A. File size value for MVS data sets . . . 449

Storage of the file size value . . . 449
System-managed PS, VSAM, and PDSE data sets . . . 449
Migrated system-managed data sets . . . 449
Non-system-managed, PDS, and direct data sets 450
How the file size value is generated . . . 450
Using fastfilesize to avoid read-for-size . . . 450
Using nofastfilesize . . . 451

Appendix B. Time stamps for MVS data sets . . . 453

Time stamps for system-managed VSAM and PS data sets . . . 453
Time stamps for non-system-managed PS and DA data sets . . . 453
Storage of time stamps . . . 453
Client program requirements . . . 454
Generating time stamps . . . 454
Time stamps for non-system-managed VSAM data sets . . . 454
Time stamps for PDSs and PDSEs . . . 454
Setting time stamps . . . 456

Appendix C. NFS server attributes 457

NFS file system attributes for MVS data sets . . . 457
NFS file system attributes for z/OS UNIX file systems . . . 458
NFS protocol attributes . . . 459

Appendix D. NSM (statd) protocol . . 463

Using supported NSM (statd) procedures . . . 463

Appendix E. NFS system server sample attribute table . . . 465

Appendix F. Sample exports data set 487

Appendix G. Sample startup procedures . . . 495

Sample z/OS NFS server startup procedures . . . 495
Sample z/OS NFS client startup procedures . . . 499

Appendix H. Retrieval of source code for client enabling commands . . . 501

Appendix I. PCNFSD protocol . . . 503

Accessing data with PCNFSD . . . 503
Accessing z/OS UNIX files . . . 503
Starting the PCNFSD server . . . 503
Using supported PCNFSD protocols . . . 504
Version 1 of the PCNFSD protocol . . . 504
Version 2 of the PCNFSD protocol . . . 504

Appendix J. SMF C and assembler header macros . . . 507

SMF C header macro GFSASSMF . . . 507
SMF assembler header macro GFSAUSMF . . . 515

Appendix K. Capturing diagnostic information using z/OS NFS log data sets and from other components . . . 523

Using log data sets . . . 523
Server log data sets . . . 523
Client log data sets . . . 524
Debug trace data capture . . . 525
z/OS NFS server debug trace capture . . . 525
z/OS NFS server DEBUG trace types . . . 526
z/OS NFS client debug trace capture . . . 526
Related component trace capture . . . 527
z/OS UNIX System Services activity trace . . . 527
z/OS hierarchical file system (HFS) physical file system activity trace . . . 527
z/OS TCP/IP activity trace . . . 527
AIX client activity trace . . . 528
SUN client activity trace . . . 529
z/OS dump . . . 529

Appendix L. GFSAMHDJ sample code for creating NFS mount handle data sets and lock data sets . . . 531

| ## Appendix M. Setting up NFS functions with Kerberos Support . . . 535

| Setting up a Linux Client/Server with NFS Version 4 Kerberos Support . . . 535
| Windows recommendations and limitations . . . 536
| Setting up a Kerberos Key Distribution Center . . 537

Appendix N. Accessibility . . . 541

Using assistive technologies . . . 541
Keyboard navigation of the user interface . . . 541
z/OS information . . . 541

Appendix O. Dotted decimal syntax diagrams . . . 543

Notices . . . 545

Programming interface information . . . 546
Policy for unsupported hardware . . . 546
Trademarks . . . 546

Glossary . . . 547

Index . . . 555

Figures

1. NFS client-server relationship 3
2. Examples of mounting MVS data sets on Windows, UNIX and Linux clients 43
3. Example of mounting an HFS or zFS file from a UNIX client 48
4. Syntax for commands 63
5. Displaying server attributes 73
6. Displaying mount point attributes 74
7. TSO MOUNT command syntax operands 85
8. TSO UNMOUNT command syntax operands 93
9. Displaying NFS Server and Client NFSv3 statistical information 95
10. Displaying NFS client rpc statistical information 95
11. Displaying NFS client NFS statistical information 96
12. Displaying all of the z/OS NFS Server (RPC, NFSv2, NFSv3, NFSv4) statistical information . 97
13. Displaying NFS mounted file system information 98
14. Displaying NFS mounted file system information with secure(upd) (Versions 2 and 3 protocol only) 98
15. Displaying NFS mounted file system information with public mountpoint (Version 4 protocol only) 98
16. Displaying attributes 100
17. Permission checking for the security(none) attribute 159
18. Permission checking for the security(exports) attribute 159
19. Permission checking for the security(saf) attribute 160
20. Permission checking for the security(safexp) attribute 161
21. Sample filesystype parmlib statement 167
22. Entry for a directory 174
23. Specifying the mount handle data set in the MVSNFS procedure 182
24. Specifying the lock data set in the MVSNFS procedure. 183
25. Modify /etc/services for mountd, mvsmount, pcnfsd, showattr, status, and nlockmgr . . . 186
26. Modify tcpip.profile for z/OS NFS server services 187
27. Retrieving the client enabling commands for AIX, Sun Solaris, and Linux. 201
28. Common source files 204
29. Summary of the modify command 223
30. Sample link-edit JCL for the NFS 240
31. Determining which login checking routines are used 242
32. Determining which file security checking routines are used 249
33. Displaying NFS client rpc and NFS statistical information 263
34. Sample network topology 266
35. Directory list comparison between DFSMS-managed and non-managed 279
36. Bufhigh utilization with percentsteal 281
37. Relationship between cachewindow and BIODs 283
38. Logicalcache utilization for cachewindows 283
39. z/OS NFS server component trace PARMLIB member CTINFS00. 300
40. z/OS NFS client component trace PARMLIB member CTINFC00 303
41. CTRACE Display Parameters panel 308
42. NFS Server Filtering Criteria 308
43. NFS Client Filtering Criteria 309
44. NFS system server sample attribute table 466
45. Sample exports data set 487
46. Sample z/OS NFS server startup procedures 496
47. Sample z/OS NFS client startup procedures 499
48. Retrieving source code for client enabling commands 501
49. SMF C header macro GFSASSMF 507
50. SMF assembler header macro GFSAUSMF 515
51. Sample code for creating mount handle data sets and lock data sets 531

© Copyright IBM Corp. 1991, 2010

Tables

1. Reference documents xiii
2. Mount statements based on the IMPPREFIX setting 11
3. View of NFS server capability 18
4. Breakdown of text and binary writes 34
5. Examples of mounting MVS data sets from clients 42
6. NFS procedures 44
7. Examples of the mount command for clients 51
8. Examples of the mvslogin command for clients 68
9. Examples of the showattr command for clients 72
10. Examples of the umount command for clients 76
11. Example of the mvslogout command for clients 76
12. Examples of the mvslogin command for clients 84
13. Attributes - z/OS NFS client 103
14. Client attributes. 103
15. Mount processing parameters 111
16. Installation parameters 111
17. z/OS NFS clients with non-z/OS based NFS servers 112
18. Attributes - z/OS NFS server 115
19. Data set creation attributes 116
20. Processing attributes 121
21. The mapped keyword and existing keywords 129
22. File tagging with Unicode Services active 131
23. File tagging with Unicode Services not active 133
24. Site attributes 134
25. Customizing NFS 153
26. UID, GID, SGID permission checking with security site attribute 162
27. z/OS server processing of a mount request 162
28. z/OS server processing of a file request 163
29. Shorthand for addresses with multiple zero bits 181
30. Shorthand for addresses in mixed IPv4 and IPv6 environments. 181
31. Modifying tcpip.ETC.RPC and etc/rpc 186
32. Files in the prefix.NFSTARB data set to download to clients 199
33. List of installation-wide exits 239
34. Format of login installation-wide exit routine parameter list 244
35. Request codes to the login exit. 245
36. Return codes from the login exit 245
37. Codes and fields for system initialization 245
38. Codes and fields for start of new user session 245
39. Codes and fields for user login request 246
40. Codes and fields for logout request 247
41. Codes and fields for system termination 247
42. Format of the parameter list for the file security installation-wide exit 251
43. Request codes to the file security exit 252
44. Return codes from the file security exit 252
45. Codes and fields for validate allocate request 252
46. Codes and fields for validate write request 253
47. Codes and fields for validate read request 253
48. Codes and fields for return security permissions 254
49. Network performance tuning symptom and action information 265
50. NFS client system performance tuning symptom and action information 267
51. z/OS constraints symptom and action information 269
52. Default block sizes for RAMAC 3 or ESS 2105 DASD 276
53. Client installation parameters for tuning 287
54. Mount parameters for performance and tuning 287
55. z/OS NFS FMIDs and release names 294
56. Summary of type-of-failure keywords 295
57. NFS symptom data 313
58. Dump content and storage areas 314
59. Diagnostic errors and messages 315
60. Messages - client operating system, NFS server, and NFS client. 319
61. Message format for the NFS server log data set 319
62. NFS client z/OS operators console message format 381
63. NFS client log data set message format 381
64. Common variables. 382
65. Message format for the NFS server or NFS client log data sets 402
66. Externalized return codes defined by the NFS version 2 protocol 417
67. Externalized return codes defined by the NFS version 3 protocol 417
68. Externalized return codes defined by the NFS version 4 protocol 418
69. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 2 return codes . . . 420
70. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 3 return codes . . . 422
71. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 4 return codes . . . 424
72. Special NFS reason codes 428
73. Parsing error (when reason code is 6E01*xxxx*) 438
74. Parsing error (when reason code is from 6E0111yy to 6E0133yy. 439
75. NFS reason codes that match USS JR*cccc* reason codes (0000-0FFF). 439
76. NFS client global reason codes (1000 - 3FFF) 443
77. Reason codes for module GFSCVNAT (xx = 12) 447
78. Reason codes for module GFSCVMNT (xx = 18) 447
79. Time stamp sources for PDS and PDSE members 455
80. Time stamp sources for PDS and PDSE data sets (directories) 455

© Copyright IBM Corp. 1991, 2010

81. File system values to get dynamic file system information 457
82. File system values to get static file system information 457
83. File system values to retrieve POSIX information 457
84. File system values to get static file system information 458
85. File system values to retrieve POSIX information 458
86. NFS Version 4 Attributes 459

About this document

This document provides users, system programmers, and operators with information about using, customizing, operating, tuning, and diagnosing the z/OS® Network File System (z/OS NFS).

For information about the accessibility features of z/OS, for users who have a physical disability, see Appendix N, “Accessibility,” on page 541.

Required product knowledge

To use this document effectively, you should be familiar with the IBM multiple virtual system (MVS) as a component of the z/OS operating system, the IBM Time Sharing Option (TSO), and their commands. In addition, you should be familiar with System Modification Program/Extended (SMP/E) and the basic concepts of the NFS protocol and networking (Transmission Control Protocol/Internet Protocol (TCP/IP)).

Where to find more information

Where necessary, this document references information in other documents, using the shortened version of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

Access to documents

Table 1 contains additional reference information.

Table 1. Reference documents

Title	Order Number
AIX Commands Reference, Volume 1	SC23-2537
AIX Commands Reference, Volume 2	SC23-2538
AIX Commands Reference, Volume 3	SC23-2539
AIX Commands Reference, Volume 4	SC23-2539
AIX Commands Reference, Volume 5	SC23-2639
AIX General Concepts and Procedures for RISC System/6000	GC23-2202
Character Data Representation Architecture Overview	GC09-2207
Character Data Representation Architecture Reference and Registry	SC09-2190
SMP/E Reference	SA22-7772
SMP/E User's Guide	SA22-7773
z/OS and Software Products DVD Collection	SK3T-4271
z/OS Collection	SK3T-4271
z/OS Communications Server: IP Configuration Reference	SC31-8776
z/OS DFSMS Introduction	SC26-7397
z/OS DFSMS Macro Instructions for Data Sets	SC26-7408
z/OS Migration	GA22-7499
z/OS DFSMS Using the New Functions	SC26-7473
z/OS DFSMS Using Data Sets	SC26-7410

© Copyright IBM Corp. 1991, 2010

Table 1. Reference documents (continued)

Title	Order Number
z/OS DFSMSdfp Advanced Services	SC26-7400
z/OS DFSMSdfp Diagnosis	GY27-7618
z/OS DFSMSdfp Storage Administration	SC26-7402
z/OS DFSMShsm Diagnosis	GC52-1083
z/OS Information Roadmap	SA22-7500
z/OS MVS Installation Exits	SA22-7593
z/OS MVS JCL Reference	SA22-7597
z/OS MVS Programming: Authorized Assembler Services Guide	SA22-7608
z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN	SA22-7609
z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG	SA22-7610
z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU	SA22-7611
z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO	SA22-7612
z/OS MVS System Codes	SA22-7626
z/OS MVS Programming: Authorized Assembler Services Guide	SA22-7608
z/OS MVS System Management Facilities (SMF)	SA22-7630
z/OS MVS System Messages, Vol 1 (ABA-AOM)	SA22-7631
z/OS MVS System Messages, Vol 2 (ARC-ASA)	SA22-7632
z/OS MVS System Messages, Vol 3 (ASB-BPX)	SA22-7633
z/OS MVS System Messages, Vol 4 (CBD-DMO)	SA22-7634
z/OS MVS System Messages, Vol 5 (EDG-GFS)	SA22-7635
z/OS MVS System Messages, Vol 6 (GOS-IEA)	SA22-7636
z/OS MVS System Messages, Vol 7 (IEB-IEE)	SA22-7637
z/OS MVS System Messages, Vol 8 (IEF-IGD)	SA22-7638
z/OS MVS System Messages, Vol 9 (IGF-IWM)	SA22-7639
z/OS MVS System Messages, Vol 10 (IXC-IZP)	SA22-7640
z/OS Security Server RACF Security Administrator's Guide	SA22-7683
z/OS Security Server RACF System Programmer's Guide	SA22-7681
z/OS Security Server RACF Callable Services	SA22-7691
z/OS Integrated Security Services Network Authentication Service Administration	SC24-5926
z/OS Integrated Security Services Network Authentication Service Programming	SC24-5927
z/OS TSO/E User's Guide	SA22-7794
z/OS Unicode Services User's Guide and Reference	SA22-7649
z/OS UNIX System Services File System Interface Reference	SA22-7808
z/OS UNIX System Services Messages and Codes	SA22-7807
z/OS UNIX System Services Programming: Assembler Callable Services Reference	SA22-7803
z/OS UNIX System Services Programming Tools	SA22-7805
z/OS UNIX System Services Command Reference	SA22-7802
z/OS UNIX System Services User's Guide	SA22-7801
z/OS UNIX System Services Planning	GA22-7800

Access to softcopy documents on CD-ROM and DVD collections and the Internet

This book will also be available on the following collections the next time they are updated.

z/OS V1Rx Collection, SK3T-4269

z/OS V1Rx and Software Products DVD Collection, SK3T-4271

This book will also be available in BookManager® and PDF format on the z/OS Internet library at:

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

Using a BookManager READ program, you can view and find information quickly in BookManager documents in a variety of environments. You can view the books directly from a CD-ROM or DVD, or copy the books to a shared workstation or local area network (LAN) server. You can also transfer the books to your host system or view them on the Internet. For instance, the z/OS product includes both BookManager READ for host viewing and BookManager BookServer, which allows you to access and read books over an Internet or intranet connection using an HTML browser. From CD-ROM or DVD you can use any supported IBM BookManager reader, such as the IBM Softcopy Reader. For more information, see:

`http://www.ibm.com/servers/eserver/zseries/softcopy/`

You can view and print PDF files using an Adobe Acrobat Reader available free on the Internet at:

`http://www.adobe.com/prodindex/acrobat/`

How to look up message explanations with LookAt

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

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

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX® System Services running OMVS).
- Your Windows® workstation. You can install code to access IBM message explanations on the *z/OS Collection* (SK3T-4271), using LookAt from a Windows DOS command line.
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux® handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

You can obtain code to install LookAt on your host system or Windows workstation from a disk on your *z/OS Collection* (SK3T-4271), or from the LookAt Web site (click **Download**, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

How to read syntax diagrams

Throughout this library, diagrams are used to illustrate the programming syntax. Keyword parameters are parameters that follow the positional parameters. Unless otherwise stated, keyword parameters can be coded in any order. The following list tells you how to interpret the syntax diagrams:

- Read the diagrams from left-to-right, top-to-bottom, following the main path line. Each diagram begins on the left with double arrowheads and ends on the right with two arrowheads facing each other.

```
►►──┤ Syntax Diagram ├──────────────────────────────────────────────►◄
```

- If a diagram is longer than one line, each line to be continued ends with a single arrowhead and the next line begins with a single arrowhead.

```
►►──┤ First Line ├──────────────────────────────────────────────────►◄


►►──┤ Second Line ├─────────────────────────────────────────────────►◄


►►──┤ Last Line ├───────────────────────────────────────────────────►◄
```

- Required keywords and values appear on the main path line. You must code required keywords and values.

```
►►──REQUIRED_KEYWORD────────────────────────────────────────────────►◄
```

If several mutually exclusive required keywords or values exist, they are stacked vertically in alphanumeric order.

```
►►──┬─REQUIRED_KEYWORD_OR_VALUE_1─┬─────────────────────────────────►◄
    └─REQUIRED_KEYWORD_OR_VALUE_2─┘
```

- Optional keywords and values appear below the main path line. You can choose not to code optional keywords and values.

```
►►──┬─────────┬─────────────────────────────────────────────────────►◄
    └─KEYWORD─┘
```

If several mutually exclusive optional keywords or values exist, they are stacked vertically in alphanumeric order below the main path line.

```
►►──┬──────────────────────┬────────────────────────────────────────►◄
    ├─KEYWORD_OR_VALUE_1─┤
    └─KEYWORD_OR_VALUE_2─┘
```

- An arrow returning to the left above a keyword or value on the main path line means that the keyword or value can be repeated. The comma means that each keyword or value must be separated from the next by a comma.

```
       ┌─,──────────────────┐
       ▼                    │
►►─────REPEATABLE_KEYWORD───┴──────────────────────────────────────────►◄
```

- An arrow returning to the left above a group of keywords or values means more than one can be selected, or a single one can be repeated.

```
►►──┬──────────────────────────────────────────┬────────────────────────►◄
    │   ┌─,──────────────────────────────────┐ │
    │   ▼                                    │ │
    └───┬─REPEATABLE_KEYWORD_OR_VALUE_1─┬────┴─┘
        └─REPEATABLE_KEYWORD_OR_VALUE_2─┘
```

- A word in all upper case is a keyword or value you must spell exactly as shown. In this example, you must code ***KEYWORD***.

```
►►──KEYWORD────────────────────────────────────────────────────────────►◄
```

If a keyword or value can be abbreviated, the abbreviation is discussed in the text associated with the syntax diagram.

- If a diagram shows a character that is not alphanumeric (such as parentheses, periods, commas, and equal signs), you must code the character as part of the syntax. In this example, you must code ***KEYWORD=(001,0.001)***.

```
►►──KEYWORD=(001,0.001)────────────────────────────────────────────────►◄
```

- If a diagram shows a blank space, you must code the blank space as part of the syntax. In this example, you must code ***KEYWORD=(001 FIXED)***.

```
►►──KEYWORD=(001 FIXED)────────────────────────────────────────────────►◄
```

- Default keywords and values appear above the main path line. If you omit the keyword or value entirely, the default is used.

```
    ┌─DEFAULT─┐
►►──┼─────────┼────────────────────────────────────────────────────────►◄
    └─KEYWORD─┘
```

- A word in all lower case italics is a *variable*. Where you see a variable in the syntax, you must replace it with one of its allowable names or values, as defined in the text.

```
►►──variable───────────────────────────────────────────────────────────►◄
```

- References to syntax notes appear as numbers enclosed in parentheses above the line. Do not code the parentheses or the number.

```
               (1)
►►──KEYWORD────────────────────────────────────────────────────────────►◄
```

Notes:

1 An example of a syntax note.

- Some diagrams contain *syntax fragments*, which serve to break up diagrams that are too long, too complex, or too repetitious. Syntax fragment names are in mixed case and are shown in the diagram and in the heading of the fragment. The fragment is placed below the main diagram.

```
►►──┤ Reference to Syntax Fragment ├─────────────────────────────────►◄
```

Syntax Fragment:

```
├──1ST_KEYWORD,2ND_KEYWORD,3RD_KEYWORD──────────────────────────────┤
```

Related protocol specifications

IBM is committed to industry standards. The internet protocol suite is still evolving through Requests for Comments (RFC). New protocols are being designed and implemented by researchers, and are brought to the attention of the internet community in the form of RFCs. Some of these are so useful that they become a recommended protocol. That is, all future implementations for TCP/IP are recommended to implement this particular function or protocol. These become the *de facto* standards on which the TCP/IP protocol suite is built.

The Network File System (NFS) is implemented as a set of RPC procedures that use External Data Representation (XDR) encoding to pass arguments between clients and servers. The NFS is based on the following RFCs.

Internet Protocol	RFC 791, J.B. Postel
NFS: Network File System Version 2 Protocol Specification	RFC 1094, Sun Microsystems, Incorporated
NFS: Network File System Version 3 Protocol Specification	RFC 1813, Sun Microsystems, Incorporated
NFS: Network File System Version 4 Protocol Specification	RFC 3530, Sun Microsystems, Incorporated
Open Group Technical Standard Protocols for Interworking: XNFS, Version 3W	Document Number: C702
RPC: Remote Procedure Call Protocol Specification Version 2	RFC 1057, Sun Microsystems Incorporated
RPC: Remote Procedure Call Protocol Specification Version 2	RFC 1831, R. Srinivasan
User Datagram Protocol	RFC 768, J.B. Postel
WebNFS Client Specification	RFC 2054, B. Callaghan
WebNFS Server Specification	RFC 2055, B. Callaghan
XDR: External Data Representation Standard	RFC 1014, Sun Microsystems, Incorporated
XDR: External Data Representation Standard	RFC 1832, R. Srinivasan
Generic Security Service Application Program Interface, Version 2	RFC 2078, J.Linn, OpenVision Technologies
RPCSEC_GSS Protocol Specification	RFC 2203, M. Eisler, A.Chiu, L. Ling
The Kerberos Version 5 GSS-API Mechanism	RFC 1964, J.Linn, OpenVision Technologies
The Kerberos Network Authentication Service (V5)	RFC 1510, J. Kohl, Digital Equipment Corporation, C. Neuman, ISI

For more information about Request for Comments (RFC), see the Internet Engineering Task Force (IETF) home page:

`http://www.ietf.org/`

The z/OS Basic Skills Information Center

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

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

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

To access the z/OS Basic Skills Information Center, open your Web browser to the following Web site, which is available to all users (no login required): http://publib.boulder.ibm.com/infocenter/zoslnctr/v1r7/index.jsp

How to send your comments to IBM

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

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

1. Send an email to mhvrcfs@us.ibm.com
2. Visit the Contact z/OS web page at http://www.ibm.com/systems/z/os/zos/webqs.html
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   U.S.A.
4. Fax the comments to us as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:

- Your name and address
- Your email address
- Your telephone or fax number
- The publication title and order number:
  z/OS Network File System Guide and Reference
  SC26-7417-10
- The topic and page number related to your comment
- The text of your comment.

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

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

If you have a technical problem

Do not use the feedback methods listed above. Instead, do one of the following:

- Contact your IBM service representative
- Call IBM technical support
- Visit the IBM zSeries support web page at http://www.ibm.com/systems/z/support/

© Copyright IBM Corp. 1991, 2010

Summary of changes

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

You may notice changes in the style and structure of some content. For example, text that has a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

The "Readers' Comments - We'd Like to Hear from You" section at the back of this publication has been replaced with a new section "How to send your comments to IBM" on page xxi. The hardcopy mail-in form has been replaced with a page that provides information appropriate for submitting readers comments to IBM.

Summary of changes for SC26-7417-10, z/OS Version 1 Release 12

This document contains information that was previously presented in *z/OS Network File System Guide and Reference*, SC26-7417-10. The following sections summarize the changes to that information.

New information

This edition includes new information:

- The **-P** *passphrase* parameter has been added to the **mvslogin** command (see "Using commands on the z/OS NFS client" on page 77).
- The z/OS NFS server can generate SMF record type 42 subtype 26 (see "Collecting NFS usage data" on page 165).
- The [, **none** | **all** | *subtype_list*] values have been added to the **MODIFY mvsnfs** command's **SMF** parameter (see "Smf operand" on page 236).
- The **-s**, **-2**, **-3**, and **-4** options have been added to the **nfsstat** command. There have also been changes to using the **nfsstat -n**, **-r**, and **-z** options (see "Displaying client and server statistical information–nfsstat" on page 94).
- NFS Server site attribute **bufhigh** has been modified so that you can specify when to issue a data buffer utilization alert message (see the bufhigh description in Table 24 on page 134)
- The **cbsniff** operand has been added to the **MODIFY mvsnfs** command (see "Cbsniff operand" on page 226)
- The following messages have been added to Chapter 19, "Network File System messages," on page 319:
  - GFSA383I
  - GFSA3841
  - GFSA385I
  - GFSA386E
  - GFSA387I
  - GFSA788I
  - GFSA805E
  - GFSA806I
  - GFSA872E
  - GFSA1034E
  - GFSC857I

© Copyright IBM Corp. 1991, 2010

Changed information

This edition includes changed information:

- The following messages have been changed in Chapter 19, "Network File System messages," on page 319:
  - GFSA954I
  - GFSA956I
  - GFSA959I
  - GFSA972I
  - GFSC840I
- The descriptions for special reason codes *xx*=05 in Table 72 on page 428 have been expanded.

Summary of changes for SC26-7417-10, z/OS Version 1 Release 11 as updated April 2010

This document contains information that was previously presented in *z/OS Network File System Guide and Reference*, SC26-7417-09. The following section summarizes the changes to that information.

Changed information

This edition includes changed information:

- The following messages have been changed in Chapter 19, "Network File System messages," on page 319:
  - GFSA1030W has been changed to GFSA1030E
  - GFSA1033W has been changed to GFSA1033E

Summary of changes for SC26-7417-09, z/OS Version 1 Release 11

This document contains information that was previously presented in *z/OS Network File System Guide and Reference*, SC26-7417-08. The following sections summarize the changes to that information.

New information

This edition includes new information:

- The **startfail** parameter has been added to the **start** command (see "Starting the z/OS NFS server" on page 215).
- NFS V4 Delegation support for MVS data sets (see "Delegating management of a file's resources to an NFS client" on page 45)
- Enhanced NFS file system type selection prefix support
- The following messages have been added to Chapter 19, "Network File System messages," on page 319:
  - GFSA444E
  - GFSA445E
  - GFSA446E
  - GFSA462E
  - GFSA463E
  - GFSA464I
  - GFSA476E
  - GFSA790I
  - GFSA796I
  - GFSA797I
  - GFSA1030W

- GFSA1031I
  - GFSA1032E
  - GFSA1033E
  - GFSC859E
  - GFSC860W
- The following reason codes have been added to Table 75 on page 439
  - 003D
  - 0051
  - 0058
  - 0078
  - 007A
  - 0090
  - 00A9
  - 00B7
  - 00C5
  - 0115
  - 014E
  - 0291
  - 029E
  - 0351
  - 03A5
  - 03BD
  - 03CA
  - 0421
- The following reason codes have been added to Table 76 on page 443
  - 100B
  - 100C
  - 100D
  - 100E
  - 100F
  - 1010
  - 1011
  - 1012
  - 1013
  - 1014
- Japanese language National Language Support (NLS) for NFS Client Console messages.

Changed information

This edition includes changed information:

- The following messages have been changed in Chapter 19, “Network File System messages,” on page 319:
  - GFSA326E
  - GFSA792I
  - GFSA866I
  - GFSC110E
  - GFSC284E has been changed to GFSC284I

  - GFSA1033E has been changed to GFSA1033W
- The descriptions for special reason codes *xx*=05 in Table 72 on page 428 have been expanded.

Deleted information

This edition has deleted the following information:

- The following messages have been deleted from Chapter 19, “Network File System messages,” on page 319:
  - GFSA926I
  - GFSA928I
  - GFSA941I
- Removed NFS V4 Mount Symlink restriction

Summary of changes for SC26-7417-08, z/OS Version 1 Release 10

This document contains information that was previously presented in *z/OS Network File System Guide and Reference*, SC26-7417-07. The following sections summarize the changes to that information.

New information

This edition includes new information:

- Addition of z/Linux to the list of tested/supported platforms.
- 64-bit addressing support for the NFS Client Utilities (i.e. mvslogin, mvslogout, showattr) for non-z/OS platform usage.
- Japanese language National Language Support (NLS) for NFS Server Console messages.
- “Displaying and modifying remote file system access control lists” on page 55 has been added.
- Chapter 5, “z/OS NFS file locking and access control,” on page 59 has been added.
- The keyword **-norpcbind** has been added to the **mvslogin**, **mvslogout**, and **showattr** commands in Chapter 6, “Commands and examples for AIX and UNIX clients,” on page 63 and the **mvslogin**, **mvslogout**, **showattr**, and **showmount**commands in “Using commands on the z/OS NFS client” on page 77.
- “Setting up reserved (privileged) ports” on page 169 has been added to describe how to reserve ports for use by the z/OS client.
- “Sockhang operand” on page 236 has been added to describe how to tell the server to create a dump when it detects a potential socket hang condition.
- “Filtering NFS ctrace records in IPCS” on page 307 has been updated to describe a new NFS Server Filtering Criteria panel (shown in Figure 42 on page 308), a new NFS Client Filtering Criteria panel (shown in Figure 43 on page 309), and to add a new section on processing traces in batch mode.
- “Configuring a secure z/OS NFS client” on page 169 has been added.
- “NFS client hang problem analysis” on page 315 has been added to describe how to resolve NFS client hang situations.

Changed information

This edition includes changed information:

- The description of the secure attribute in “Client attribute syntax” on page 103 has been updated to include the krb5, krb5i, and krb5p values.

- The description of the secure attribute in Table 15 on page 111 has been updated to include the krb5, krb5i, and krb5p values.
- The description of the nfstasks attribute in “Site attributes syntax” on page 134 has been updated to include values to specify the number of transport subtasks that handle TCP and UDP network requests.
- Appendix E, “NFS system server sample attribute table,” on page 465 has been updated to show the latest sample file.

Deleted information

This edition has deleted the following information:

- TBD
- The following messages have been deleted from Chapter 19, “Network File System messages,” on page 319:
  - TBD

Summary of changes for SC26-7417-07, z/OS Version 1 Release 9

This document contains information that was previously presented in *z/OS Network File System Guide and Reference*, SC26-7417-06. The following sections summarize the changes to that information.

New information

This edition includes new information:

- NFS Client NFS version 4 protocol support:
  - “Tested servers for the z/OS NFS client” on page 16
  - “Name space and file system management” on page 8
  - “Accessing z/OS UNIX files from a client” on page 51, “Using commands on AIX” on page 63, and “Mount command syntax and examples” on page 85 have been updated to indicate a restriction when using NFS protocol version 4 on mount path names for UNIX files.
  - “Additional mount command examples” on page 89
  - Figure 9 on page 95
  - Figure 15 on page 98
- “Invoking the Mount command on the z/OS platform” on page 88 has been added to describe the different ways of invoking the Mount command in z/OS.
- A new **public** option has been added to “Client attribute syntax” on page 103. The public option forces the use of the public file handle when connecting to the NFS server.
- A new **stringprep** option has been added to “Client attribute syntax” on page 103. The stringprep option specifies whether the z/OS NFS Client is to enable or disable stringprep normalization. Stringprep normalization is the NFS version 4 globalization function for converting inbound strings to UTF-8 format.
- “Terminal ID based restricted MVSLOGIN” on page 191 describes how MVSLOGIN to the z/OS NFS server can be controlled based on an NFS client's IP address.
- “SERVAUTH based restricted MVSLOGIN” on page 192 describes how MVSLOGIN to the z/OS NFS server can be controlled using the SERVAUTH class.
- “Data Labeling” on page 194 has been added to describe the z/OS NFS server support for the RACF Data Labeling option MLNAMES (also known as name-hiding).

- “Starting component tracing for the z/OS NFS client” on page 212 contains new TRACE CT command options for the new client trace record types and new option shorthand values, and ability to turn off individual options, and the new default trace buffer.
- “Starting component tracing for the z/OS NFS server” on page 218 contains new TRACE CT command options for the new server trace record types and new option shorthand values, and ability to turn off individual options, and the new default trace buffer.
- A new operator command "MODIFY mvsnfs,ADDDS" for providing a replacement lock data set or mount handle data set to NFS is described in “Addds operand” on page 225.
- A new operator command "MODIFY mvsnfs,FREEDS" for removing an existing lock data set or mount handle data set is described in “Freeds operand” on page 227.
- A new operator command "MODIFY mvsnfs,SWAPMHDB" for swapping the Mount Handle Database (MHDB) data sets is described in “Swapmhdb operand” on page 231.
- A new operator command "MODIFY mvsnfs,SWAPLDB" for swapping the Lock Database (LDB) data sets is described in “Swapldb operand” on page 231.
- The MVS DISPLAY TRACE command can now be used to display component trace status and active options for the NFS server or NFS client, as described in “Displaying NFS trace information” on page 238.
- New reason codes have been added to Table 72 on page 428 and Table 73 on page 438.
- New messages have been added to Chapter 19, “Network File System messages,” on page 319.

Changed information

This edition includes changed information:

- Additional resource constraint relief below the 16-MB line has been achieved, helping to prevent out of storage conditions.
- The name of the NFS makefile has been corrected throughout the book.
- “Ownership and permissions” on page 32 has been updated to clarify how UNIX UID and GID file attributes are set.
- Figure 9 on page 95 has been updated to show NFS V4 client information.
- Figure 10 on page 95 has been updated to show NFS V4 client information.
- Figure 14 on page 98 has been updated to show nfsstat -m output for the Version 3 protocol.
- Figure 15 on page 98 has been updated to show nfsstat -m output for the Version 4 protocol.
- In Table 14 on page 103, the default for the **disablella** client attribute has been changed to **N**. That is, the default is now to enable Lookup Look-Aside caching.
- NFS V4 Server RPCSEC_GSS single stack support limitation removed:
  - “Using multiple TCP/IP stacks” on page 195 has been updated to indicate that each NFS server can now support multiple TCP/IP stacks.
- Numerous messages have been changed in Chapter 19, “Network File System messages,” on page 319.
- Chapter 21, “Reason codes,” on page 427 has been updated with new and changed reason codes.

- Appendix E, "NFS system server sample attribute table," on page 465 has been updated.
- Appendix F, "Sample exports data set," on page 487 has been updated.
- A note has been added to Appendix G, "Sample startup procedures," on page 495 indicating that the SYSTCPD DD statement is not required for either the NFS server or the NFS client, unless a custom configuration is required for NFS.

Deleted information

This edition has deleted the following information:

- The MODIFY FLUSHCTR operator command has been deleted from Chapter 11, "Network File System operation," on page 211. This function will be done automatically when the Component Trace external writer is stopped.
- The following messages have been deleted from Chapter 19, "Network File System messages," on page 319:
  - GFSA350E (replaced by GFSA1017E)
  - GFSA351E (replaced by GFSA1018E)
  - GFSA374I (replaced by GFSA1017E)
  - GFSC215E

Summary of changes for SC26-7417-06, z/OS Version 1 Release 8

This document contains information that was previously presented in *z/OS Network File System Guide and Reference*, SC26-7417-05. The following sections summarize the changes to that information.

Some actions need to be taken before using the Network File System (NFS) in the z/OS V1R8 release; for information on those actions, see the NFS section of *z/OS Migration*.

New information

This edition includes new information:

- Component trace support for the z/OS NFS client has been added to "Starting component tracing for the z/OS NFS client" on page 212 and to "Using z/OS component tracing" on page 297.
- "Exports data set" on page 172 has been updated with new ways to specify client ids, including netgroup definitions, wildcard characters, and IPv6 address notations. Separator characters for client ids and security values have been changed from colons (:) to vertical bars (|).
- The LIST operand of the NFS server's modify command now lists the hostnames of clients with active mounts to a given mount point. See "List operand" on page 228 for more information.
- Support for the NFS version 4 protocol's globalization stringprep and UTF-8 encoding functions has been added.
- The following site attributes have been added to Table 24 on page 134:
  - **stringprep|nostringprep**, to enable or disable the NFS version 4 globalization function for string conversion.
- The following client attributes have been added to Table 14 on page 103:
  - **accesschk**, to specify whether the z/OS NFS client or NFS server is to check that a user has the requested access to a file or directory.
- A list of the FMIDs for each release of NFS has been added in Table 55 on page 294.

- Linux client/server setup steps are added in "Setting up a Linux Client/Server with NFS Version 4 Kerberos Support" on page 535.
- New messages have been added to Chapter 19, "Network File System messages," on page 319.

Changed information

This edition includes changed information:

- The command for stopping the NFS Client has been changed from `f stop mvsnfsc` to `f omvs,stoppfs=NFS`. See "Stopping the z/OS NFS client" on page 212 for details.
- The default value for the **delim** client attribute has been changed from *binary* to *na*. To obtain the *binary* value's function it must be explicitly specified on the **delim** attribute in the NFS client parameter list in the BPXPRMxx parmlib member.
- Numerous messages have been changed in Chapter 19, "Network File System messages," on page 319.

Deleted information

This edition has deleted the following information:

- Checklist data sets are no longer supported in z/OS V1R8; their function has been incorporated into the exports data set using new parameters described in "Exports data set" on page 172.

Part 1. Using z/OS Network File System

Chapter 1. Introduction 3
Overview 3
z/OS UNIX files 4
z/OS UNIX advantages. 4
NFS protocol compliance 5
Crossing between file systems–NFS server . . . 5
z/OS conventional MVS data sets 5
Mounting MVS data sets onto a client mount point 5
Creating z/OS conventional MVS data sets . . . 6
Serializing and sharing data sets. 6
NFS version 2 and version 3 statelessness. 7
NFS version 4 state 7
Name space and file system management. 8
z/OS NFS File System Type Selection 9
Specifying the path type prefix and the customer-configurable path resolution heuristic. . 9
Valid mount path specifications. 10
Implicit prefix support restrictions. 11
Server control files 13
Attributes data set 13
Exports data set 13
Mount handle data set. 14
Log data set 14
Tested clients for the z/OS NFS server 14
NFS protocol attributes for the z/OS NFS server 15
z/OS NFS server restrictions 15
Tested servers for the z/OS NFS client 16
z/OS NFS client restrictions. 17
WebNFS support 17
NFS version 3 and version 4 with TCP/IP protocols 18
Internet Protocol version 6 19
User-specified port range support 20
Dynamic addressing 20
Data transfer and conversion 21
Native ASCII support 21

Chapter 2. Creating z/OS conventional MVS data sets. 23
Overriding data set creation attributes 23
Preparing to create an MVS file. 23
Naming MVS files 24
Restrictions on using alias names for MVS files 24
Creating physical sequential files 24
Creating direct access files 25
Creating PDSs and PDSEs 25
Creating a PDS or PDSE - mkdir dsntype(pds), dsntype(library) 25
Removing a PDS or PDSE - rm, rmdir 26
Accessing PDS or PDSE members 26
Updating or extending a PDS or PDSE member 27
Timing out while writing a PDS or PDSE member 27
Wildcard copy to a PDS or PDSE 27
Limitations of a PDS 27
Concurrent writes to a PDSE 27
Creating VSAM files 27
Exploiting SAM striped files. 28
Exploiting large format data sets 29
Exploiting data sets on extended address volumes 29

Chapter 3. Using z/OS conventional MVS data sets. 31
Special MVS considerations 31
Selection of an MVS data storage format . . . 31
File size determination and time stamps. . . . 32
Ownership and permissions 32
File reading and writing 33
Random access to files. 33
Cached data writing 33
Case sensitivity–maplower, nomaplower. . . . 34
Selection of text or binary processing modes–text, binary 34
Binary processing mode 34
Text processing mode 34
MVS prefix support 36
Number representation 36
Partial record identification 36
Symptoms of GFSA824W/GFSA825W messages 37
Access to migrated files–retrieve, noretrieve; wait, nowait 38
Access to migrated system-managed data sets. . 39
File handle refresh 39
Mapping between the workstation and MVS file systems. 40
File extension mapping 40
Mounting of MVS data sets onto a client mount point 41
Mount examples. 42
Variants of the mount command 42
Use of a PDS or PDSE as a directory 43
Use of multiple mount points 43
Data set serialization and sharing 44
NFS protocol 44
NFS file system attributes 44
Delegating management of a file's resources to an NFS client 45

Chapter 4. Using z/OS UNIX System Services files. 47
z/OS UNIX file system 47
POSIX compatibility 48
NFS protocol 49
Attributes specific to z/OS UNIX System Services 49
Synchronous write to a z/OS UNIX file for NFS version 2 protocol 49
Synchronous write to a z/OS UNIX file for NFS version 3 or 4 protocol. 49
Authorization checking when writing to a z/OS UNIX file 50

© Copyright IBM Corp. 1991, 2010

HFSPREFIX site attribute 50
Protecting your z/OS UNIX System Services files . 50
Accessing z/OS UNIX files from a client 51
Mount examples. 51
z/OS UNIX data transfer and conversion 52
Data transfer under the NFS version 4 protocol 52
Text or binary processing - NFS version 2 and 3 protocols 52
Linking an MVS data set to a z/OS UNIX file system 53
Creating an external link 53
Displaying the contents on an external link. . . 54
Deleting an external link 54
Accessing symbolic links on z/OS NFS version 4. . 54
UNIX look and feel. 55
NFS file system attributes 55
Displaying and modifying remote file system access control lists 55
Remote ACL management restrictions: 56
AIX restrictions 56
z/OS client restrictions 57
z/OS server restrictions 57

Chapter 5. z/OS NFS file locking and access control. 59
Locking in NFS versions 2 and 3 (NFS server only) 59
Using Network Lock Manager (NLM) in NFS V2 and V3 59
Monitored locks 59
Non-monitored locks 59
Specifying a grace period for reclaiming locks 60
Listing locks held for a file 60
Releasing locks held for a file 60
Using Network Status Monitor (NSM) in NFS V2 and V3 60
Locking in NFS version 4. 61
Byte-range locking 61
Share reservations 61
Specifying a grace period for reclaiming locks . . 61
Listing locks held for a file 62
Releasing locks held for a file 62

Chapter 6. Commands and examples for AIX and UNIX clients 63
Using commands on AIX 63
Quick reference of AIX and UNIX commands . . . 66
Accessing z/OS UNIX file systems and z/OS conventional MVS files 67
Mvslogin command examples 68
“Permission denied” message 69
Mount command examples using an MVS prefix 69
Overriding default attributes 70
Mount command example: overriding server default attributes without an MVS prefix . . . 70
Getting authorization to access files 71
Saving of mount points 71
Automatic timed logout - logout attribute . . 71
Displaying default and mount point attributes - showattr 72
Unmounting and logging out of z/OS 75
Disconnecting your mount point - umount . . . 75
Ending your z/OS session - mvslogout 76

Chapter 7. Commands and examples for z/OS NFS clients 77
Using commands on the z/OS NFS client 77
Accessing z/OS 83
Mvslogin command examples 84
“Permission denied” message 84
Mount command syntax and examples 85
Data conversion 86
BSAM, QSAM, and VSAM ESDS access to remote files 87
Invoking the Mount command on the z/OS platform 88
Additional mount command examples . . . 89
Getting authorization to access files 92
Saving of mount points 92
Automatic timed logout - logout attribute . . 92
Unmount command syntax and examples . . . 92
Disconnecting your mount point - unmount 92
Displaying client and server statistical information–nfsstat 94
Displaying server mount information–showmount 98
Displaying default and mount point attributes–showattr 99
Ending your z/OS session - mvslogout. . . . 101

Chapter 8. Initialization attributes for the z/OS NFS client 103
Client attribute syntax 103
Datacaching attribute. 110
Mount processing parameters and installation parameters 111
NFS client translation support 112
z/OS NFS client with z/OS NFS server 112

Chapter 9. Initialization attributes for the z/OS NFS server 115
Attributes used for z/OS UNIX file access. . . . 115
Multipliers 116
Duplicate attributes 116
Data set creation attributes syntax 116
Processing attributes syntax 121
Timeout attributes. 128
Retrieve attributes. 129
Mapped keyword processing attribute 129
Native ASCII processing attributes 130
Considerations for native ASCII environment support 130
NFS servers with non-z/OS based NFS clients 131
Site attributes syntax 134

Chapter 1. Introduction

This topic explains the Network File System's client-server relationship and introduces the IBM Network File System (z/OS NFS). When used to access z/OS UNIX System Services (z/OS UNIX) data, which conforms to portable operating system interface (POSIX) standards, it is similar to other UNIX/AIX Network File Systems.

Overview

A client is a computer or process that requests services on the network. A server is a computer or process that responds to a request for service from a client. A user accesses a service, which allows the use of data or other resources.

Figure 1 illustrates the client-server relationship. The upper right portion of the figure shows the z/OS NFS server. The lower right portion of the figure shows the z/OS NFS client. The left portion of the figure shows various NFS clients and servers which can interact with the z/OS NFS server and client. The center of the figure shows the Transmission Control Protocol/Internet Protocol (TCP/IP) network used to communicate between the clients and servers.

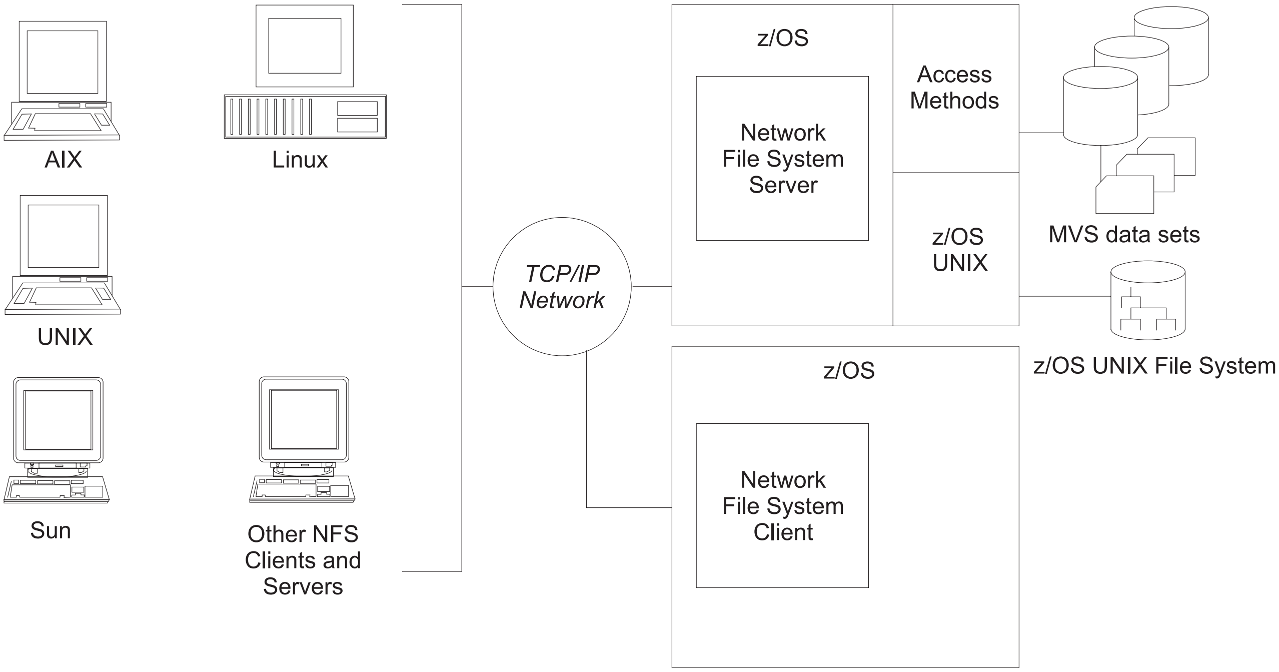


Figure 1. NFS client-server relationship

With the z/OS NFS server, you can remotely access MVS z/OS conventional data sets or z/OS UNIX files from workstations, personal computers, and other systems that run client software for the Sun NFS version 2, version 3, and version 4 protocols, and the WebNFS protocols over TCP/IP network.

The z/OS NFS server acts as an intermediary to read, write, create or delete z/OS UNIX files and multiple virtual storage (MVS™) data sets that are maintained on a z/OS host system. The remote MVS data sets or z/OS UNIX files are mounted

© Copyright IBM Corp. 1991, 2010

from the host processor to appear as local directories and files on the client system. This server makes the strengths of an z/OS host processor (storage management, high-performance disk storage, security, and centralized data) available to the client platforms.

With the z/OS NFS client, you can allow basic sequential access method (BSAM), queued sequential access method (QSAM), virtual storage access method (VSAM), and z/OS UNIX users and applications transparent access to data on systems that support the Sun NFS version 2 protocols, the Sun NFS version 3 protocols, and the Sun NFS version 4 protocols. The remote NFS Server can be a z/OS, UNIX, AIX®, or other system. The z/OS NFS client is implemented on z/OS UNIX and implements the client portion of the Sun NFS version 2 protocols, the Sun NFS version 3 protocols, and the Sun NFS version 4 protocols.

The NFS uses the communication services provided by TCP/IP, a suite of protocols that includes the remote procedure call (RPC) and External Data Representation (XDR) protocols. RPC allows a program on one machine to start a procedure on another machine, as if the procedure is local. XDR resolves the differences in data representation of different machines.

The NFS, then, can be used for file sharing between platforms and file serving (as a data repository).

If you use NFS as a file server, the z/OS UNIX file system might be a better choice than using z/OS conventional MVS data sets, because of its UNIX-based features.

z/OS UNIX files

The NFS server enables the client user remote access to z/OS UNIX files from a client workstation.

z/OS UNIX provides a hierarchical file system (HFS) for z/OS. The HFS file system is similar to a UNIX file system. All z/OS UNIX files reside in a directory, which in turn is a file in a higher level directory. The highest level directory is called the root directory.

When client users mount files from your server system, you can use a common HFS prefix to distinguish z/OS UNIX files from z/OS conventional MVS data sets or you can use an implied prefix. You see z/OS UNIX files in a standard UNIX format on your workstation, but the files are stored on a z/OS host system.

Using the NFS, the client can mount all or part of the z/OS UNIX file system and make it appear as part of your local file system. From there the client user can create, delete, read, write, and treat the host-located files as part of the workstation's own file system. For more information about z/OS UNIX see *z/OS UNIX System Services User's Guide*.

z/OS UNIX advantages

z/OS UNIX file system support provides these advantages over z/OS conventional MVS data sets:

- Support for hierarchical directories
- File names up to 255 characters in length
- Path names up to 1023 characters in length

- Mixed case names and special characters, except nulls and slash characters, in file and path names
- UNIX style access permission support
- Group and user ID support at a file level
- Ability to link z/OS conventional MVS data sets to a POSIX path name.

NFS protocol compliance

The z/OS Network File Systems provides full NFS protocol compliance for accessing the z/OS UNIX file system.

Crossing between file systems–NFS server

Crossing file systems means the NFS client can also potentially be a server, and remote and local mounted file systems can be freely mixed. This leads to some problems when a client travels down the directory tree of a remote file system and reaches the mount point on the server for another remote file system. Allowing the server to follow the second remote mount would require loop detection, server lookup, and user revalidation. When a client does a lookup on a directory on which the server has mounted a file system, the client sees the underlying directory instead of the mounted directory.

The NFS server does not support crossing file systems in NFS protocol versions 2 (NFSv2) and 3 (NFSv3) for either local or remote file systems. In NFS protocol version 4 (NFSv4) the z/OS NFS server does support crossing local file systems, but not remote file systems. For example, if a server has a file system called /usr and mounts another local file system on /usr/src, a client can also mount /usr, but the server will only see the mounted version of /usr/src with NFSv4. In NFSv2 and NFSv3, a client could perform remote mounts that match the server's mount points to maintain the server's view. In this example, the client would also have to mount /usr/src in addition to /usr, even if the mounts are from the same server.

z/OS conventional MVS data sets

Using NFS, you can access z/OS conventional MVS data sets from a client workstation, personal computer, or any client system using software for the NFS protocol.

In MVS, a file is called a data set. The NFS allows client users to mount z/OS conventional MVS data sets from their workstations. It presents the information to them in the form of a UNIX (or AIX) file, though the information is actually stored on an MVS-owned DASD.

The files for an operating system are organized into a file system. The UNIX environment use a file system that is a hierarchy of directories. z/OS conventional MVS, in contrast to z/OS UNIX, uses a non-hierarchical file system in which groups of data sets are referred to by specifying a high-level qualifier (HLQ).

The MVS HLQ can include the first (leftmost) qualifier of data sets, or the first and second qualifiers, or the first, second, and third qualifiers, and so on. For example, `SMITH` is the HLQ for the files named `SMITH.TEST.DATA` and `SMITH.PROJ7.SCHED`, while `SMITH.TEST` is the HLQ of `SMITH.TEST.DATA` and `SMITH.TEST.DOCS`.

Mounting MVS data sets onto a client mount point

To access an MVS file system from the client, client users use the **mount** command to create a temporary link (until unmounted) between specific MVS data sets and their UNIX directory (preferably empty) or an unused logical drive on their workstations. The empty UNIX directory or logical drive is called a mount point.

Client users use an MVS HLQ in the **mount** command to specify which MVS data sets to mount at a mount point. The MVS data sets beginning with the specified HLQ appear as files under the mount point.

Client users can can mount a fully-qualified PDS but not a fully-qualified PS dataset. Only cataloged data sets are supported by the z/OS NFS server. Tape data sets and generation data sets are not supported.

Some client platforms support both TCP and the user datagram protocol (UDP). Users can choose either TCP or UDP to access the server. The default protocol option depends on the NFS client platform. For NFS version 4, some platforms do not support UDP.

Note: When directly mounting on a fully qualified data set name, the server must return the mount size as part of getting the attributes for the mount. This can slow down the completion of the **mount** command.

Clients using the NFS version 4 protocol must pass mount requests to the server using a series of **lookup** operations. Client users may still use the mount command, and the client must convert the mount command into the lookup operations.

Creating z/OS conventional MVS data sets

Client users can create MVS data sets from a client system using the NFS. The default data set creation attributes specified by the system administrator are used to create MVS data sets, unless the user overrides them. These attributes determine how the MVS data sets are structured and where they are stored. Client users can override the default data set creation and processing attributes for a mount point when issuing the **mount** command. In addition, you can override these attributes at file creation time.

Serializing and sharing data sets

The z/OS NFS server handles data set serialization and sharing differently, depending on the type of data set:

Physical sequential
: The server ensures physical sequential data set read/write integrity by SVC 99 dynamic allocation with exclusive option whenever a physical sequential data set is opened for output. Otherwise, it allocates with share option.

Virtual storage access method (VSAM)
: The server dynamically allocates a VSAM data set with share option and allows the VSAM access method to manage data sharing using the **shareoptions** specified during data set definition.

Partitioned data set extended (PDSE)
: The server dynamically allocates a PDSE data set with share option and allows the PDSE functions to manage the serialization of the PDSE data set and its members.

Partitioned data set (PDS)
: For read and write, the z/OS NFS server issues ENQ SHR on QNAME=SYSDSN and RNAME=dataset_name (through an SVC 99). For write, the server issues an exclusive ENQ against QNAME=SPFEDIT and

RNAME=dataset_name.member_name, in addition to the serialization of resources by SVC 99. For all MVS users who are allocating their data set with exclusive status, this provides write protection. It only provides read integrity for ISPF users.

NFS version 2 and version 3 statelessness

Under the NFS version 2 and version 3 protocols, the z/OS NFS server strives to be as *stateless* as possible; that is, it tries to work correctly without maintaining any state information about any of its clients. However, a failure of the server causes cached writes to be lost, some attributes to be reset, and file handles to become stale, or not valid.

With the NFS version 4 protocol, the z/OS NFS server maintains state information for some client operations, to prevent such loses.

NFS version 4 state

The NFS version 4 protocol introduces *state information* that allows clients and servers to keep track of certain resources.

NFS version 4 uses a value of *clientid* or *stateid* to represent the current state (instance) of client-held resources such as locks, opens, and host restarts. The client and server pass this state information between them on certain operations, allowing both to agree on the current instance of resources held by the client.

NFS version 4 includes new states for the following:

1. Client/Server restart instance
2. Open Share/Deny instance
3. Byte Range Locks instance
4. Client Delegation instance.

The NFS version 4 state that is passed between the client and server represents a single instance in a dynamically changing environment; it is incremented when a state is changed within a group of held resources (restart, open, or lock). Once state is established on the server, the client returns what it believes is the current state. The server compares the client state to the server state to detect stale and out of order requests.

The client uses the **setclientid** operation to notify the server of its intention to use a particular client identifier for subsequent requests that entail creating lock, share reservation, and delegation state on the server. Upon successful completion the server returns a shorthand *clientid* , which, if confirmed in a separate step, will be used in subsequent file locking and file open requests. Confirmation of the *clientid* must be done using the **setclientid_confirm** operation to return the *clientid* and *setclientid_confirm* values, as verifiers, to the server.

NFS version 2 and version 3 used the Network Status Monitor (NSM) protocol to determine if resources such as file open share or byte range locks were still in use by a remote client. NFS version 4 no longer uses NSM to communicate a client or server restart. NFS version 4 instead uses a current state on both the client and server, where the state is established and passed in subsequent NFS version 4 operations.

In NFS versions 2 and 3, a client or server issues an NSM **sm_notify** RPC procedure to notify the remote host of a restart. Server resources such as an exclusive byte range lock on a file might remain held until explicitly released by the client. If a client that holds a server resource is removed from the network for a long period without the server being notified, the server resource would be unavailable to other clients until timed out by the server.

NFS version 4 provides a protocol for the client to establish or reestablish state, and associates ownership of subsequent server *stateful* operations to previously established states. To resolve the absent client problem, the NFS version 4 client must routinely refresh the state within the server-specified lease time. Upon lease time-out, the server may release resources for the client and make them available to other applications.

- A client obtains the server-specified lease time-out attribute by issuing a **getattr** operation. **getattr** is not a stateful operation, thus it does not require prior state to be established. A **getattr** operation may precede a **setclientid** or **setclientid_confirm** operation.
- Refer to the NFS server's **leasetime** site attribute for setting and tuning lease time periods.

Name space and file system management

NFS versions 2 and 3 used the mount protocol to define the initial "mount point" and its associated file handle. File locking was done with the Network Lock Manager protocol. NFS version 4 uses a well-defined port as an anchor. This allows the hookup (no longer called "mount") to occur implicitly, because the concept of a "Root" file handle, in combination with the port, allows the equivalent of a mount to take place on the server side.

In NFS versions 2 and 3, a client application would request to mount a file system object; the NFS client would then issue a "mount" protocol operation to the server, providing usage attributes, and specifying a file system object that is exported by the server. This mount command would specify to the server the name of the object to be mounted. The server would then provide a handle to the client for use in accessing objects related to this mount point.

In NFS version 4, this mount protocol is no longer used. Instead, the server provides a name space to the objects that are exported by the server. Standard non-mount operations such as LOOKUP and READDIR are changed by the NFS version 4 protocol to accomplish this. These changes are transparent to the client application. The NFS Client translates the mount request into the proper NFS version 4 operations that accomplish this access.

In NFS versions 2 and 3, objects in the server file system are accessed by a filehandle. This filehandle is given to the client in response to a mount or lookup operation, and is provided by the client when attempting to access objects in that file system. The NFS version 4 protocol specifies a pair of operations, PUTROOTFH and PUTPUBFH, that allow the client to request a starting point in the exported (or public, respectively) file system.

The NFS version 4 protocol also uses a COMPOUND procedure in which many operations can be sent in a single request. For this purpose, a filehandle is known within the COMPOUND structure as one of two items: a "current filehandle" and/or a "saved filehandle". NFS Version 4 operation PUTFH allows the client to provide a previously returned (by operation GETFH) filehandle, and operations

SAVEFH and RESTOREFH allow the client to manipulate the current and saved file handles within the compound procedure. Further operations within the COMPOUND RPC will make use of the handles, once established by these "filehandle manipulation" operations. Refer to the NFS version 4 protocol (RFC3530) for usage of the current and saved file handles.

For NFS version 4, when the client receives a mount command from an application, the client translates the command into a PUTROOTFH operation followed by a series of LOOKUP operations. If this series of LOOKUPs deviates from a path that would lead to an exported object, the LOOKUP that starts this deviation will be rejected with NFSERR_NOENT.

The elimination of the mount/unmount operations from the NFS version 4 protocol means that the NFS client can not tell the NFS server when an application unmounts a file system. As a result, the NFS server keeps the file system mounted until the mount point times out. Therefore, those unmounted file systems will still appear in the mount list produced by a **Modify mvsnfs,LISTMOUNT** operator command.

z/OS NFS File System Type Selection

The z/OS NFS Server must distinguish between the two different file system types when processing mount requests and other requests that involve the specification of path names: z/OS UNIX file systems and MVS data sets. Prior to z/OS NFS V1R11 this was done via the specification of the special "hfs" prefix for z/OS UNIX file systems and the absence of the prefix for MVS data sets.

As of NFS V1R11, the z/OS NFS file system type management function has been expanded by adding an mvs prefix and a customer configurable path resolution heuristic. The new mvs prefix provides the capability to explicitly specify a prefix for identifying MVS data sets, as the hfs prefix does for z/OS Unix files. The new customer-configurable heuristic enables you to specify how to interpret absolute path names that do not have a prefix specified. By setting the heuristic to indicate that MVS data sets now require a prefix and z/OS Unix files do not, symbolic links in z/OS Unix mount paths can be supported.

Specifying the path type prefix and the customer-configurable path resolution heuristic

The following z/OS NFS server site attributes apply to all NFS path names and their resolution processing. This includes directories specified in the Exports list and the checklist in the Exports file, as well as mount and root lookup objects.

HFSPREFIX(*prefix*)
specifies the z/OS UNIX file system prefix to be appended to the front of z/OS UNIX file system path names for a mount path directory. The default value of the prefix is /hfs

MVSPREFIX(*prefix*)
specifies an MVS data set prefix to be appended to the front of MVS data set name for a mount path directory. The default value of the prefix is /mvs.

IMPPREFIX(*impprefix*)
specifies how to interpret a mount path that does not have a path type prefix, where *impprefix* is one of the following:

NONE
An explicit prefix must always be specified for an absolute path. Implicit prefix resolution is not valid in this case.

HFS If no explicit prefix is present, assume the path is a z/OS UNIX file system.

MVS If no explicit prefix is present, assume the path is an MVS data set. This is the default.

HFS, MVS
If no explicit prefix is present, first assume the path is a z/OS UNIX file system. If no matching z/OS UNIX file system can be found, assume that it is an MVS data set.

MVS, HFS
If no explicit prefix is present, first assume the path is an MVS data set. If no data set with a matching high-evel qualifier can be found, assume that it is a z/OS UNIX file system.

Valid mount path specifications

This section shows valid mount path specifications for back-level (prior to V1R11), MVS prefix, and the MVS implicit prefix.

Back-level (prior to V1R11):

Note: An ellipsis (...) denotes that more similarly-specified items are possible.

- v2/v3,

```
/hfs/[,procattr]   '/' after the prefix is required
/hfs/abc[/def...][,procattr]
mvsdsn[(mbr)][,procattr]
```

- v4

```
/hfs[/][,procattr]
/hfs/abc[/def...][,procattr]
mvsdsn[(mbr)][,procattr]
```

MVS prefix:

- v2/v3

```
/mvs/mvsdsn[(mbr)][,procattr]
```

- v4

```
/mvs/mvsdsn[(mbr)][,procattr]
```

MVS implicit prefix: Table 2 on page 11 summarizes the interpretation of the mount statements based on the IMPPREFIX setting.

Table 2. Mount statements based on the IMPPREFIX setting

Mount\IMPPREFIX	MVS	HFS	MVS,HFS	HFS,MVS
v2/v3				
/abc[,procattr]	MVS HLQ	z/OS UNIX dir	1. MVS HLQ if a data set exists 2. z/OS UNIX dir otherwise	1. z/OS UNIX dir if exists 2. MVS HLQ otherwise
/abc[/def...][,procattr]	Invalid	z/OS UNIX dir	1. MVS Invalid, so 2. z/OS UNIX dir	1. z/OS UNIX dir if exists, 2. Invalid otherwise
/a[.b...][(mbr)][,procattr]	MVS PDS with member 'mbr' if PDS exists; otherwise invalid	z/OS UNIX dir	1. MVS PDS with member 'mbr' if PDS exists 2. z/OS UNIX dir otherwise	1. z/OS UNIX dir if exists, 2. MVS PDS with member 'mbr' otherwise
v4				
/abc[,procattr]	MVS HLQ	z/OS UNIX dir	1. MVS HLQ if a data set exists 2. z/OS UNIX dir otherwise	1. z/OS UNIX dir if exists 2. MVS HLQ otherwise
/abc/def[,procattr]	MVS PDS with member 'def' if PDS exists; otherwise invalid	z/OS UNIX dir	1. MVS PDS with member 'def' if PDS exists; otherwise invalid, so 2. z/OS UNIX dir	1. z/OS UNIX dir if "/abc" exists, 2. MVS PDS with member 'def' if PDS exists; otherwise invalid
/abc/def/...[,procattr]	Invalid	z/OS UNIX dir	1. MVS Invalid, so 2. z/OS UNIX dir	1. z/OS UNIX dir if "/abc" exists, 2. Invalid otherwise
/a[.b...][(mbr)][,procattr]	MVS PDS with member 'mbr' if PDS exists; otherwise invalid	z/OS UNIX dir	1. MVS PDS with member 'mbr' if PDS exists; otherwise invalid, so 2. z/OS UNIX dir otherwise	1. z/OS UNIX dir if exists, 2. MVS HLQ otherwise

Implicit prefix support restrictions

The following processing characteristics and restrictions must be considered when using the path name prefix processing support provided by the NFS Server Site Attributes:

1. When both path options are available based on the IMPPREFIX site attribute (when IMPPREFIX = mvs,hfs or hfs,mvs), only the existence or nonexistence of the first path name qualifier is used to determine whether the second option is tried. That is, if the first path qualifier exists, but the next one does not, the object is considered not to exist and the mount/lookup fails.

2. Prior to V1R11, an MVS mount to an HLQ (for example, a.b.c) for which no datasets exist was considered valid and would mount to that HLQ, allowing the first dataset to be created via NFS. As of V1R11, if the IMPPREFIX site attribute specifies mvs,hfs, NFS Version 4 mounts to such an HLQ fail on the MVS side and then attempt to mount z/OS UNIX node /a.b.c. If that z/OS UNIX node does not exist, the mount fails. If this behavior is not desired, either an MVS prefix must be specified on the path or one of the other IMPPREFIX site attribute values must be specified.
3. For IMPPREFIX(HFS,MVS), if the object does not exist (neither z/OS UNIX nor MVS), it creates the mount point as a new MVS HLQ with no entries, just as an MVS mount does in prior releases.

   Conversely, for IMPPREFIX(MVS, HFS), if the object does not exist (neither z/OS UNIX nor MVS), it tries MVS, then z/OS UNIX, then fail, just as a z/OS UNIX mount for a non-existent object did prior to V1R11.

   Once NFS has switched to option 2, it cannot switch back to option 1.
4. For NFS Version 2 or Version 3 mount requests, NFS clients send the entire mount path to the NFS server as a single string. By contrast, for NFS Version 4 mount requests, NFS clients send a series of lookup requests (there is no mount request) to the NFS server for one path qualifier at a time. Consequently, the NFS server does not know whether additional path qualifiers follow. This can produce unexpected results.
5. If a mount request is issued as an NFS Version 2 or Version 3 mount request, the path is handled as a single string entity and is resolved by z/OS UNIX resolving the symbolic link to the z/OS UNIX /a directory, ignoring the IMPPREFIX setting. This is effectively no change from prior releases.

   The same is true for NFS Version 4 mount requests if the symbolic link is the last name in the path and it is followed by some processing attributes.

   Note: If the NFS client is z/OS there will always be at least one processing attribute, automatically added by the client identifying it as z/OS.

   However, if the mount request is issued as an NFS Version 4 mount request and the symbolic link is not the last name in the path, or it is not followed by any processing attributes, then the symbolic link will be identified as such back to the NFS client. The client will then read the link data and reinitiate the path resolution. In this case, assuming the link is defined as an absolute path, then the path type resolution will come into play based on whether a prefix is included and based on the implicit prefix resolution heuristic.

   Note: This can cause the symbolic link to resolve into MVS space, not just z/OS UNIX space.
6. The implicit prefix heuristic also applies to the exports file; that is, for export entries that do not include an explicit prefix, the IMPPREFIX( ) site attribute is used to determine the specified path. If both the HFS and the MVS options are specified, the export entry applies to both types of file systems, assuming that the specified entry exists in both file system spaces.
7. When the NFS Server restarts, it attempts to recover mount points recorded in the MHDB. If the HFS or MVS prefix and/or implicit prefix site attributes were changed before the restart, the new mount points will reflect the new HFS and MVS prefixes. Implicit prefix changes will have no effect.

   NFS Version 4 mount points that were established without specifying the MVSMNT processing attribute were not recorded in the MHDB. The NFS Client will attempt to re-establish these mount points when it receives a stale file handle response from the NFS Server. However, the NFS Client has no knowledge of the changed prefix site attributes and will use the original mount name string in this attempt. This can result in the NFS Client not being able to reestablish the mount points.

Note: This statement only applies when the NFS server prefix site attributes are changed during the server restart. Otherwise, the NFS client should be able to re-establish the mount points.

8. If the IMPPREFIX(NONE) Site Attribute is specified, **all** path names, including those in the exports file (if used), must be specified with a prefix.

Server control files

These special files are used by the z/OS system administrator to control the z/OS NFS server:

Attributes data set

Exports data set

Mount handle data sets

Log data sets

For information about customizing these control files, see "Configuring the z/OS NFS server" on page 171.

Attributes data set

The attributes data set contains the settings for the z/OS NFS server. There are three types of attributes stored in this data set:

Data set creation attributes
Used to define the structure of MVS data sets when creating a file (for z/OS conventional MVS data sets only).

File processing attributes
Used to control how files are accessed by the client.

Site attributes
Used to control z/OS NFS server resources.

The system administrator changes the default settings by editing the attributes data set and restarting the server. Client users can override the data set creation and file processing attributes at the command line. For z/OS conventional MVS data sets, the client user can specify the data set creation attributes when mounting, creating, or accessing files. The client user can override the file processing attributes when mounting, creating, or accessing files. However, some file processing attributes can only be overridden on a mount point basis.

Note: Many of the attributes are valid only for z/OS conventional MVS files, and not for z/OS UNIX files. "Attributes used for z/OS UNIX file access" on page 115 gives a complete list of attributes that are valid for z/OS UNIX.

Exports data set

The exports data set can control which client users can mount which MVS data sets. The entries in the exports data set specify which MVS high-level qualifiers or HFS directories can be mounted. The system administrator can use this data set to limit mounts to accredited clients only. It also controls which client users can mount all or part of the z/OS UNIX file system, based on the client machine's specified Internet Protocol (IP) address. To use the exports data set, the **security** site attribute must be set to either **safexp** or **exports** by the MVS system administrator.

In z/OS V1R8, the exports data set also provides the function previously provided by the checklist data set: specifying files or directories that are exempt from System Authorization Facility (SAF) checking even though **saf** or **safexp** is specified as the security option.

Mount handle data set

The z/OS NFS server maintains a list of the active mount points in a pair of files called the mount handle data sets on MVS. The two data sets are used alternately to automatically reestablish the client mount points when the server is started. If the file system is not available, the mount point is not reestablished and the mount failure is recorded in the log data set.

The z/OS NFS server does the cleanup activity during z/OS NFS server shutdown and daily at the cleanup time specified by the **restimeout** site attribute.

During cleanup time, the z/OS NFS server reads the list and checks all mount points against the retention period specified in the restimeout site attribute. If your mount points are idle longer than the retention period specified in the **restimeout** site attribute, they are removed. Only the active mount points are reconnected.

If a mount handle is removed by the cleanup activity, the client user might receive the “Stale NFS File Handle” message or some other appropriate message. If so, all the client user needs to do is unmount the stale mount point and mount it again.

Log data set

The log data sets store the messages for the z/OS NFS start-up procedures. This log can be used to identify the user's correctable errors or the user's problem errors. There are two logs that this information is stored in; the primary log and the secondary log. The primary log is used at start-up until it is filled and then overflows into the secondary log. When the secondary log is full, the primary log will then be overwritten with new error messages. The number of log records is dependent on the number of transactions that the server can handle.

The z/OS NFS server also records messages and diagnostic information in a z/OS component trace buffer, if one is specified. Component trace buffers can be used in addition to or instead of the log data sets. Using a component trace buffer can provide performance improvements over the log data sets. For details, see “Using z/OS component tracing” on page 297.

Tested clients for the z/OS NFS server

Tested clients for the z/OS NFS server, using the NFS version 2, version 3, and version 4 protocols, are:

- z/OS NFS Client
- Linux
  - RedHat EL 5
    - Intel x86
    - PPC64
    - z/Linux

- AIX
  - 5.3
  - 6.1
- Sun Solaris version 10
  - Sparc
  - X86_64
- Windows/Hummingbird
  - Windows XP
    - Hummingbird Maestro™ 2008
  - Vista Business
    - Hummingbird Maestro™ 2008
  - Windows 2003 Server
    - Hummingbird Maestro™ 2007
    - Hummingbird Maestro™ 2008

  Notes:

  1. Maestro 2007 and 2008 clients have a restriction that prevents you from specifying server attributes on the mount command. Hummingbird is aware of the issue and is working to resolve the problem.
  2. IBM has performed limited testing with Hummingbird Maestro 10, 2006 and 2007.
  3. Older versions of these clients are still supported under the NFS version 2, version 3, and version 4 protocols, but not all have been tested by IBM.
- Other client platforms should work as well, since NFS is an industry standard protocol, but have not been tested by IBM.

NFS protocol attributes for the z/OS NFS server

The NFS protocol defines file attributes that NFS clients can read and set on NFS servers. In the NFS version 4 protocol, some file attributes are mandatory and others are recommended for servers to support. For a list of the NFS version 4 file attributes that the z/OS NFS server supports, see Appendix C, "NFS server attributes," on page 457.

z/OS NFS server restrictions

As of the writing of this publication, use of the following functions must be restricted because they have not been successfully tested on the listed systems:

- AIX 5.3
  - Cannot mount NFSv4 file systems that include Symbolic Links
  - AIX system will crash after the AIX NFSv4 client attempts to mount a symlink without Fix IZ28515 for AIX5.3
- AIX 6.1
  - NFSv4 mounts to MVS datasets must include a leading slash before the HLQ.
- Linux
  - Redhat EL 5.2
    - Cannot mount NFSv4 shares that include Symbolic Links
  - Suse SLES 10.2
    - Cannot mount NFSv4 shares that include Symbolic Links
    - KRB5p is not supported

- Windows
  - Double quotes have to be used when Maestro client 2008 (minimal maestro patch 15) mounts to the z/OS NFS server if the remote mount path doesn't exactly match the NFS exports.

    Example 1: Export USS /u/user* or MVS A.B in export file.

    Maestro GUI:

    ```
     "\\server\/hfs/u/user4"
     "\\server\/A.B.CD"
    ```

    DOS/Shell:

    ```
    nfs link mountdrive \"\\server\/hfs/u/user4\" /T /4 /A:u
    nfs link  mountdrive \"\\server\/A.B.CD\" /T /4 /A:u
    ```

    Example 2: Use double quotes when mounting to the z/OS server with z/OS server attributes.

    Maestro GUI:

    ```
    "\\server\/nfsexport,binary"
    ```

    DOS/Shell:

    ```
    nfs link  mountdrive  \"\\server\/nfsexport,binary\"  /T /4 /A:u
    ```

Note: This is not intended to be an all-inclusive list of remote platform restrictions.

Tested servers for the z/OS NFS client

The z/OS NFS client supports servers that implement the server portion of the Sun NFS Version 2, Version 3, and Version 4 protocols:

- z/OS NFS Server
- Linux
  - RedHat EL 5
    - Intel x86
    - PPC64
    - z/Linux
- AIX
  - 5.3
  - 6.1
- Sun Solaris version 10
  - Sparc
  - X86_64
- Windows/Hummingbird
  - Windows XP
    - Hummingbird Maestro™ 2008
  - Vista Business
    - Hummingbird Maestro™ 2008
  - Windows 2003 Server
    - Hummingbird Maestro™ 2007
    - Hummingbird Maestro™ 2008

Notes:

1. IBM has performed limited testing with Hummingbird Maestro 10, 2006 and 2007.

2. Older versions of these servers are still supported under the NFS version 2, version 3, and version 4 protocols, but not all have been tested by IBM.

- Other server platforms should work as well, since NFS is an industry standard protocol, but have not been tested by IBM.

A **mount** parameter **vers(x)**, where *x* is either 2, 3, or 4 is provided to make the z/OS NFS client communicate with the server at the specified protocol level. The z/OS NFS client also communicates at the highest protocol level that is supported by the server if no level is specified.

- If **no version** is specified and if the server supports:
  - Only the NFS version 2 protocol, then the z/OS NFS client will use the NFS version 2 protocol to communicate
  - The NFS version 2 and 3 protocols, then the z/OS NFS client will use the NFS version 3 protocol to communicate
  - The NFS version 2, 3 and 4 protocols, then the z/OS NFS client will use the NFS version 4 protocol to communicate.
- If **vers(2)** is specified, then use NFS version 2 protocol to communicate with the server.
- If **vers(3)** is specified, then use NFS version 3 protocol to communicate with the server. z/OS NFS client fails the mount command if the server does not support NFS version 3 protocol.
- If **vers(4)** is specified, then use NFS version 4 protocol to communicate with the server. z/OS NFS client fails the mount command if the server does not support NFS version 4 protocol.

z/OS NFS client restrictions

As of the writing of this publication, use of the following functions must be restricted because they have not been successfully tested on the listed systems:

- Sun

  Sun NFS Server may fail to start if a Solaris 10 system was brought down with outstanding locks on a NFSv4 share. Patch 13463-01 for SunOS 5.10_x86 and Patch 139462-01 for SunOS 5.10 address this issue.

Note: This is not intended to be an all-inclusive list of remote platform restrictions.

WebNFS support

The z/OS NFS server supports the WebNFS protocol. WebNFS specification extends the semantics of NFS versions 2, 3 and 4 protocols to allow clients to obtain file handles without the mount protocols. The z/OS NFS server supports the public filehandle and multi-component lookup features as well as other additional requirements as described in RFC 2055. A keyword, **public**, is added for the system administrator to specify the public paths that the public file handle can access. A public path for z/OS conventional MVS data and a public path for HFS data can both be specified. When a lookup request comes in from an NFS client and an absolute path name is specified, it will be matched with the public paths to determine which public path it is trying to reference. If a relative path is specified and both HFS and MVS public paths are defined then the lookup request will be processed relative to the HFS public path.

The following are restrictions for the WebNFS support provided by the z/OS NFS server in this release.

Export Spanning Pathnames

Lookup requests that reference files or directories outside of the exported public path, will result in an error condition.

Native Path

Only canonical pathnames will be supported.

Canonical path

A canonical path is a hierarchically-related, slash-separated sequence of components, in the form: <directory>/<directory>/../<name>.

Processing attributes

NFS version 2, version 3, and version 4 public MVS mount does not support attributes inheritance. Therefore server attributes specified with the mount command (such as `text`, `srv_ccsid()`, and `cln_ccsid()`) cannot be inherited when subsequently working with MVS data under a mount point. The z/OS NFS server processes MVS data as binary.

For example, to work with a PDS member a.b.c(text_memb) in text processing mode it is necessary to do a public mount directly to the member:

```
mount -o vers=x,public zNFSserver:"/a.b.c/text_memb,text" client_mount_path
```

In this case, the text attribute specified on the mount command will be taken into account when processing the data.

NFS version 3 and version 4 with TCP/IP protocols

Information for NFS version 3 and version 4 protocols with **proto=tcp** can be found on the mount man page on a UNIX client. The NFS client automatically selects the **proto=tcp** option, unless the end-user overrides the option. For example, you can enter this command:

```
unix$ mount -o vers=2,proto=udp mvshost1:smith /mnt
```

This example shows a specification of NFS version 2 with **udp** protocol, even though the client platform can handle the NFS version 4 and **tcp** protocol.

Users can issue the `rpcinfo -p <hostname>` to show all the RPC programs available on the server. Table 3 shows an example.

```
$ rpcinfo -p mvshost1
```

Table 3. View of NFS server capability

program	vers	proto	port	service
100000	4	tcp	111	portmapper
100000	3	tcp	111	portmapper
100000	2	tcp	111	portmapper
100000	4	udp	111	portmapper
100000	3	udp	111	portmapper
100000	2	udp	111	portmapper
150001	1	udp	4954	pcnfsd
150001	2	udp	4954	pcnfsd
100024	1	udp	4955	status
100024	1	tcp	4944	status
100021	1	udp	4956	nlockmgr
100021	1	tcp	4945	nlockmgr
100021	3	tcp	4945	nlockmgr
100021	3	udp	4956	nlockmgr
100021	4	tcp	4945	nlockmgr

Table 3. View of NFS server capability (continued)

program	vers	proto	port	service
100021	4	udp	4956	nlockmgr
100003	2	tcp	2049	nfs
100003	2	udp	2049	nfs
100003	3	tcp	2049	nfs
100003	3	udp	2049	nfs
100003	4	tcp	2049	nfs
100059	2	udp	4953	
100059	2	tcp	4943	
100044	1	udp	4952	
100044	1	tcp	4942	
100005	1	udp	4951	mountd
100005	1	tcp	4941	mountd
100005	3	tcp	4941	mountd
100005	3	udp	4951	mountd

Users can issue `rpcinfo -s <hostname>` from Solaris clients to show a concise list of all the RPC programs available on the server.

Here is an example of output from `rpcinfo -s <hostname>` in an IPv4 environment:

```
program version(s) netid(s)                         service     owner
 100000  4,3,2     udp,tcp                          rpcbind     superuser
 150001  2,1       udp                              pcnfsd      unknown
 100024  1         tcp,udp                          status      unknown
 100021  4,3,1     tcp,udp                          nlockmgr    unknown
 100003  4,3,2     udp,tcp                          nfs         unknown
 100059  2         tcp,udp                          showattr    unknown
 100044  1         tcp,udp                          mvsmount    unknown
 100005  3,1       tcp,udp                          mountd      unknown
```

Here is an example of output from `rpcinfo -s <hostname>` in an IPv4/IPv6 environment:

```
program version(s) netid(s)                         service     owner
 100000  4,3,2     udp,udp6,tcp,tcp6                rpcbind     superuser
 150001  2,1       udp6,udp                         pcnfsd      unknown
 100024  1         tcp6,tcp,udp6,udp                status      unknown
 100021  4,3,1     tcp6,tcp,udp,udp6                nlockmgr    unknown
 100003  4,3,2     udp6,udp,tcp6,tcp                nfs         unknown
 100059  2         tcp6,tcp,udp6,udp                showattr    unknown
 100044  1         tcp6,tcp,udp6,udp                mvsmount    unknown
 100005  3,1       tcp6,tcp,udp6,udp                mountd      unknown
```

Internet Protocol version 6

Internet Protocol version 6 (IPv6) expands the range of addresses that are available for internet communications. IPv6 extends address sizes from a 32-bit value to a 128-bit value, vastly expanding the number of globally unique addresses that can be assigned. Both the z/OS NFS Client and the z/OS NFS Server support the longer addresses of IPv6, as well as the 32-bit addresses of IPv4 and below. Your network infrastructure must be enabled to use IPv6; if the network does not support IPv6, z/OS NFS will use IPv4 instead.

The z/OS NFS server can use both IPv6 and IPv4 for all NFS protocols.

SMF records for the z/OS NFS server report client IP addresses for both IPv4 and IPv6, with separate address fields for each.

The z/OS Portmapper does not support IPv6. Therefore, when using IPv6 addresses, the z/OS server host must be configured with RPCBIND, not the Portmapper. RPCBIND supports both IPv6 and IPv4. As of z/OS V1R8, Portmapper should only be used for IPv4 only systems. Otherwise, RPCBIND should be used.

User-specified port range support

The z/OS NFS server supports a user-specified range of ports. The z/OS NFS server allows users to specify port assignments for services **mountd**, **mvsmount**, **pcnfsd**, and **showattr**. Additional ports are also required by the server for locking functions. The port assignments for these services can be any port number (except for reserved port 2049 for the NFS program) but must be a contiguous port range for the z/OS NFS server to identify them. The user specified range of ports provides a flexible port range to accommodate programs such as a firewall that supports a range of ports for security purposes.

Users wanting a user-specified port range setup must change the /etc/rpc file for the z/OS NFS client and the /etc/services and tcpip.profile files for the z/OS NFS server. For more information, see “Setting up a user-specified port range” on page 186.

Dynamic addressing

Before z/OS V1R7, the z/OS NFS client and server were based on the static IP address model to handle all communications with other systems. However, many systems have migrated from the use of static IP addresses to the dynamic host configuration protocol (DHCP). Now, the z/OS NFS server accepts dynamic NFS client IP address changes and properly understands the source of the communication even if the sender's IP address has changed. Since not all customers' environments use dynamic IP addresses, NFS server site attributes have been added to specify whether the NFS server should use the dynamic IP algorithm (**dhcp**) or the current static IP algorithm (**nodhcp**). The default is **nodhcp**, to use the static IP algorithm.

To use dynamic IP addressing, the client must:

- Have a constant host name that the NFS server can identify it by.
- Dynamically update the authentication DNS (dynamic name server) with new IP addresses whenever they change.
- Maintain the TTL (time to live) value that the authentication DNS server specifies to any caching DNS server, based on the frequency with which system IP addresses might change.

If you are using the static IP algorithm (nodhcp) and there are changes to network addressing, the exportfs command will not rebuild the exports list correctly. See “TCP/IP” on page 316 for further details.

For more information, see “Using dynamic client IP addressing” on page 191.

The z/OS NFS Server continues to have a static IP address, based on the standard industry practice of assigning static IP addresses to servers.

Data transfer and conversion

With the NFS version 4 protocol, text data and metadata are transferred between the server and client in the UTF-8 data format (ASCII text is not transferred directly). z/OS NFS conversion of UTF-8 text data and metadata requires setting up a conversion environment using the z/OS Unicode Services by creating a Unicode conversion image that defines conversion tables with UTF-8 [CCSID 1208].

With the NFS version 4 protocol, *stringprep* provides preparation of internationalized strings. Stringprep helps ensure that character string input and string comparisons work consistently and correctly for users of multilingual text. The z/OS NFS server supports the UTF-8 encoding and stringprep requirements in the NFS Version 4 protocol, using z/OS Unicode services to normalize inbound UTF-8 encoded strings when comparisons are needed.

The server site attributes **stringprep** and **nostringprep** let you enable or disable stringprep normalization. You can use this attribute to disable stringprep normalization if necessary, for example if needed for compatibility with existing client workaround utilities. See “Site attributes syntax” on page 134 for information on the **stringprep** attribute.

Native ASCII support

The z/OS NFS client and server support applications running on z/OS V1R2 (and higher) in a native ASCII environment. Applications can operate on files in either EBCDIC or ASCII format, as well as other data formats defined with a coded character set identifier (CCSID). Native ASCII support is provided with a mechanism called file tagging where the file is defined with a tag to identify the CCSID to use for data conversion. File tagging is defined in the appropriate z/OS UNIX System Services documents. The z/OS NFS client and server provide the necessary support to provide data conversion between different CCSIDs specified for the client and server. The z/OS NFS client **cln_ccsid** and **srv_ccsid** parameters are also supported by the z/OS NFS server to identify the CCSID to be used in the data conversion. See “Processing attributes syntax” on page 121 for more information about the **cln_ccsid** and **srv_ccsid** parameters.

Chapter 2. Creating z/OS conventional MVS data sets

This topic explains how to create the various types of data sets (files) that are supported by the z/OS NFS server.

The examples shown are from an AIX client platform perspective. Any examples for other platforms are so indicated.

Overriding data set creation attributes

When you create an MVS file, default file creation attributes are applied, unless you override them. The attributes are passed to the z/OS host.

Data set creation attributes are controlled in the following ways, in increasing order of priority.

- Default server data set creation attributes
- Default installation data set creation attributes, specified by the system administrator in the attributes data set
- DFSMS data class attributes
- Data set creation attributes specified in the **mount** (or **nfs link**) command
- Data set creation attributes specified in the **mkdir**, **vi** (edit), or **cp** (copy) commands (highest priority)

The z/OS NFS server does not support the following data class attributes.

- CI size of data component
- Number of volumes
- Percentage of CI or CA free space
- Retention period/Expiration date
- VSAM imbed index option
- VSAM replicate index option

Preparing to create an MVS file

When creating an MVS file, you should know whether to process the file in text or binary mode (see “Selection of text or binary processing modes–text, binary” on page 34) and what type of file to create.

The z/OS NFS server supports the following types of files.

- Physical sequential (PS) data sets, including basic format and extended format data sets but excluding compressed format data sets.
- Direct access (DA)
- Partitioned data sets (PDS)
- Partitioned data sets extended (PDSE)
- VSAM KSDS
- VSAM ESDS
- VSAM RRDS

Keyed access to files and GDG are not supported.

© Copyright IBM Corp. 1991, 2010

Naming MVS files

The z/OS NFS server uses the comma (,) as a delimiter to a list of file attributes. Do not use a comma as a special character in file name. For example, you can enter this command:

```
$ vi "/u/smith/new,text"
```

This indicates to NFS that a file called *new* is being edited in the attribute text mode, not file `new,text`.

When naming z/OS conventional MVS files, you must follow the MVS file naming conventions, as described in *z/OS DFSMS Using Data Sets*rig.

For information about the z/OS UNIX System Services naming conventions, see Chapter 4, "Using z/OS UNIX System Services files," on page 47.

An MVS file name (or data set name) can consist of one or several simple names joined so that each represents a level of qualification. For example, the MVS file name `DEPT58.SMITH.DATA3` is composed of three qualifiers.

The following characteristics apply to the MVS file name.

- Each qualifier consists of 1 to 8 alphanumeric characters, national characters (@, #, $), or a hyphen (-)
- Each qualifier must start with an alphabetical or national character
- The period (.) separates simple names from each other
- Including all simple names and periods, the length of the MVS file name must not exceed 44 characters
- PDS and PDSE member names can be up to 8 characters long

For information about the MVS file system, see "Mapping between the workstation and MVS file systems" on page 40.

Restrictions on using alias names for MVS files

For non-VSAM files, alias names can be used interchangeably with the true file name except on remove (**rm** and **rmdir**) and rename (**mv**) requests. Renaming or removing an alias name results in an I/O error. This is due to an MVS restriction.

If the true name of an MVS file is renamed or removed and alias names have been defined for the file, MVS deletes the alias names during execution of the rename or remove request.

Creating physical sequential files

When creating a physical sequential (PS) file, specify the **dsorg(ps)** attribute (if it is not the default already) with the **mount** command or a file creation command, such as the **vi** UNIX (or AIX) command.

1. Create a local directory on your client to be used as a mount point. For example (with UNIX), enter this command:

   ```
   $ mkdir /u/smith/mnt
   ```

2. Mount the MVS file system. For example, suppose your host is `mvshost1`, and you want to issue a mount on the high-level qualifier `smith`. You can enter this command:

   ```
   # mount mvshost1:"smith,dsorg(ps)"  /u/smith/mnt
   ```

If you do not specify any other attributes, the MVS site defaults are used. You can use the **showattr** command to display the site defaults.

3. You can use the **vi** UNIX command to create the `new` file.

   ```
   $ vi /u/smith/mnt/new
   ```

   When you save the file using **vi**, you have just created a new MVS PS file named `SMITH.NEW`.

You can get the same results by specifying **dsorg**(*ps*) in the file creation command rather than in the **mount** command.

```
# mount mvshost1:smith /u/smith/mnt
$ vi "/u/smith/mnt/new,dsorg(ps)"
```

Creating direct access files

When creating a direct access (DA) file, specify the **dsorg(da)** attribute (if it is not the default already) with the **mount** command or a file creation command (such as the **vi** UNIX command).

1. Create a local directory on your client to be used as a mount point. For example (with UNIX), enter this command:

   ```
   $ mkdir /u/smith/mnt
   ```

2. Mount the MVS file system. For example, suppose your host is `mvshost1`, and you want to issue a mount on the high-level qualifier `smith`. You can enter this command:

   ```
   # mount mvshost1:"/mvs/smith,dsorg(da)" /u/smith/mnt
   ```

   If you do not specify any other attributes, the multiple virtual system (MVS) site defaults are used. You can use the **showattr** command to display the site defaults.

3. Next, you can use the **vi** UNIX command to actually create the `new` file.

   ```
   $ vi /u/smith/mnt/new
   ```

   You have just created a new MVS DA file named `SMITH.NEW`.

You can get the same results by specifying **dsorg(da)** in the file creation command, rather than in the **mount** command.

```
# mount mvshost1:smith /u/smith/mnt
$ vi "/u/smith/mnt/new,dsorg(da)"
```

Creating PDSs and PDSEs

Partitioned data sets (PDS) and partitioned data sets extended (PDSE) can be used as directories, and their members are files within those directories. An illustration of the use of PDSs to act as directories is shown in Figure 2 on page 43. For general information on PDSs and PDSEs, see *z/OS DFSMS Using Data Sets*.

You cannot create new directories within a PDS or PDSE, due to the nature of these data structures.

Creating a PDS or PDSE - mkdir dsntype(pds), dsntype(library)

To create a PDS or PDSE, perform the following steps:

1. Create a local directory on your client to be used as a mount point. For example (with UNIX), enter this command:

   ```
   $ mkdir /u/smith/mnt
   ```

2. Mount the MVS file system (accessing files that begin with the high-level qualifier of smith).

   ```
   # mount mvshost1:"smith,mgmtclas(normal)"
   /u/smith/mnt
   ```

3. If creating a PDSE, use the **mkdir** (make directory) UNIX command, specifying the **dsntype(library)** attribute to create a PDSE named smith.datalib.

   ```
   $ mkdir /u/smith/mnt/"datalib,dsntype(library)"
   ```

   If creating a PDS, use the mkdir (make directory) UNIX command, specifying the following dsntype(pds) attribute.

   ```
   $ mkdir /u/smith/mnt/"datalib,dsntype(pds),dir(20)"
   ```

   Omitting dsntype(pds): You can omit specifying the **dsntype(pds)** attribute if **pds** has been specified for the **dsntype** attribute either in your site attribute data set or in your mount point.

4. You can use the **vi** UNIX command to create a PDS or PDSE member named smith.datalib(member1).

   ```
   $ vi "/u/smith/mnt/datalib/member1,text"
   ```

   Type your text, save it, and quit.

You have now created a PDS or PDSE member, which is processed in text processing mode. You can use the **cat** UNIX command to view the contents of your PDS or PDSE member.

Note: z/OS NFS server supports a maximum of 14,562 members in a PDS or PDSE data set. When a NFS read-directory request on a PDS or PDSE is processed, the z/OS NFS server will return up to 14,562 member names. Other requests, such as read and write, to individual members are not affected.

Removing a PDS or PDSE - rm, rmdir

To remove a PDS or PDSE, first make sure that the PDS or PDSE is empty. You can delete all members under the directory using the **rm** UNIX command. Then use the **rmdir** (remove directory) UNIX command. This example removes the datalib directory, and confirms its removal by a failed try to query it (**ls** is the UNIX list files command).

```
$ ls -F /u/smith/mnt/datalib
data1* data2* data3*
$ rm /u/smith/mnt/datalib/*
$ rmdir /u/smith/mnt/datalib
$ ls -F /u/smith/mnt/datalib
/u/smith/mnt/datalib not found
```

Accessing PDS or PDSE members

There is more than one way to mount and access PDS and PDSE members. For example, you can display the existing PDS member smith.source(bigblue) by entering either of these command sequences.

```
$ mkdir /mnt
# mount hostname:"smith.source,text" /mnt
$ cat /mnt/bigblue
```

Or

```
$ mkdir /mnt
# mount hostname:"smith,text" /mnt
$ cat /mnt/source/bigblue
```

These two approaches are equivalent.

Updating or extending a PDS or PDSE member

The z/OS NFS server does not support updating or extending a PDS or PDSE member directly. To update or extend a PDS or PDSE member, a client program must follow these steps.

Step 1. Copy the file to the client machine

Step 2. Update or extend the copied version on the local system

Step 3. Truncate the original MVS file to zero size by sending a `SETATTR` request with zero file size

Step 4. Copy the updated version on the local host to MVS by writing request

Some client editors follow these steps, for example, the AIX and UNIX **vi** editor. Other editors do not follow these steps, for example, the z/OS UNIX OEDIT editor. In the latter case the user must save the updated version into a new file.

Timing out while writing a PDS or PDSE member

If you are writing to a PDS or PDSE member and a timeout occurs, the timeout causes the member to close. The remaining write requests appear to append to a PDS or PDSE member. The write request does not complete successfully and causes an I/O error. To avoid timing out, increase the time on the timeout setting.

Wildcard copy to a PDS or PDSE

To ensure that a wildcard copy of a PDS or PDSE is completed successfully, the PDS or PDSE member must be closed and dequeued (if necessary). For example, for the statement

```
$ cp smith.*  /u/smith/mnt/datalib
```

the wildcard copy will fail if any member inside of `smith.datalib` is open.

Limitations of a PDS

The PDS support in NFS adheres to the conventions used in MVS. For example, you cannot have more than one member of a PDS open for output at a time. If you try to create, remove, rename, or write a member of a PDS while another member is open for output, you get a “Permission denied” message.

A PDS member stays open for the timeout period specified in the appropriate timeout processing attribute, or until you try to create or write to another member.

Concurrent writes to a PDSE

NFS supports concurrent writes to a PDSE. If you are writing to one member of a PDSE, another client can write to any other member in the same PDSE. You can also use ISPF to edit a PDSE member while another NFS client is writing to a different member of the same PDSE.

Creating VSAM files

The NFS supports three types of VSAM files: key-sequenced (KSDS), entry-sequenced (ESDS), and relative record (RRDS). However, keyed access and relative-number access to the files are not supported.

If you plan to update a VSAM data set (for example, with the **vi** editor or with the **cp copy** command), the data set must have been defined with the **reuse** option. Trying to write back a VSAM data set that was not defined as reusable results in an “I/O error”, “failure to open”, or similar error message. If you create a VSAM file using the NFS, the reuse option is used by the server.

For more information on VSAM files, see *z/OS DFSMS Using Data Sets*.

In the following example, the attributes indicate that:

Spanned records are allowed

Organization is key-sequenced

Keys are 8 bytes long and start in position 0 of each record

Average record size is 1024

Maximum record size is 4096

Space is allocated for 50 records with a secondary allocation of 10

Cross-region and cross-system share options are provided

The file is to be created on a volume named D80CAT

```
$ cp ksds.old "ksds.new2,spanned,dsorg(indexed),keys(8,0),
   recordsize(1K,4K),space(50,10),shareoptions(1,3),
   vol(D80CAT)"
```

In the following example for creating a VSAM ESDS file, the attributes indicate that:

Spanned records are allowed

Organization is entry-sequenced

Average record size is 1024

Maximum record size is 4096

Space is allocated for 50 records with a secondary allocation of 10

Cross-region and cross-system share options are provided

The file is to be created on a volume named D80CAT

```
$ cp esds.old "esds.new3,spanned,dsorg(nonindexed),
   recordsize(1K,4K),space(50,10),shareoptions(1,3),
   vol(D80CAT)"
```

In the following example, the attributes indicate that:

- Spanned records are not allowed
- Organization is relative record, numbered in ascending order
- Average record size is 1024
- Maximum record size is 1024
- Space is allocated for 50 records with a secondary allocation of 10
- Cross-region and cross-system share options are provided
- The file is to be created on a volume named D80CAT

```
$ cp rrds.old "rrds.new4,nonspanned,dsorg(numbered),
   recordsize(1K,1K),space(50,10),shareoptions(1,3),
   vol(D80CAT)"
```

Exploiting SAM striped files

With SAM striping, data I/O is done in parallel to improve performance. For a file with 16 stripes, data is simultaneously processed on the first track of the allocated space in each of the 16 volumes. This allows for quick access to all the information.

The z/OS NFS server can support data set striping through the use of data class and storage class attributes that define extended format data sets. The z/OS NFS server can exploit the performance of extended format data sets by reading multiple blocks at a time when reading ahead.

For more information about striped files, see *z/OS DFSMS Using Data Sets*.

Exploiting large format data sets

Large format data sets are a type of physical sequential data set, other than extended format data sets, which can grow beyond a size limit of 65 535 tracks on each volume. That size limit applies to z/OS conventional (basic format) sequential data sets. Large format data sets can exploit the increased storage capacity of most hardware storage devices, and reduce the need for very large data sets to span multiple volumes.

To create a large format data set with z/OS NFS, specify a data class which has a dsntype value of *large*. The dsorg value must be ps, psu, or omitted.

For more information about large format data sets, see *z/OS DFSMS Using Data Sets*.

Exploiting data sets on extended address volumes

An extended address volume is a volume with more than 65 520 cylinders. NFS can read, write, and create VSAM data sets on extended address volumes.

For more information about data sets on extended address volumes, see *z/OS DFSMS Using Data Sets*.

Chapter 3. Using z/OS conventional MVS data sets

This topic explains what you need to know to use z/OS conventional MVS data sets on a client workstation. This topic discusses the following topics:

- Special MVS considerations
- Reading and writing MVS data sets
- Accessing MVS data sets
- Mapping between the workstation and MVS file systems.

In MVS, a file is called a data set, and the two terms are used interchangeably in this book. The z/OS NFS server presents information to you in the form of a UNIX (or AIX) file, though the information is actually stored on MVS-owned DASD in the form of an MVS data set.

Special MVS considerations

In addition to mapping between the workstation and the MVS file systems, the z/OS NFS server might be different from non-MVS NFS servers in other ways, including these differences:

- Selection of an MVS data storage format
- File size determination and time stamps
- Ownership and permissions
- Selecting MVS file systems versus z/OS UNIX file systems
- State
- File reading and writing
  - Random access to files
  - Cached data writing
- Case sensitivity–**maplower**, **nomaplower**
- Selection of text or binary processing modes–**text**, **binary**
  - Binary processing mode
  - Text processing mode
- Number representation
- Partial record identification
- Access to migrated files–**retrieve**, **noretrieve**; **wait**, **nowait**
- Access to system-managed migrated data sets
- File handle refresh
- File extension mapping

Selection of an MVS data storage format

The files you create with the z/OS NFS server are contained in MVS data sets. These MVS data sets are record-oriented and can be sequential, direct, VSAM, partitioned, and so forth. These MVS data sets are variable or fixed in record length. UNIX files, however, are byte-oriented and typically written or read at certain offsets in the file.

You can map non-MVS files to most types of MVS data set organizations. However, how the time stamps and file size value are handled depends on the

© Copyright IBM Corp. 1991, 2010

type of MVS data set used, and the file size processing can affect performance. See Appendix A, “File size value for MVS data sets,” on page 449.

Direct reads with record format **recfm(fbs)** or **recfm(f)** can be fast. In some cases, the z/OS NFS server can determine the physical block addresses from the record offsets. The MVS sequential file organization with **recfm(fbs)** or **recfm(f)** on DASD allows for efficient updating or reading at any offset in the file. Other supported MVS access methods (for example, VSAM) can be used if required by a given application but, in general, the sequential file organization is the best choice for files that are used mainly by UNIX clients.

File size determination and time stamps

How the z/OS NFS server handles the file size value and time stamps depends on the type of MVS data set used and the attributes used to access the data set. See Appendix A, “File size value for MVS data sets,” on page 449 and Appendix B, “Time stamps for MVS data sets,” on page 453.

Ownership and permissions

The UNIX UID and GID file attributes are reset to their default state (UID=0 and GID=0) after a restart of the z/OS NFS server or an unmount of the file system. In some cases, this requires that a superuser on the client workstation reissue **chown** and **chgrp** commands to reset the UID and GID. These commands can be included in the same script used to mount the file system.

The permissions checking done by RACF®, a component of the Security Server for z/OS, or an equivalent security package, is transparent to you. Access to a data set is granted, provided that the server's exports list, the MVS security subsystem, and the customized installation security exit allow access to the data set. Which of these security systems are active depends on the security settings used at your installation. The UNIX file modes or permission bits are ignored by z/OS NFS server and authorization is done with the RACF or equivalent security package.

UNIX's UID, GID, and MODE attributes are not used by the z/OS NFS server for checking user access to z/OS conventional MVS data sets (see previous paragraph). UNIX's **chown**, **chgrp**, and **chmod** commands do not update z/OS RACF security policies and will not alter access to files. Do not use returned values to determine access rights. The z/OS NFS server supports the setting and obtaining of these attributes to minimize impact to client applications. For performance, validation of passed values is limited to the following checks for proper operations.

When a new MVS data set is created, the UID and GID are inherited from the NFS RPC, or from z/OS UNIX segment, or from the RPC Authentication, in the listed priority order.

CHOWN
Request is failed with EPERM if changing to a value other than yourself. Change to yourself is allowed for mount support.

CHGRP
No checking.

CHMOD
No checking, new value ignored, existing z/OS NFS server value is left unchanged, “success” is returned to the client.

File reading and writing

After the z/OS NFS server is started and you have mounted the MVS data set, you can use regular data access or creation commands from your workstation to access files that reside on MVS.

For example, suppose you accessed an MVS file named `prefix.file3` mounted on the local directory /mnt. This is how you could use the UNIX **cat** command (or a similar command) to display the file:

```
$ cat /mnt/file3
```

Suppose you accessed an MVS file named `prefix.file12` mounted on the local directory /mnt. This is how you could use the UNIX **vi** command (or a similar command) to edit the file.

```
$ vi /mnt/file12
```

Writing a file on MVS is straightforward. If the file already exists, the file's existing attributes are used; they are not modified during the write operation. For the priorities of attributes, see "Overriding data set creation attributes" on page 23.

Random access to files

If your application accesses the files at random offsets, there is a performance implication.

In the UNIX environment, a file is represented as a byte stream. That byte stream is accessible for reading and writing at any byte offset for any byte length. In the MVS environment, a file is represented as a collection of records. The record, rather than a single byte, is the smallest object that can be processed. Therefore, the z/OS NFS server has to convert the byte stream operations from NFS clients into standard access method operations on MVS.

To convert byte stream operations to MVS access method operations, the server has to determine which record in the MVS file contains the offset specified in the NFS read or write request. To determine this, the server reads, mapping byte offsets to records, from the last known location in the file until the record containing the requested byte offset is located.

For example, suppose a file on MVS contains 10,000 variable-length records with a maximum length of 80 bytes for any record. Suppose the first NFS request received tells the server to read 4,000 bytes starting at offset 10,000 bytes. Because the file has not been opened yet, the server would open the file and start reading at the first record, searching for the record that contains offset 10,000. Once it found the record, the server would process the request, which might involve reading more records to find enough bytes to satisfy the request.

Another complication involved in mapping byte offsets to records is the processing defined by the user to apply to a file. For example, if you specify text mode processing with end-of-line terminators, the perceived offset into a file from a given client changes.

Cached data writing

The z/OS NFS might cache writes if out-of-sequence data packets are received or if a block of data is partially filled. If NFS is processing in the binary data mode, the writes will remain cached until one of the following occurs: the timeout for the request has been reached or the number of cached packets exceeds the number specified in **cachewindow**. If the NFS is processing text data, the writes remain cached until a timeout occurs. The missing data is padded with binary zeroes and record delimiters so that cached writes for text processing are written in the MVS data set on DASD at the location specified in each cached data packet. In the case of cached data packets for binary processing, only binary zeroes will be used to pad the missing data written at the specified location on DASD. See Table 4.

Attention: For the NFS version 3 commit procedure, the z/OS server will only support committing the cached data when the data set is timed out. For the NFS version 4 commit operation, the z/OS Server will only support committing the cached data upon receiving the close operation.

Table 4. Breakdown of text and binary writes

Description	Binary	Text
Data is flushed to DASD when the	Number of cached packets exceeds the amount specified in **cachewindow**, or the file times out.	File times out. If the number of packets exceeds the amount specified in **cachewindow**, all new out of sequence packets will be dropped.
Padding	Binary zeros	Binary zeros
Record delimiters	There are no record delimiters. Therefore, there is no attempt to add end of line characters.	There can be record delimiters. Therefore, an end of line character is added to the end of the record.

Case sensitivity–maplower, nomaplower

If the processing attribute **maplower** is specified, the MVS file name is mapped to the lower case when returned to the client and all client specified names are mapped to upper case. If the processing attribute **nomaplower** is specified in the attributes, all entries in the exports data set are case-sensitive. Therefore the client mount request must specify the MVS qualifier with the correct case to successfully match the exports data set entry.

Selection of text or binary processing modes–text, binary

You can specify either **text** or **binary** processing mode when you access files. This processing mode does not describe the type of data in the original file, but rather, it specifies whether to convert between ASCII and EBCDIC when sending file contents between the z/OS host and the client workstation. See “Mount command syntax and examples” on page 85 for more information about text and binary processing of files using the z/OS NFS client.

Binary processing mode

The **binary** processing mode specifies to send and receive file contents between the z/OS host and the client in binary form, avoiding the ASCII/EBCDIC conversion required in text mode. This is faster than text mode. However, users on MVS cannot read the file, because the contents are not in EBCDIC. Therefore, use the binary processing mode to create or access a file only if the file is not intended to be shared with users on the z/OS host, or the file content is binary.

When fixed-length records are written in binary mode, the server pads the last record of the file with null characters if the last record is less than the fixed record length. These padding bytes are counted in the file size.

Text processing mode

With the **text** processing mode, when data is read, record boundaries are marked with the end-of-line terminators such as **lf** or **crlf**. These terminator characters have the same data representation as the data that is read (the CCSID of the data read in is the same as the CCSID of the appended end-of-line terminators). The data representation of the record formatted data is then changed to the client representation (client CCSID). If the client data representation is the same as the data representation of the data that is read, then there is no data translation.

When the record formatted data is received from the client for writing to the data set, its data representation is changed to the data representation of the data stored in that data set, with the converted end-of-line terminators used to recognize the record boundaries.

In text processing mode, the representation of data along with end-of-line terminators is changed between client representation (client CCSID) and data set representation. All data and end-of-line terminators are converted according to the active translation table. Therefore, if the data set contains a mixture of characters and binary data, binary data is converted as well. In text mode, be careful not to mix your text data (characters) with binary data. Also pay attention that end-of-line terminator conversion depends on the Unicode Conversion Technique - convserv() attribute.

For example:

- With `convserv(LRE)`, ASCII LF 0x0a is converted to EBCDIC NL 0x15 (Language Environment behavior)
- With `convserv(RE)`, ASCII LF 0x0a is converted to EBCDIC LF 0x25

NFS version 4 protocol (NFSv4) differs from NFSv2 and NFSv3 protocol in handling single to multiple byte conversion. Therefore, the technique-search-order specified in the **convserv()** attribute should consider the effects of the NFS protocol being used. See“Creating the conversion environment for Unicode Services” on page 164 for further details.

Using the lf line terminator: For an AIX or UNIX client, use **lf** as a line terminator when using **text** processing mode.

Selection of how blanks are handled–blankstrip, noblankstrip: When fixed-length records are written in **text** mode, records are padded with blanks if the record length is larger than a line, and if the **blankstrip** processing attribute is enabled. (When sending data from MVS, **blankstrip** strips trailing blanks. When sending data to MVS, **blankstrip** pads the records with blanks).

If you are writing data to a fixed record length MVS data set in text mode with blank stripping enabled, and the data contains blanks at the end of the line, an I/O error occurs. This is because the server is not able to return the blanks to the client when the file is read back.

If you get an error message when trying to create or access an MVS data set, see Chapter 19, “Network File System messages,” on page 319 for further explanation of the message.

Potential fixed/variable/undefined record length data set logical I/O errors: If you save a fixed/variable/undefined record length MVS data set in **text** mode with one or more lines exceeding the maximum record length, an I/O error occurs. For example, suppose an MVS data set has fixed-length records of 80 bytes. After you edit the file using the **vi** editor on your workstation, one of the file’s records is 83 bytes long (exceeding the fixed length by 3 bytes). When you save the file back to the server, the MVS data set may be either partially or totally destroyed, and the “I/O Error” message appears on your screen. While you are still in the editing session, save the edited file in an alternate local file. After you correct the local file so that no line exceeds 80 bytes, save it back into the MVS data set.

For example, suppose a variable-length MVS data set is defined with an LRECL of 132 bytes. The maximum effective record length of the dataset is actually 128 bytes, because 4 bytes are reserved by DFSMS SAM to accommodate the record descriptor word (RDW). As a result, any line that is attempted to be written into this dataset that exceeds this maximum effective record length will result in an I/O error.

For more information about I/O errors related to different MVS dataset types, see *z/OS DFSMS Using Data Sets*

MVS prefix support

New for V1R11, the MVS prefix support enables you to explicitly specify a prefix for identifying MVS data sets.

All path type resolution (checking) processing uses the presence or absence of a leading slash (/) to indicate whether the path is an absolute or relative path. If the slash is present, the first qualifier after the slash is compared against the MVS prefix to determine if it matches the prefix. If so, then the path type will be considered to be explicitly resolved via the prefix. If no match is found, or no slash was present, the implicit path type resolution heuristic is used. To avoid causing unnecessary customer impacts, the default settings cause the file system type resolution algorithm to give the same results as the resolution algorithm in previous releases.

For more detailed information, see “z/OS NFS File System Type Selection” on page 9.

Number representation

The **text** processing mode does not change the number representation format between the host and client. When you choose **text** as the processing mode, the NFS converts characters between ASCII format and EBCDIC format, and processes end-of-line terminators, but no other translation of user data occurs.

When you select **binary** as the processing mode, NFS stores your data unchanged. Therefore, regardless of the processing mode you choose, you cannot change numbers from one client workstation’s format to another client workstation’s format.

Partial record identification

The term *partial record* or *incomplete record* applies to both z/OS NFS server text and binary modes whenever processing data blocks in RPC WRITE operations for the legacy path.

For text mode, the z/OS NFS server finds and extracts a record from the byte data stream sent by a client by checking sequentially for end-of-line (EOL) delimiters. The scope of a record being defined as a text partial record is limited to an RPC WRITE size of a data block.

An RPC WRITE data block usually contains several text records. If the last or the only record in the first RPC WRITE data block (starting at offset=0 of the byte data stream) has no tail with EOL then it is a text partial record. The second sequential

RPC WRITE data block (with the byte data stream offset = size of the first RPC WRITE data block) contains the tail of the partial record from the first RPC WRITE data block, and so on for further RPC WRITEs.

RPC WRITEs may come to the z/OS NFS server out-of-sequence, so there can be a lost RPC WRITE data block within the byte data stream and thus RPC WRITE data blocks cannot be chained together one-by-one by the z/OS NFS server. If an RPC WRITE data block is not the first one (offset not equal to 0) and the previous sequential RPC WRITE data block is lost, then the first record in this RPC WRITE data block may be incomplete as it has no beginning, just some tail bytes before EOL (or only EOL).

The z/OS NFS server treats a lost or absent RPC WRITE data block as containing a partial record by default.

For binary mode, there is no need to look for records or EOL in the byte data stream, but a lost or absent RPC WRITE data block leads to the same partial record problem (a very long record of the data stream size has data holes somewhere inside).

For both text and binary modes the z/OS NFS server counts the number of bytes of data stream arrived from a client and waits for a new RPC WRITE data block from the client for a specified writetimeout. After this writetimeout expiration, the z/OS NFS server flushes the record_structured cached data (raw data for binary mode) to disk, and closes/deallocates the data set. The z/OS NFS server fills in the "holes" caused by lost RPC WRITE data blocks with zeroes during data flushing. So if an RPC WRITE data block is truly lost (never retransmitted) to the z/OS NFS server, a closed data set will contain zeroes for that data portion that contains holes. The zeroes in a closed data set may be put by the z/OS NFS server at the end of a data set if the last record (referenced by the last data stream byte) is a partial record.

Also, data flushing may be initiated by the z/OS NFS server on RPC WRITE and COMMIT operations without the data set being closed or deallocated.

To indicate the cases of lost RPC WRITE data blocks or partial records, APAR OA16182 introduced special GFSA824W and GFSA825W messages that are issued to the console/log data sets after original writetimeout(n) expiration before data set closure/deallocation.

Symptoms of GFSA824W/GFSA825W messages

The GFSA824W message presents that at least one partial/lost record case has happened during the processing of a data set.

There is an explicit requirement for NFS users who use the z/OS NFS server text mode for data processing to end the byte data stream with an EOL delimiter for client applications writing to the z/OS NFS server. If a user's application does not comply with this requirement, it may provoke partial record conditions for the last RPC WRITE operation that corresponds to the end of data stream (even if all data from the client application was sent to the z/OS NFS server). If a user works with the z/OS NFS server in text mode and does not end its byte data stream with EOL, the GFSA824W/GFSA825W messages appear as a reaction on the last partial record case.

That partial/lost record case results in GFSA824W messages appearing and, subsequently, data set closure/deallocation problems. If a partial/lost record case happened in the beginning/middle of user data stream, the impact of data set closure/deallocation results is data corruption in the beginning/middle of a data set. The holes of the user data portion is filled in by zeroes by the z/OS NFS server.

If a partial/lost record case happened at the end of a user data stream (no EOL delimiter), the impact results in a prolonged timeout before data set closure/deallocation.

For data sets with F/FB record format, the last record in the data set is filled out with trailing zeroes. For example if a data set has RECFM=F,LRECL=40, but the last partial record is 20 bytes in the last RPC WRITE data block, the data set contains hex zeroes in the last physical partial record **xxxxxxxxxxxxxxxxxxxx....................**, where '.' is a x'00' hex byte.

For data sets with V/VB/U record format, the last record in the data set is filled in with one trailing zero. For example if a data set has RECFM=V,LRECL=40, but the last partial record is 20 bytes in the last RPC WRITE data block, the data set contains one hex zero in the last physical partial record **xxxxxxxxxxxxxxxxxxxx.**, where '.' is a x'00' hex byte.

The GFSA825W message presents that a data set is closed/deallocated after extended writetimeout with at least one partial/lost physical partial record.

If any late RPC WRITE operations with lost data portion are sent to the z/OS NFS server after the data set has been closed/deallocated, the data set will be re-opened, re-read and updated with the lost portion of data. But this late RPC WRITE should not alter the existing record structure in the data set. The shorter or longer record from the update against the existing data set records may cause logical errors in the z/OS NFS server that will be reported to a client as EIO(5) error code.

The z/OS NFS server supports correct recovery/extension of the last physical partial F/FB/V/VB/U record on disk during processing of further RPC WRITEs from a client which contains the rest of the record.

The z/OS NFS server does not support the last physical partial record recovery for VS/VBS data sets.

To eliminate conditions reported by GFSA824W/GFSA825W messages, the user should adhere to the EOL requirement for text mode and tune the z/OS NFS server by setting the proper timeouts, that is, writetimeout(n) value must be greater than the maximum delay between WRITE operations in a slow client/network. Improper writetimeout settings may cause performance problems and data set closure/deallocation problems due to partial/lost record conditions.

Access to migrated files–retrieve, noretrieve; wait, nowait

Sometimes files on MVS are migrated to another storage level, such as a space-saving format on DASD or tape. If your file has been migrated and you try to access it, it might take a while for it to be recovered back into primary storage. The **retrieve** and **noretrieve** processing attributes control what happens when you try to access a migrated file.

There are three ways that the **retrieve** or **noretrieve** option is controlled.

1. Using the Default Retrieve Attribute

You can use the default retrieve processing attribute by not entering **retrieve** or **noretrieve** in your **mount** command or **file access** command.

2. Specifying **retrieve** with the **mount** command

   You can issue the **mount** command, specifying **retrieve** or **noretrieve**. The attributes specified in the **mount** command override the attributes in the default attribute data set.

   In this example, migrated files under the mount point are not retrieved. However, you can access files under the mount point which are not migrated.

   ```
   $ mount mvshost1:"smith,noretrieve" /u/smith/mnt
   ```

   Conversely, the next command causes the migrated files under the mount point to be retrieved when accessing the files.

   ```
   $ mount mvshost1:"smith,retrieve" /u/smith/mnt
   ```

3. Specifying **retrieve** with a file access command

   You can issue a file access command with the attribute **retrieve** or **noretrieve** specified. The attributes specification in the **file access** command overrides the attributes in the **mount** command and the server default attributes.

   This command causes all files under the mount point /u/smith/mnt to be retrieved if they are migrated:

   ```
   $ ls -l "/u/smith/mnt,retrieve"
   ```

   This command, however, does not cause migrated files under the mount point /u/smith/mnt to be retrieved:

   ```
   $ ls -l "/u/smith/mnt,noretrieve"
   ```

Access to migrated system-managed data sets

z/OS DFSMS allows access to data set attributes for migrated SMS-managed data sets, without having to recall the data set if the data set was migrated under DFSMS/MVS V1R3 or later. Supported data set types are SMS-managed PS, VSAM ESDS, VSAM KSDS, VSAM RRDS, PDS, and PDSE. Migrated PDS/PDSE members are not supported.

The z/OS NFS server is able to obtain the attributes of a supported SMS-managed migrated data set without recalling the data set. Attributes such as the time stamp and file size are saved to DASD. Subsequent file size requests do not cause a recall of the supported SMS-managed migrated data set, thus improving performance. However, when the data set is modified outside the server by a non-NFS application (for example, by the TSO/E editor) before it was migrated, the stored file size could be incorrect. When the data set is accessed again by the server, a recall must be done to determine the correct file size.

When a request is made to remove any of the supported SMS-managed migrated data sets, the data set will be deleted without recall. For PDS and PDSE migrated data sets, the data set will be recalled in order to read its member information.

File handle refresh

File handles of mounted objects (directories or file systems) are saved on DASD in a mount handle data set and are automatically established again when the server restarts. However, file handles for the files within a mounted object are kept in virtual storage (memory) and they are lost if the server restarts. This may result in stale file handles for NFS version 2 and 3, or file handle expired in NFS version 4, and the clients may be required to request a new file handle by redoing the **lookup** on the file.

Mapping between the workstation and MVS file systems

In MVS, a file is called a data set, and the two terms are used interchangeably in this book. The z/OS NFS server presents information to you in the form of a UNIX (or AIX) file, though the information is actually stored on MVS-owned DASD in the form of an MVS data set.

The files for a computer system are organized into a file system. The UNIX environment uses a file system that is a hierarchy of directories. MVS, however, uses a non-hierarchical file system in which groups of data sets are referred to by specifying a high-level qualifier.

The MVS high-level qualifier can include the first (leftmost) qualifier of data sets, or the first and second qualifiers, or the first, second, and third qualifiers, and so on. For example, `SMITH` is the high-level qualifier for the files named `SMITH.TEST.DATA` and `SMITH.PROJ7.SCHED`, while `SMITH.TEST` is the high-level qualifier of `SMITH.TEST.DATA` and `SMITH.TEST.DOCS`.

File extension mapping

File extension mapping allows users to access members of z/OS conventional MVS PDS or PDSE data sets on the z/OS host that are mapped from client machine files that contain file extensions. File extension mapping also allows the selection of text or binary processing for members of z/OS conventional MVS data sets and z/OS UNIX files.

Each PDS or PDSE data set on the host can only be mapped with one unique file extension. For example: IBMUSER.TEXT(M1), IBMUSER.TEXT(M2) will map to m1.txt, m2.txt under directory ibmuser.text on the client machine. This capability allows client machine tools such as editors and compilers, to process host files remotely without modification. There are site and processing attributes and an operator command that are associated with the file extension mapping.

The rules for file extension mapping are contained in a data set called a *side file*. The side file consists of two sections: NFS.MAPPING and NFS.MAPPING.MAPPED.

Section NFS.MAPPING is used to define file extension mapping for z/OS conventional MVS PDS/PDSE data sets. The client user can specify the **fileextmap** attribute to turn the file extension mapping on and the **nofileextmap** attribute to turn the file extension mapping off. Section NFS.MAPPING.MAPPED is used to specify whether text or binary processing is performed for file extensions of z/OS conventional MVS data sets and z/OS UNIX files. The client user can specify the **mapped** attribute to specify when a mixed set of data types is to be processed. For z/OS conventional MVS data sets, the **maplower** and **nomaplower** attribute controls the mapping of lower case files names to upper case when accessing files on z/OS.

You can establish a default side file, with the default rules for file extension mapping, by specifying the **sidefile(dsname)** attribute in the attributes data set. A client user can also specify this attribute on a mount command to override the default side file name, as shown in the following example:

```
[C:\] mount z: mvshost1:"user1.pds,sidefile(hlq.nfs.mapping)"
```

The side file specified at the **mount** command will be searched first followed by the default side file. The system administrator can specify the maximum space available for side files using the site attribute **sfmax(n)**. The mapfile operand of the modify command can be used to have a side file read again and rebuilt without stopping and restarting the server or remounting of mount points. A sample mapping side file is provided as GFSAPMAP in the SYS1.NFSSAMP library. See "Entering operands of the modify command for the z/OS NFS server" on page 222, "Processing attributes syntax" on page 121, and "Site attributes syntax" on page 134 for more information.

Specifying the side file on an NFS version 4 **mount** command has the following effects:

- Loads the side file if it is not loaded
- Does not reload the side file if it was already loaded (does not change the current side file).

Note: With NFS version 4, an **unmount** command does not unload the side file, because no UNMOUNT_RPC is sent to the z/OS NFS server.

Mounting of MVS data sets onto a client mount point

To access an MVS file system from the client, you use the **mount** command to create a temporary link between specific MVS data sets and a UNIX directory (preferably empty) or an unused logical drive. The UNIX directory or drive is called a mount point.

You use an MVS high-level qualifier in the mount command to specify which MVS data sets to mount onto a mount point. The MVS data sets beginning with the specified high-level qualifier appear as files under the mount point. See Figure 2 on page 43.

You can also perform a mount using a fully qualified partitioned data set name (PDS or PDSE) or an alias to a user catalog, but not the catalog name itself. Only cataloged data sets are supported by the z/OS NFS server, and tape data sets are not supported.

Data set organizations supported include:

- Physical sequential (PS)
- Direct access (DA)
- Partitioned data sets (PDS)
- Partitioned data sets extended (PDSE)
- VSAM KSDS
- VSAM ESDS
- VSAM RRDS
- SAM extended format data sets

Both SMS-managed and non-SMS-managed data sets are supported.

- NFS supports multivolume data sets.
- Generation data sets are not supported.

Notes:

1. The filesize for the MVS z/OS conventional data set as a directory has a dummy size with a value of 8192.
2. For the NFS Version 3 CREATE procedure, the z/OS server does not harden the exclusive create verify token to disk.

Mount examples

Table 5 shows how to mount z/OS MVS data sets from various platforms.

Table 5. Examples of mounting MVS data sets from clients

Clients	Command Examples
UNIX	mount -o vers=n,sec=*r*,proto=*x* mvshost1:"MVSHLQ,procattr1,procattr2,...." /u/smith/mnt
Linux (NFS version 4)	mount -t nfs4 -o sec=*r*,proto=*x* mvshost1:MVSHLQ /u/smith/mnt
Linux (NFS version 2 and 3)	mount -o vers=*n* ,proto=*x* mvshost1:MVSHLQ /u/smith/mnt
Windows	nfs link z:"\\mvshost1\MVSHLQ,procattr1,procattr2,..." id pw
Any (UNIX, Linux, Windows), with explicit MVS prefix	mount ... mvshost1:"/mvs/MVSHLQ"...

In the examples:

Operand **Description**

mvshost1
Specifies the name of the z/OS host.

/mvs
Assuming that MVSPREFIX(/mvs) is set in the site attribute file (this is the default), identifies this as an MVS mount. Note that this does not need to be **/mvs**. It only needs to match the MVSPREFIX value (case-insensitive).

MVSHLQ
Specifies the high-level qualifier of the MVS data set.

procattr
Specifies any valid processing attribute, such as text/binary or hfsfbtimeout(n).

/u/smith/mnt
Specifies the local mount point.

z:
Specifies the drive letter on the Windows system.

id pw
Specifies login id and password for pcnfsd.

-t nfs4
Specifies NFS protocol version 4 for Linux (optional).

-o vers=n
Specifies the NFS protocol version to be used (2 or 3 for Linux; 2, 3 or 4 for others) (optional).

-o sec=r
Specifies RPCSEC_GSS security flavors, which is available only on the z/OS NFS version 4 server. Valid options are **sys**, **krb5**, **krb5i**, and **krb5p**.

-o proto=x
Specifies the transport protocol for the NFS client to communicate with the NFS server. Valid options are **tcp** or **udp**. (Note for IPv6, some platforms use proto=tcp6 instead of tcp)

Variants of the mount command

The name of the command that performs the mount operation varies for different client implementations. For example, the Windows implementation uses the **nfs link** command while most UNIX and Linux implementations use the **mount** command.

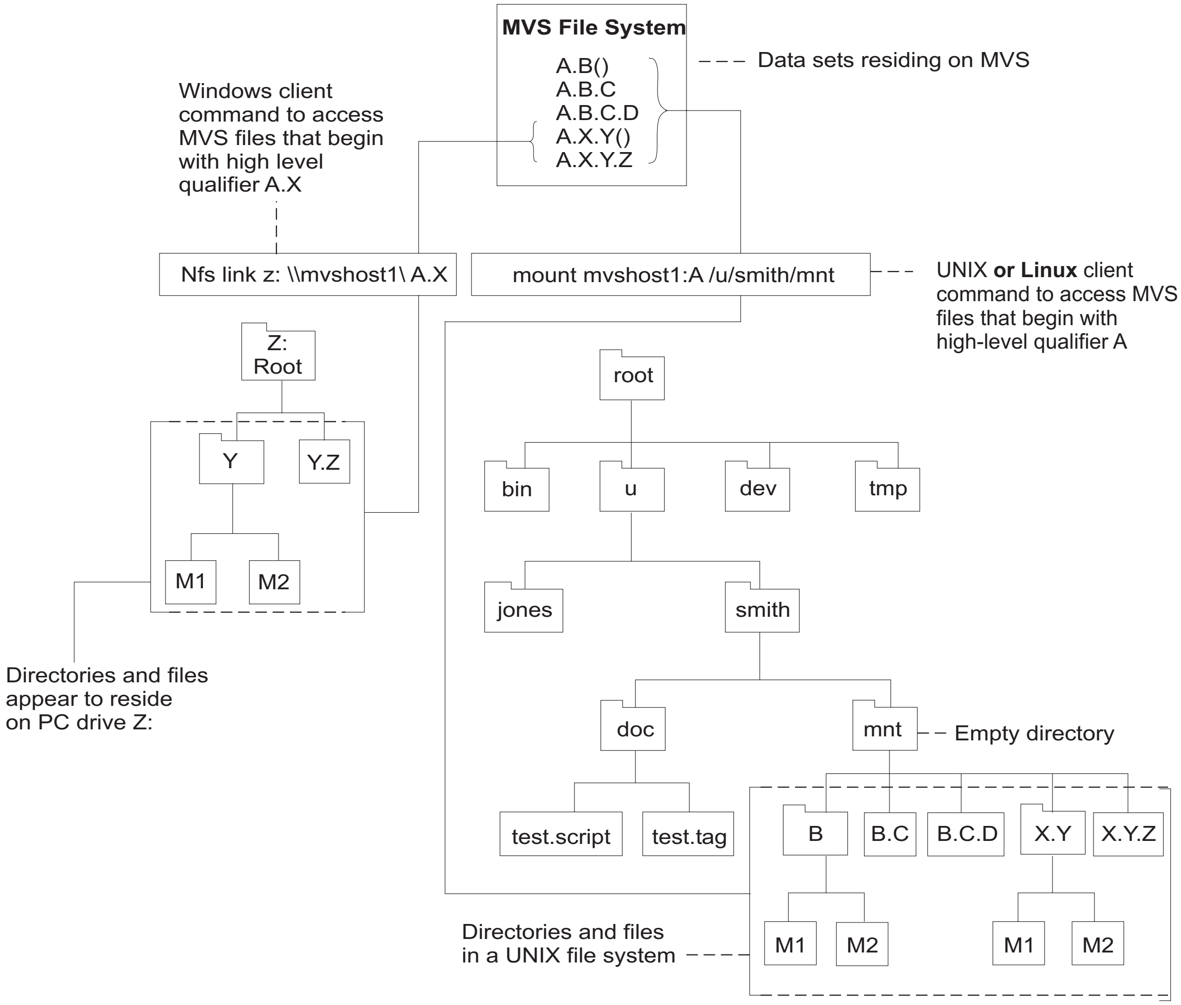

MVS File System
A.B()
A.B.C
A.B.C.D
A.X.Y()
A.X.Y.Z
Data sets residing on MVS
Windows client command to access MVS files that begin with high level qualifier A.X
Nfs link z: \\mvshost1\ A.X
mount mvshost1:A /u/smith/mnt
UNIX or Linux client command to access MVS files that begin with high-level qualifier A
Z: Root
Y
Y.Z
M1
M2
Directories and files appear to reside on PC drive Z:
root
bin
u
dev
tmp
jones
smith
doc
mnt
Empty directory
test.script
test.tag
B
B.C
B.C.D
X.Y
X.Y.Z
M1
M2
M1
M2
Directories and files in a UNIX file system

Figure 2. Examples of mounting MVS data sets on Windows, UNIX and Linux clients. The Windows client is mounting the MVS data sets which begin with the high-level qualifier "`A.X`". The UNIX and Linux clients are mounting the MVS data sets which begin with the more inclusive high-level qualifier "`A`". The parentheses indicate a PDSE containing the members M1 and M2. The PDSEs `A.B()` and `A.X.Y()` appear as directories, and their members appear as files within those directories.

Use of a PDS or PDSE as a directory

If the data sets specified include partitioned data sets, a second level of hierarchy is shown. This allows you to define one level of directories under the mount point. Thus, you can issue **mkdir** to create a directory (stored as a PDS or PDSE) and then create files (stored as members of a PDS or PDSE) within that directory.

This use of a PDSE is shown in Figure 2, which illustrates the mapping of file names between client file systems and the MVS file system resulting from a **mount** command.

Use of multiple mount points

You can arrange groups of data sets into several UNIX mount directories (or PC mount drives) by using MVS naming conventions to mount specific data sets at each mount point. For example, you could mount user1.project1 to get all data sets beginning with user1.project1 mounted at one point in the local file system, and you could mount user1.project2 at another point. This would create the effect of two distinct directories (or drives), one containing the user1.project1.* data sets, and the other containing the user1.project2.* data sets.

Data set serialization and sharing

The z/OS NFS server handles data set serialization and sharing differently depending on the type of data set:

Physical sequential
: The server insures the read/write integrity of a physical sequential data set by SVC 99 dynamic allocation with exclusive option whenever a physical sequential data set is opened for output; otherwise it is allocated with share option.

VSAM data set
: The server dynamically allocates a VSAM data set with share option and allows the VSAM access method to manage data sharing using the SHAREOPTIONS specified during data set definition.

PDSE data set
: The server dynamically allocates a PDSE data set with share options and allows the PDSE functions to manage the integrity of the PDSE data set and its members.

PDS data set
: The server dynamically allocates a PDS data set with share option and surrounds the PDS and its members with exclusive ENQs against the QNAME=SPFEDIT and RNAME=data set name. This does not protect the PDS from other z/OS users who are attempting to access the PDS without performing ENQ against SPFEDIT similar to the z/OS NFS server.

NFS protocol

Table 6 illustrates that the NFS procedures are not all supported for z/OS conventional MVS data sets.

Table 6. NFS procedures

Procedure	Version 2 Protocol	Version 3 Protocol	Version 4 Protocol
link	no	no	no
mknod	N/A	no	no
readlink	no	no	no
readdirplus	N/A	no	no
setattr	yes	yes	yes
statfs	yes	N/A	N/A
symlink	no	no	no

Note: Setattr only supports filesize=0 truncation and UNIX permission is set to 777.

NFS file system attributes

The z/OS NFS server generates MVS-specific values for certain UNIX file system attributes. Table 81 on page 457, Table 82 on page 457, and Table 83 on page 457 illustrate the MVS values that the z/OS NFS server generates.

Delegating management of a file's resources to an NFS client

The NFS version 4 protocol enables an NFS Server to temporarily delegate management of a file's resources to an NFS Client. The key purpose of the delegation is to provide improved performance by eliminating communications with the NFS Server.

When a file’s management is delegated to an NFS Client, all file access requests can be managed locally by the NFS Client while the file is delegated.

Use the following NFS Server Site Attributes to control the activation of the delegation function:

DELEGATION/NODELEGATION
specifies whether or not the NFS server allows NFS Version 4 protocol file management delegation to NFS clients. The default setting is NODELEGATION.

A new Modify operator command, V4DELG=ON/OFF can be used to dynamically turn on or off NFS Version 4 protocol file management delegation to NFS clients.

- When Modify operator command V4DELG=ON is specified, the server checks each client for a valid callback path. This will take some time. The server can only grant the delegation if a valid callback path exists between the server and the client.
- When Modify operator command V4DELG=OFF is specified, any existing delegations continue until they are recalled by the NFS server due to a conflicting request, or returned by the NFS client. However, no new delegations are started.

File delegation is at the NFS Server’s discretion. When an NFS Client sends a file open request, the server determines whether to delegate the file or not. The NFS Client cannot assume that the file will be delegated.

When the client has not contacted the server for a lease time interval, the server can remove all delegations to the client.

If a local user or another client requests share reservations or access to the delegated file, the server will recall the delegated file.

The NFS Server may recall the delegation at any time. The NFS Client must then send any modified data buffers and attributes back to the server. It should also send any locally established file locks to the server so that those locks can be established on the NFS Server to maintain the Client’s lock status over those files.

When the delegation is granted to the client, the result is as follows:

- Request for open Read share - Granted: Read delegation (READ Delegation)
- Request for open Write share - Granted: Read and Write delegation (Write Delegation)

When the file is closed, the delegation returns back to the server.

When another application on the z/OS system requires access to the delegated file, the system must notify the NFS Server that the delegation is being recalled. The server must then recall the file from the delegated NFS Client(s). Any modified data buffers and file attributes must be updated, and any necessary file locks must be established before the recall can be completed. .The NFS server initiates a delegation recall during the following circumstances:

- If the NFS server receives an NFS client access request for a file and this request is not compatible with existing client delegations for the file, then the existing delegations are recalled and access is granted to the interested clients in a non-delegated mode, on a normal file open priority order.
- If an NFS Client does not respond to a recall request within a reasonable amount of time, the delegation is treated as having been successfully recalled from the standpoint of any other delegation requests. At that point, any subsequent file access operation requests from this client will fail.
- When an NFS Client's lease expires, the z/OS NFS Server recalls any outstanding delegations as part of the lease expiration process.

Notes:

1. Currently delegation is only supported for MVS data sets. zOS Unix file delegation is not provided.
2. Only the AUTH_UNIX RPC flavor is supported for delegation.
3. NFS V4 delegation reclaim after an NFS Client reboots (that is, Open claim type, CLAIM_DELEGATE_PREV) is not supported due to potential issues if the client reboots in close succession.
4. z/OS NFS Server NFSV4 MVS data set delegation function does not support the recall of PDS/PDSE member delegations as a result of local z/OS user attempts to perform such data set level operations as data set rename, data set remove, and data set compress. By contrast, local z/OS user attempts to access a member causes the member delegation to be recalled when appropriate. Note that the initial access attempt by the user may fail, but another attempt a short time later should succeed once the delegation recall has completed.

Chapter 4. Using z/OS UNIX System Services files

This topic explains what you need to know to access z/OS UNIX files from a client workstation.

- The z/OS UNIX file system
- POSIX compatibility
- Attributes specific to z/OS UNIX
- Protecting your z/OS UNIX files
- Accessing z/OS UNIX files from the client
- Linking an MVS data set to a hierarchical file system
- Selecting z/OS UNIX file systems versus MVS file systems
- UNIX look and feel
- Displaying and modifying remote file system access control lists

For detailed information about z/OS UNIX, see *z/OS UNIX System Services User's Guide*.

z/OS UNIX file system

z/OS UNIX provides a hierarchical file system (HFS) for z/OS. z/OS UNIX also provides the z/Series File System (zFS). An HFS or zFS file within z/OS UNIX is called a z/OS UNIX file. HFS and zFS files are organized in a hierarchy of files and directories in a tree structure. A directory can contain files or other sub-directories. The highest level directory is called the root directory. Figure 3 on page 48 shows an example of mounting an HFS or zFS directory from a UNIX client.

© Copyright IBM Corp. 1991, 2010

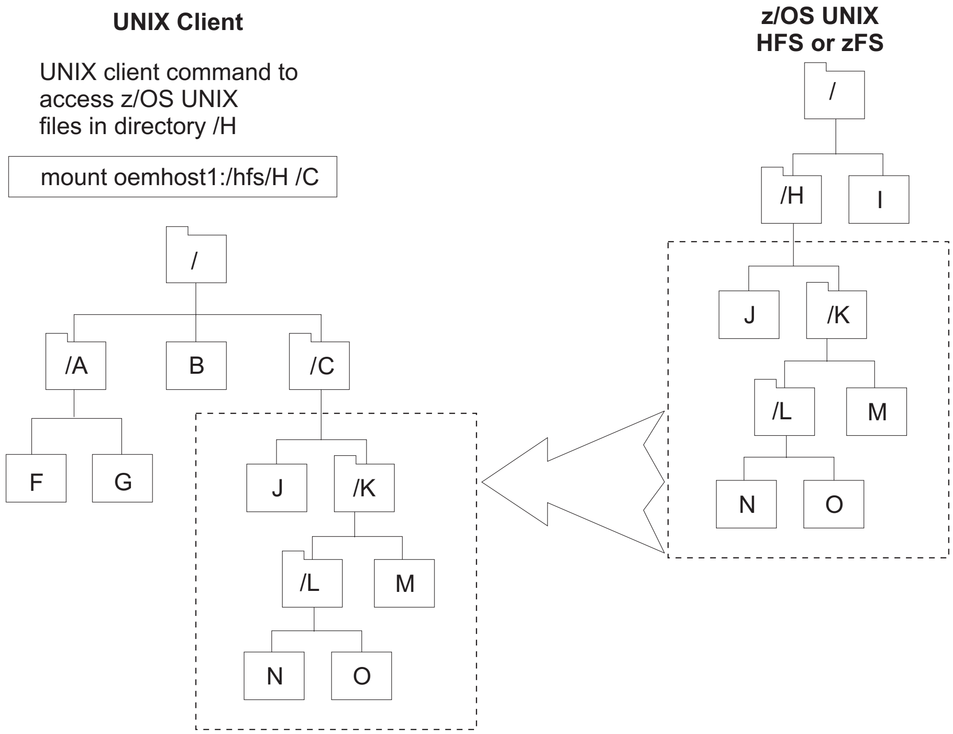


Figure 3. Example of mounting an HFS or zFS file from a UNIX client

A z/OS UNIX file system must be mounted by an MVS system operator using a TSO MOUNT command before that z/OS UNIX file system can be mounted by an NFS client through the z/OS NFS server. If a z/OS UNIX file system is mounted after the NFS client mount is already established, this file system's directory structure and contents will not be visible to the remote NFS Client until the NFS mount is unmounted and remounted or the z/OS NFS Server is restarted.

z/OS UNIX files are byte-oriented rather than record-oriented (unlike z/OS conventional MVS data sets). This data can be shared with TSO/E z/OS UNIX users in addition to NFS clients. All data written to a z/OS UNIX file can be read by all programs as soon as it is written. You can also copy data between z/OS UNIX files and MVS data sets using z/OS UNIX utilities like ISHELL.

POSIX compatibility

The NFS supports file access to the z/OS UNIX file system. z/OS UNIX supports a set of standards called the portable operating system interface (POSIX). See *z/OS UNIX System Services User's Guide* for more information about POSIX compliance. With z/OS UNIX, the NFS performs the following functions.

- Supports hierarchical directories
- Allows file names up to 255 characters in length
- Allows path names up to 1023 characters in length
- Supports mixed-case names and special characters, except nulls, slashes, and commas in file and path names
- Supports UNIX-style file access permissions
- Supports group ID and user ID at the file level
- Supports the full NFS protocol (including external links)
- Enables data sharing between clients and the z/OS UNIX

- Enables you to link z/OS conventional MVS data sets to a POSIX path name

This support incorporates the basic strengths of the z/OS system for both existing MVS data and applications and for new POSIX conforming data and applications.

NFS protocol

z/OS UNIX is compliant with all of the z/OS Network File Systems version 2, version 3 and version 4 protocols.

Attributes specific to z/OS UNIX System Services

The following attributes are specific to the z/OS UNIX:

async	Processing attribute for version 2 protocol only
sync	Processing attribute for version 2 protocol only
hhfsprefix	Site attribute
extlink	Attribute, see "Linking an MVS data set to a z/OS UNIX file system" on page 53

Note: These attributes are also explained in Chapter 9, "Initialization attributes for the z/OS NFS server," on page 115.

Synchronous write to a z/OS UNIX file for NFS version 2 protocol

Use the **sync** and **async** processing attributes to specify whether data received by a write request for a z/OS UNIX file object is committed to nonvolatile storage before the write response is returned to you.

If **sync** is specified for a z/OS UNIX file object, the data is written to z/OS UNIX and immediately committed to non-volatile storage.

For greater throughput, you can alternatively specify **async**. Your data is then committed to the disk some time after the write request is received from the NFS client. Your data is written to disk when the write timeout occurs, or if z/OS UNIX reclaims buffer cache storage.

The **sync** and **async** processing attributes only apply to z/OS UNIX data access. They are ignored for MVS data set access. A TSO/E z/OS UNIX user doesn't have to wait for the data to be committed to nonvolatile storage before accessing. z/OS UNIX maintains a central buffer cache and a TSO/E z/OS UNIX user can access the data as if it were in the file once it has been written by the z/OS NFS server.

Synchronous write to a z/OS UNIX file for NFS version 3 or 4 protocol

For the NFS version 3 WRITE procedure or NFS version 4 WRITE operation there is a processing argument ***stable*** and output parameter ***commit*** which specifies whether data received by a write request for a z/OS UNIX file object is committed to nonvolatile storage before the write response is returned to you.

If the ***stable*** processing argument is used during the write procedure, there are three modes when the write procedure writes to a file:

file_sync
The z/OS NFS server must commit all data written and all file system data to stable storage before returning commit results.

data_sync
The z/OS NFS server must commit all data written and sufficient metadata to enable retrieval of data, before it returns a reply to the client.

unstable
The z/OS NFS server may not commit any part of the data and metadata to stable storage, before returning a reply to the client. The data will be committed when a timeout occurs.

For the commit procedure, the z/OS server will support committing the entire data and metadata to stable storage.

Authorization checking when writing to a z/OS UNIX file

The z/OS Network File System server allows the owner of an z/OS UNIX file to write to the file regardless of the UNIX permission bits setting on the file.

HFSPREFIX site attribute

The HFS Prefix is not required, depending on the implicit prefix selection algorithm specified in the IMPPREFIX site attribute.

If the system administrator has not specified an implicit prefix selection, then to access z/OS UNIX files, you must know the HFS prefix defined by your system administrator (the default is /hfs). You can use the **showattr** command to display the HFS prefix defined for your location. You use this prefix in your mount command before the path name of the z/OS UNIX directories that you are mounting. The HFS prefix is used by the NFS server to distinguish z/OS UNIX directories from z/OS conventional MVS data sets, but the HFS prefix isn't part of the path name that you see. After you have entered the mount command, you access HFS files using the local mount point.

See “z/OS NFS File System Type Selection” on page 9 for more information on selecting NFS file system types.

Protecting your z/OS UNIX System Services files

As an z/OS UNIX user, you can control the read, write, and execute access to your files by other users in and outside of your group by setting the permission bits associated with the files.

To access z/OS UNIX files from the NFS, you must be defined as an z/OS UNIX user. The system programmer defines you as a z/OS UNIX user by assigning a z/OS UNIX user ID (UID) and a z/OS UNIX group ID (GID) to you. The UID and GID are numeric values associated with a TSO/E user ID. The values are set in the RACF user profile and group profile when you are authorized to use z/OS UNIX. The system uses the UID and GID to identify the files that you can access. Your specific UID value identifies you as a user of z/OS UNIX services. A GID value is a unique number assigned to a group of related users. These numbers appear in the RACF user profile. See *z/OS UNIX System Services Planning* for more information.

Accessing z/OS UNIX files from a client

Most of the commands that are used to access z/OS UNIX files are identical to the commands that are used to access z/OS conventional MVS data sets.

mvslogin
showattr
mount
umount
mvslogout

The only command that is changed for z/OS UNIX is the **mount** command.

Note: The syntax of these commands may vary between platforms; see the appropriate topic for examples specific to the platform you are using to access z/OS UNIX files.

If you are using AIX (or any other UNIX-based operating system) see Chapter 6, "Commands and examples for AIX and UNIX clients," on page 63.

Mount examples

Table 7 shows how to mount z/OS UNIX files from various platforms.

Table 7. Examples of the mount command for clients

Clients	**Command Examples**
AIX, UNIX, Solaris	mount -o vers=*n*,sec=*r*,proto=*x* mvshost1:"/hfs/smith" /u/smith/mnt
Linux with NFS V4	mount -t nfs4 -o sec=*r*,proto=*x* mvshost1:"/hfs/smith" /u/smith/mnt
Linux with NFS V2 or V3	mount -o vers=*n* ,proto=*x* mvshost1:"/hfs/smith" /u/smith/mnt
Windows	nfs link z: '\\mvshost1\hfs/smith' /n id pw nfs link z: "\\mvshost1\hfs/smith,procattr1,procattr2,...." /n id pw
Any (UNIX, Linux, Windows), with implicit prefix	mount .... mvshost1:/smith ....

In the examples:

Operand	**Description**
mvshost1	Specifies the name of the MVS host.
/hfs	Specifies the HFS prefix. Note that the HFS Prefix is not required, depending on the implicit prefix selection algorithm specified in the IMPPREFIX site attribute.
/smith	Specifies the HFS directory to be mounted.
/u/smith/mnt	Specifies the local mount point.
-t nfs4	Specifies NFS protocol version 4 for Linux (optional)
-o vers=n	Specifies the NFS protocol version to be used (2 or 3 for Linux; 2, 3, or 4 for others) (optional)
-o sec=r	Specifies RPCSEC_GSS security flavors, which are available only on the z/OS NFS version 4 server. Valid options are **sys**, **krb5**, **krb5i**, and **krb5p**.

-o proto=x Specifies the transport protocol for the NFS client to communicate with the NFS server. Valid options are **tcp** or **udp**. (Note for IPv6, some platforms use proto=tcp6 instead of tcp)

/n Specifies the NFS protocol version for Windows (2, 3, or 4) (optional)

id pw Specifies login id and password for pcnfsd.

Notes:

1. The */hfs* prefix value is used by the z/OS NFS server to determine if a file is a z/OS UNIX file, and does not appear in the path name of an HFS file once it is mounted.
2. The HFS prefix is not required, depending on the implicit prefix selection algorithm specified in the IMPPREFIX site attribute.

z/OS UNIX data transfer and conversion

With the NFS version 4 protocol, text data and metadata are transferred between the server and client in the UTF-8 data format (ASCII text is not transferred directly). z/OS NFS conversion of UTF-8 text data and metadata requires setting up a conversion environment using the z/OS Unicode Services by creating a Unicode conversion image that defines conversion tables with UTF-8 [CCSID 1208].

Data transfer under the NFS version 4 protocol

With the NFS version 4 protocol, text data and metadata are transferred between the server and client in the **UTF-8** data format. With NFS version 4, ASCII text is not transferred directly.

Text or binary processing - NFS version 2 and 3 protocols

z/OS UNIX is a byte-oriented, hierarchical, EBCDIC file system. For the NFS version 2 and version 3 protocols, the z/OS NFS server provides ASCII to EBCDIC text translation. For these versions of NFS, if you are just using the mainframe as a repository for your workstation (ASCII) data, you should use the binary mode to speed processing. If you use text mode, data from your workstation is converted into EBCDIC when it is stored on the mainframe. Conversely, when the z/OS NFS server returns the data to your client system, it converts the data back into ASCII. The conversion can slow processing, but might be necessary if you are sharing data with other MVS users. All data is converted according to the active translation table. Therefore, if the data set contains a mixture of characters and binary data, binary data is converted as well. In text mode, then, be careful not to mix your text data (characters) with binary data.

In text mode, you can either use the OEMVS311 translation table or the replacement customized translation table to convert data between ASCII and EBCDIC. If you are using z/OS UNIX and text mode processing, specify the OEMVS311 translation table with the xlat processing attribute. The OEMVS311 table translates ASCII (ISO 8859-1) to and from EBCDIC (1047 - z/OS UNIX System Services). TCP/IP for MVS version 3.1 provides the OEMVS311 table. This table translates the UNIX line terminator (lf) to the z/OS UNIX line terminator (nl). See *z/OS Communications Server: IP Configuration Reference*for more information about creating and customizing your own translation tables.

This is an example of specifying the OEMVS311 translation table during a mount:

```
$  mount 1stc3mvs:"/hfs/usr/man/C,text,xlat(oemvs311)" /mnt
$  export MANPATH=/mnt
$  man more
```

In this example:

Operand	**Description**
1stc3mvs	Specifies the name of the z/OS host.
/hfs	Specifies the HFS prefix.
/urs/man/C	Specifies the HFS directory to be mounted.
text	Specifies that data be converted between ASCII and EBCDIC
xlat(oemvs311)	Specifies that the translation table named OEMVS311 is used to convert data between ASCII and EBCDIC.
/mnt	Specifies the local mount point.
man more	Obtains a Man Page description of the **more** command

Linking an MVS data set to a z/OS UNIX file system

This section explains how to access an MVS data set through a z/OS UNIX path name by using the external link command. It also explains how to display the contents of an external link and how to delete an external link.

Creating an external link

You can create an external link to an MVS data set, and then transparently access the MVS data set by referencing the external link. The external link simulates a UNIX-like hierarchical naming convention for z/OS conventional MVS data sets. This is done using the **ln** command, for example:

```
mount mvshost1:USER1 /mnt
mount mvshost1:/hfs/u/nfs /samples
ln -s USER1.MVSFILE /samples/linkfile,"extlink"
```

In this example a z/OS UNIX file object, */linkfile*, of the file type "extlink" is created containing the file name of the MVS data set USER1.MVSFILE to be accessed. The source file must be mounted to a z/OS UNIX file system. The external link must reference an MVS data set. All future references to */samples/linkfile* access USER1.MVSFILE transparently.

In this example the file */usr/pub/myfile* is copied to the MVS data set USER1.MVSFILE that is contained in the external link */samples/linkfile*:

```
cp /usr/pub/myfile /samples/linkfile
```

Your installation should make sure that the appropriate security permissions have been obtained to access the MVS data set. You will receive "Permission Denied" message if the mount point */mnt* has not been established on USER1.

A mount point must be established before the external link is established. Otherwise, the error code ACCESS DENIED is returned. For physical sequential data sets, the high level qualifier of a data set must be established. For example, if you had a file called smith.test.data you can mount with smith, smith.test, or smith.test.data as your high level qualifier. For PDS and PDSE data sets, the fully qualified name must be established as a mount point. An example of a fully qualified name would be, smith.test.data.

Displaying the contents on an external link

You can display the contents of an external link by appending the "extlink" sequence to the external link path name. This permits the user to inspect the contents of the external link with the **ls -l** command.

This example shows how to display the attributes and contents of the external link `/samples/linkfile`:

```
ls -l /samples/linkfile,"extlink"
lrwxrwxrwx  1 user1 13 Jun 17 20:43 /samples/linkfile ->USER1.MVSFILE
```

This example shows how to display the attributes of the MVS target data set `USER1.MVSFILE`:

```
ls -l /samples/linkfile
-rw-rw-rw-  1 root  2112 Sep 28 13:50 /samples/linkfile
```

Deleting an external link

The external link file object is deleted with the remove request:

```
rm /samples/linkfile,"extlink"
rm /samples/linkfile
```

Either **rm** command results in the z/OS UNIX external link file object alone being removed. The target MVS data set is not affected.

Accessing symbolic links on z/OS NFS version 4

The issues associated with accessing symbolic links on z/OS NFS version 4 due to two file system types has been resolved in z/OS V1R11. This has been done with enhanced prefix support. See “z/OS NFS File System Type Selection” on page 9 for more information.

When using an Exports file (Security(EXPORTS) or Security(SAFEXP) mode), both the initial path containing the symbolic link and the target path must be exported. Otherwise, the mount will fail. This is necessary because in NFS v4, the initial mount emulation (lookup) processing proceeds until the symbolic link is found. If that is not exported, that initial mount processing will fail. Once the symbolic link is discovered, the NFS client starts over with the mount emulation (lookup) processing using the target path name. If that path is not exported, then that mount processing will fail.

The NFS version 4 protocol (NFSv4) does not include a mount operation. Instead, mount requests are processed as a series of lookup operations starting from the root node. If the mount request includes a symbolic link in the path name, the fact that the node is a symbolic link is returned to the NFS Client. It is the client's responsibility to read the link to get the defined data path and then restart the lookup process starting from the root node, but using this defined data path. From the NFS Server's perspective, this is a brand new request. A side effect of this protocol defined processing sequence is that when using an Exports file, both the original path name and this symbolic link defined path name must be exported, as previously described.

By contrast, NFSv2 and NFSv3 use the Mount protocol for mount requests which sends the entire path name to the NFS Server in a single request. In this case, after that path name has been export checked, the path name is passed to z/OS UNIX

System Services which resolves the entire path, including any symbolic link resolution, bypassing any separate export checking of the defined symbolic link path.

One exception to the above documented NFSv4 mount request processing is for the case when the symbolic link is the last qualifier in the path and it is followed by any processing attributes. In this case, the z/OS NFS server cannot follow the NFSv4 defined process because it would cause the processing attributes to get lost since non-z/OS NFS Clients do not understand the concept of z/OS NFS processing attributes and thus would not append them to the new lookup path. Therefore, if the z/OS NFS Server detects a lookup operation for a symbolic link which is followed by processing attributes, instead of returning the fact that this node is a symbolic link to the NFS Client, the z/OS NFS Server resolves the mount path in a similar manner to that used for NFSv2 and NFSv3 mount requests. The one difference between the NFSv4 and the NFSv2 and NFSv3 behavior is that NFSv4 performs export checking of the defined symbolic link path, while NFSv2 and NFSv3 do not.

UNIX look and feel

Using the z/OS NFS with z/OS UNIX managed files provides UNIX client users with a transparent view of their data. The file attributes are maintained in the same way as is found on any UNIX system.

- Regular, directory, link, device, and FIFO file types
- User, group, and other read/write/execute access permissions
- UID and GID file ownership
- File size

To access z/OS UNIX files, it is necessary to be defined to RACF as an z/OS UNIX user. Some installations might prefer to provide users with unrestricted access to their z/OS UNIX data by specifying **security(none)** or **security(exports)** in the site attributes. With this setting, the client's user ID and group ID credentials are used for all file access authentication, and there is no requirement for the user to be defined to RACF or to perform the mvslogin command.

Note: For the **security(none)** and **security(exports)** options, the UID of the root (UID=0 from the workstation) is mapped to UID of NOBODY (UID=65534) by the z/OS NFS server. The implication is that the z/OS NFS server will use the mapped UID of 65534 for all z/OS UNIX file authorization checking. For example, file creation owner UID is set to 65534 in the z/OS UNIX file attribute.

NFS file system attributes

For z/OS UNIX files, see Table 84 on page 458 and Table 85 on page 458 for the file system values that are returned for NFS attributes.

Displaying and modifying remote file system access control lists

POSIX provides limited file security management granularity. File access security can only be controlled via the permission bits as specified for the user, group and other classes. Some UNIX platforms have introduced additional security granularity by adding access control list (ACL) support to provide security specification on an individual user and/or group basis. An ACL is simply a list that specifies which users and groups get access to a file with what type of permission. The precise characteristics of this ACL support are platform specific.

The NFS Version 4 protocol provides the ability to remotely manage ACLs by providing the ability to display and modify ACL values with the ACL attribute. The NFS v4 protocol has provided a very rich ACL definition with granularity beyond that provided by many platform ACL implementations. Therefore, it is necessary to map between the NFS ACL definition and the platform definition. The key is to ensure that in this mapping process, the mapping should err in the direction of more restricted access, not less. When the NFS server sets an ACL it must be set at least as secure as specified by the NFS request. When an NFS server sends an ACL to an NFS client, the client must not perceive the file as more secure than it really is. For details on the NFS version 4 ACL definition, see the NFS version 4 protocol.

For POSIX permission bits, and some flavors of ACLs, a single entry specifies whether permission is being granted or denied for the target user or group to access the file. By contrast, NFS version 4 ACLs have two types of ACLs: “Allow” and “Deny”. An “Allow” indicates that the target user or group is being given the specified permission to access the file while a “Deny” indicates that the target user or group is explicitly being denied the specified permission to access the file.

Further, the ordering of a POSIX ACE (access control entry - an entry in the ACL) differs from that of an NFS Version 4 ACE. POSIX has a defined ordering as follows: owner, supplemental users, owning group, supplemental groups, and other. This is a kernel maintained ordering and can not be changed by the user. NFS version 4 ACEs do not have a rigid order. It is defined by the order of the entries in the ACL as created by the user. If an ACL conflict arises because of differences between the two ordering algorithms, then the POSIX rules will apply, since the ACL access authority is determined by the underlying z/OS UNIX system, not by the z/OS NFS server.

NFS can display and modify remote file system access control lists, provided that the function is supported by the remote NFS server. This support is limited to z/OS UNIX access control lists, as described in *z/OS UNIX System Services Planning*, GA22-7800, under Using Access Control Lists (ACLs). Access control list checking is controlled by the underlying file systems on the server systems, not by z/OS NFS server or client.

Remote ACL management restrictions:

Due to the differences in ACL implementations on the various platforms, the following restrictions must be applied when attempting to remotely manage ACLs to/from z/OS.

Note: At this time only the z/OS, AIX and Sun platforms are supported.

AIX restrictions

- AIX supports two kinds of ACLs: AIX ACLs and NFS ACLs. Only the NFS ACLs can be remotely managed from z/OS.
- The AIX Client requires a mount option on the mount command to enable ACLs on the mount:

```
mount -o vers=4,acl maxi:/nfs-authsys/user11_acltest2 /nfs-authsys/user11_acltest
```

- AIX only supports NFS ACLs on JFS2 extended attributes v2 file systems.

z/OS client restrictions

- Because z/OS Unix does not have a unique Base Mask ACL entry:
  - The z/OS NFS Client uses the Base Group ACL entry permission value for calculating the Mask ACL entries sent to NFS Servers.
  - The z/OS NFS Client fails 'get attribute' requests containing Mask ACL entries which do not match the Base Group ACL entry. The failure returns an ENODEV error.
  - Unlike Sun, the z/OS NFS Client set both the Group and Mask ACL entries when a 'chmod' is issued for a file/directory which has Extended ACLs. If only the Group or Mask ACL entry were changed, it would no longer be possible to display or change the ACLs for this object since the Group and Mask entries would be different.
- Because z/OS Unix does not have unique Default Base ACL entries:
  - The z/OS NFS Client uses the Access Base ACL entries for calculating the Default Base ACL entries sent to NFS Servers (that is, the Access and Default Base ACL entries will always match).
  - The z/OS NFS Client fails get attribute requests containing Default Base ACL entries that do not match the Access Base ACL entries. The failure returns an ENODEV error.
- Because non-z/OS platforms do not have separate File and Directory Default ACLs:
  - The z/OS NFS Client uses the File Default ACL entries for the NFS Default ACL entries sent to non-z/OS NFS Servers. The entries are used as both File and Directory Defaults on the non-z/OS platforms.

    The z/OS NFS Client checks any Dir Default ACL set requests to verify that they match the existing NFS Default ACL entries. If the two lists do not precisely match (same entries with same permissions), the set request fails with an ENODEV error. This requires that the File Default ACL entries must be set before the Directory Default ACL entries.
  - Ideally, Directory Default ACL entry set requests should not be issued in this case.
  - The z/OS NFS Client uses the NFS Default ACL entries sent by a non-z/OS Server for both File and Directory Default ACL get requests.
- The z/OS NFS Client supports the File and Directory Default ACL entries as unique entries when communicating with a z/OS NFS Server. There are no request order or permission value restrictions for this case.

z/OS server restrictions

- Because z/OS Unix does not have a unique Base Mask ACL entry:
  - The z/OS NFS Server uses the Base Group ACL entry permission value for calculating the Mask ACL entries sent to NFS Clients.
  - The z/OS NFS Server fails set attribute requests containing Mask ACL entries that do not match the Base Group ACL entry. The failure returns an NFS4ERR_ATTRNOTSUPP error.
- Because z/OS Unix does not have unique Default Base ACL entries:
  - The z/OS NFS Server uses the Access Base ACL entries for calculating the Default Base ACL entries sent to NFS Clients (that is, the Access and Default Base ACL entries will always match).
  - The z/OS NFS Server fails set attribute requests containing Default Base ACL entries that do not match the Access Base ACL entries. The failure returns an NFS4ERR_ATTRNOTSUPP error.

  - Since z/OS Unix only has Default Extended ACLs, Default Base ACLs are returned by the z/OS Server if Default Extended ACLs are also present.
- Because non-z/OS platforms do not have separate File and Directory Default ACLs:
  - The z/OS NFS Server uses the File Default ACL entries for the NFS Default ACL entries sent to non-z/OS NFS Clients. The entries are used as both File and Directory Defaults on the non-z/OS platforms.

    The z/OS NFS Server checks all existing z/OS Unix Directory Default ACL entries to verify that they match the z/OS Unix File Default ACL entries. If the two lists do not precisely match (same entries with same permissions), the get request fails with an NFS4ERR_ATTRNOTSUPP error.
  - The z/OS NFS Server uses the NFS Default ACL entries sent by a non-z/OS Client to set both the File and Directory Default ACL entries.
- The z/OS NFS Server supports the File and Directory Default ACL entries as unique entries when communicating with a z/OS NFS Client. There are no ACL entry restrictions in this case.

Chapter 5. z/OS NFS file locking and access control

This topic provides an overview of the z/OS NFS locking and access control functions.

Locking in NFS versions 2 and 3 (NFS server only)

This topic provides an overview of the z/OS NFS locking and access control functions provided by the Network Lock Manager (z/OS NFS NLM) and the z/OS NFS Network Status Monitor (z/OS NFS NSM). It explains how they work together to provide file locking and access control capability over z/OS NFS. In addition, this topic also explains the following features:

- Monitored lock
- Non-monitored locks
- Locking files
- Locking records

In NFS versions 2 and 3, the locking of a file on the z/OS NFS server is managed by Network Lock Manager (NLM) and Network Status Monitor (NSM). NLM and NSM are integrated into the z/OS NFS server to facilitate the expanded locking and serialization functions. Separate procedures for starting and stopping NLM and NSM were replaced in z/OS V1R7 by the server site attribute **nlm|nonlm**, which specifies their startup along with the NFS server. This integration also coordinates the locking function with stale file handle processing; when a file handle becomes stale, not only will the code clean up the file related data as it does in prior releases, but it will also release any locks that remain held for that file.

Using Network Lock Manager (NLM) in NFS V2 and V3

In NFS version 2 and version 3, the z/OS NFS NLM allows a client on the host to lock range of bytes or an entire file on the z/OS NFS server. The two types of locks that the client host uses are monitored locks and non-monitored locks.

The z/OS NFS NLM supports only advisory locking. Advisory locking is when the operating system keeps track of which files have been locked by which process, but does not prevent a process from writing to a file that is locked by another process. This means that a process can ignore an advisory lock if the process has adequate permission.

Monitored locks

Monitored locks provide the client user with reliability. If the server host on which the monitored locks are established fails, the locks are reinstated when the server host recovers. The locks that are held by the client host are discarded by the z/OS NFS NLM on the server host if the client host fails before the locks are released. Monitored locks will only work correctly if both the server host and the client host are running NSM.

Non-monitored locks

Non-monitored locks are used on personal computer (PC) operating systems. Non-monitored locks provide the same functionality as the monitored locks with one exception. If the server host on which the locks are established, fails and

© Copyright IBM Corp. 1991, 2010

recovers, the locks will not be re-established. The client host is responsible for detecting a server host failure and re-establishing the locks. In addition, the client host informs the z/OS NFS NLM when it has rebooted so that the client host can discard all of the locks and file shares held for the client.

Specifying a grace period for reclaiming locks

You can specify a time limit, or grace period, for clients to reclaim NFS V4 or NLM locks and share reservations when the z/OS NFS server restarts after a failure. To set this time limit, use the **leasetime** site attribute. For details, see "Site attributes syntax" on page 134.

During the reclaim grace period after a z/OS NFS server restart, the grace period may be extended after an open or lock reclaim event. If the **leasetime** site attribute is greater than 1200 seconds (20 minutes), the grace period will not be extended at a reclaim event; if the **leasetime**site attribute is less than 20 minutes, the grace period will be extended to one lease time after the reclaim event, up to but not exceeding 20 minutes after the z/OS NFS server completed its restart.

Note: In order for z/OS NFS client to reclaim locks from an AIX NFS server, AIX must be at Technology level 5 service pack 1 (5300-05-01).

Listing locks held for a file

To diagnose possible problems with conflicting locks, z/OS operators can issue a **listlock** command that displays all client programs and users which hold a lock on a file. The output messages include client and user id, the lock ranges held, and lock status. The **listlock** command can be used for MVS data sets, PDS or PDSE members, or z/OS UNIX files. For more information, see "Listlock operand" on page 234.

You can use the listlock command to find locking information in cases where a lock is unavailable and the blocker is managed by another NFS server address space running on the z/OS system. To determine the identity of the blocker in this case, the listlock command should be issued on the system which owns the lock.

Releasing locks held for a file

To release all locks for a file, z/OS operators can issue a **release** command that releases locks for z/OS UNIX files and MVS data sets or members. The command also forces the NFS server to release the file, and if the file is active, to close and deallocate it. For more information, see "Release operand" on page 230.

Note: The NLM protocol does not provide any means for NFS server to notify the NFS client that the locks were released. Thus, the client may proceed under the false assumption that it still has the locks. Therefore, the release command should only be used in extreme circumstances.

Using Network Status Monitor (NSM) in NFS V2 and V3

In NFS version 2 and version 3, the z/OS NFS NSM is a service that provides applications with information on the status of network host. Each z/OS NFS NSM keeps track of its own "state" and notifies any interested parties of a change in its state.

For correct operation of the z/OS NFS NSM, the client and the server hosts are required to monitor each other. When a lock request is issued by a process running on the client host, the NLM on the client host requests the NSM on the client host to monitor the server host. The client NLM then transmits the lock request to the z/OS NFS NLM on the server. On recept of the lock request the z/OS NFS NLM on the server host will request the z/OS NFS NSM on the server host to monitor the client host. In this way, each host is monitored by the NSM on the other host.

Locking in NFS version 4

With the NFS version 4 protocol, the support for file locking is part of the NFS protocol. The file locking support is structured so that an RPC callback mechanism is not required. This is a departure from the previous versions of the NFS file locking protocol, Network Lock Manager (NLM). The state associated with file locks is maintained at the server under a lease-based model. The server defines a single lease period for all states held by an NFS client. If the client does not renew its lease within the defined period, all state associated with the client's lease may be released by the server. The client may renew its lease with use of the RENEW operation or implicitly by use of other operations (primarily READ).

With the NFS version 4 protocol, a client user can choose to lock the entire file, or a byte range within a file. z/OS NFS client file locking requests can be managed with the llock(Y|N) parameter on the mount command or as an installation default.

z/OS NFS supports only advisory locking. Advisory locking is when the operating system keeps track of which files have been locked by which process, but does not prevent a process from writing to a file that is locked by another process. This means that a process can ignore an advisory lock if the process has adequate permission.

Byte-range locking

Byte-range locking is used to serialize activity to a range of bytes within a file. Byte-range locking is an advisory locking mechanism; that is, it does not prevent access to any application, but provides a mechanism for applications to communicate cooperatively through obtaining locks and querying if a lock is held.

Share reservations

Share reservations are a new concept in the NFS version 4 protocol and are different than byte-range locking. Share reservations provide a method for controlling access to a file. When an OPEN request is sent for a file, the requester can indicate the type of access that should be denied to other requesters attempting to access the same file, which is NONE, READ, WRITE or BOTH.

Share reservations have an advantage over byte-range locking in that they provide a mandatory locking interface; any application that attempts to OPEN a file that is already opened and locked by another application is denied access. However, in these cases, access to files (and application processing) may be slowed if files are not shared.

Specifying a grace period for reclaiming locks

You can specify a time limit, or grace period, for clients to hold an exclusive lock on a z/OS NFS server resource. This time limit, or lease time, determines how long a client can hold a lock against a conflicting lock request. Once the time limit is reached, if the client has not extended the lease time, the lock may be revoked, and the lock will be revoked if NLM receives a conflicting request for the lock. This lease time also applies to the grace period that NFS clients have to reclaim NFS V4 locks and share reservations when the z/OS NFS server restarts after a failure. It also dictates the amount of time that a client ID can remain active on the server without communicating. To set this time limit, use the **leasetime** site attribute. For details, see “Site attributes syntax” on page 134.

During the reclaim grace period after a z/OS NFS server restart, the grace period may be extended after an open or lock reclaim event. If the **leasetime** site attribute is greater than 1200 seconds (20 minutes), the grace period will not be extended at a reclaim event; if the **leasetime**site attribute is less than 20 minutes, the grace period will be extended by one lease time after the reclaim event, up to but not exceeding 20 minutes after the z/OS NFS server completed its restart.

Note: For the z/OS NFS client to reclaim locks from an AIX NFS server, AIX must be at Technology level 5 service pack 1 (5300-05-01).

Listing locks held for a file

To diagnose possible problems with conflicting locks, z/OS operators can issue a **listlock** command that displays all client programs and users which hold a lock on a file. The output messages include client and user id, the lock ranges held, and lock status. The **listlock** command can be used for MVS data sets, PDS or PDSE members, or z/OS UNIX files. For more information, see “Listlock operand” on page 234.

You can use the listlock command to find locking information in cases where a lock is unavailable and the blocker is managed by another NFS server address space running on the z/OS system. To determine the identity of the blocker in this case, the listlock command should be issued on the system which owns the lock.

The listlock command also lists file delegations. For more information, see “Delegating management of a file's resources to an NFS client” on page 45.

Releasing locks held for a file

To release all locks for a file, z/OS operators can issue a **release** command that releases locks for z/OS UNIX files and MVS data sets or members. The command also forces the NFS server to release the file, and if the file is active, to close and deallocate it. For more information, see “Release operand” on page 230.

Chapter 6. Commands and examples for AIX and UNIX clients

This topic gives the syntax and examples of commands that AIX users need to know to access MVS data sets from a client. Some examples are also provided for UNIX and Sun Solaris environments. This topic shows how to perform the following tasks.

- Log on to z/OS from your client
- Access MVS data sets from your client
- Display default mount point attributes
- Query mount points
- Unmount MVS data sets from the client
- Log out of z/OS.

The **mount** and **umount** commands are operating system specific commands. They are not shipped with z/OS NFS. See your NFS client documentation for the exact syntax and usage.

Using commands on AIX

Figure 4 shows a summary of the syntax for the commands described in this topic:

The **mvslogin** command is used to log in to z/OS from your workstation. The mvslogin command can be issued multiple times, and the last one overrides the previous one.

Note: When the z/OS NFS server site attributes hfssec, mvssec, or pubsec specify any of the Kerberos security flavors (krb5, krb5i, or krb5p):

- An MVS login is no longer required when using RPCSEC_GSS. The RACF authentication is done automatically based on your Kerberos principal.

The following is the mvslogin command syntax.

```
►►──mvslogin──┬────────────────────┬──┬────────────┬──┬─────────────┬──────────►
              ├─-p─────────────────┤  └─-g──group──┘  └─-norpcbind──┘
              ├─-n─────────────────┤
              ├─-pn────────────────┤
              ├─-P── mvs_passwd────┤
              └─-P── 'passsphrase'─┘

►──┬──────────────┬──hostname──┬────────────┬──────────────────────────────►◄
   └─-a──account──┘            └─mvs_userid─┘
```

Figure 4. Syntax for commands

where

Operand **Description**

-p Causes a prompt for your z/OS password. The password is passed to z/OS to validate the user logging in. Your security procedures determine whether you should use this parameter.

-n Causes a prompt for a new password.

-pn Causes a prompt for the user's current password and then causes two prompts for the user's new password.

-P *mvs_passwd* No prompt for your z/OS password; just type your z/OS password after the -P. This enables you to automate your z/OS login.

-P *'passphrase'* No prompt for your z/OS password phrase, a text string of 14 to 100 characters. This enables you to automate your z/OS login. See *z/OS Security Server RACF Command Language Reference* for more information on the z/OS password phrase and its syntax rules.

-g *group* A group name string passed to z/OS for accounting purposes. The maximum length is 8 characters.

-norpcbind Specifies that mvslogin should not look for the RPCBIND protocol on the NFS server system. The default is that mvslogin will first look for the RPCBIND protocol. If that request fails, or times out, it will then look for the PORTMAPPER. If this keyword is used, mvslogin will immediately use the PORTMAPPER protocol. Using this keyword, when it is known that the NFS server system does not support RPCBIND, can improve the performance of mvslogin, because it does not look for RPCBIND first. This keyword has no effect if the client system is not enabled for IP version 6 (IPv6).

-a *account* An account string passed to z/OS for accounting purposes. The maximum length is 16 characters.

hostname The name of the z/OS host (for example, `mvshost1`).

mvs_userid A user ID that z/OS recognizes as valid. If you do not specify this parameter, your workstation user name is used. The z/OS NFS server does not support the use of an alias user ID or a mixed case user ID with the mvslogin command.

The **mount** command is used to make a connection between a mount point on your local file system and one or more files in the z/OS file system.

The following is the **mount** command syntax.

```
►►──mount──┬───────────────┬──hostname:"/prefix/mvs_qual─────────────────►
           └──-o clnt_opt──┘

►──┬─────────────┬──"──/localpath─────────────────────────────────────────►◄
   └─,attribute──┘
```

where

Operand **Description**

-o *clnt_opt* The client mount command options (such as `soft,timeo=20`). Refer to the documentation of your client operating system for a description of the options for your client environment.

hostname The name of the z/OS host (for example, `mvshost1`).

/prefix An optional explicit prefix for selecting the z/OS UNIX file system

type (the HFSPREFIX( ) site attribute value), or for selecting the MVS file system type (the MVSPREFIX( ) site attribute value). If no prefix is specified, then the implicit prefix heuristic specified in the IMPPREFIX site attribute is used for determining the file system type.

mvs_qual The path name of a z/OS UNIX directory or an MVS high-level qualifier for accessing z/OS MVS data sets.

attribute A z/OS NFS server data set creation or file processing attribute (such as `text`). See Chapter 9, “Initialization attributes for the z/OS NFS server,” on page 115. If you specify any attributes, make sure you enclose *mvs_qual* and the attributes in double quotation marks.

/localpath The mount point on your client system (for example, `/u/smith/mnt`). This should be an empty directory.

The **showattr** command is used to display the default attributes or the attributes that have been set for a specific mount point. If you specify a mount point, showattr shows the attributes for the mount point, including the overriding values. For descriptions of the attributes, see Chapter 9, “Initialization attributes for the z/OS NFS server,” on page 115.

The following is the **showattr** command syntax.

```
►►──showattr──┬────┬──┬─────────────┬──hostname──┬──────────────┬──────────────►◄
              └─-t─┘  └─-norpcbind──┘            └─/localpath───┘
```

where

Operand **Description**

-t Used to specify tersed output.

-norpcbind Specifies that showattr should not look for the RPCBIND protocol on the NFS server system. The default is that showattr will first look for the RPCBIND protocol. If that request fails, or times out, it will then look for the PORTMAPPER. If this keyword is used, showattr will immediately use the PORTMAPPER protocol. Using this keyword, when it is known that the NFS server system does not support RPCBIND, can improve the performance of showattr, because it does not look for RPCBIND first. This keyword has no effect if the client system is not enabled for IP version 6 (IPv6).

hostname The name of the z/OS host (for example, `mvshost1`).

/localpath The mount point on your client system (for example, `/u/smith/mnt`). This should be an empty directory.

The **umount** command is used to break the connection between the mount point on your client and the server. When you issue this client command, the file you were editing is released (written to DASD). You do not need to unmount after each session, unmount only when you no longer have a need to access the z/OS file system. Check the documentation for your client operating system to ensure that you enter the umount command correctly.

The following is the **umount** command syntax.

```
►►──umount──/localpath──────────────────────────────────────────────►◄
```

where

Operand **Description**

/localpath The mount point on your client system (for example, `/u/smith/mnt`). This should be an empty directory.

The **mvslogout** command is used to disconnect from the remote z/OS NFS server host. The mvslogout command is only required when the mvslogin command was used to begin the connection.

The following is the mvslogout command syntax.

```
►►──mvslogout──┬─────────────┬──hostname──────────────────────────────►◄
               └─-norpcbind──┘
```

where

Operand **Description**

-norpcbind Specifies that mvslogout should not look for the RPCBIND protocol on the NFS server system. The default is that mvslogout will first look for the RPCBIND protocol. If that request fails, or times out, it will then look for the PORTMAPPER. If this keyword is used, mvslogout will immediately use the PORTMAPPER protocol. Using this keyword, when it is known that the NFS server system does not support RPCBIND, can improve the performance of mvslogout, because it does not look for RPCBIND first. This keyword has no effect if the client system is not enabled for IP version 6 (IPv6).

hostname The name of the z/OS host (for example, `mvshost1`).

Quick reference of AIX and UNIX commands

The following information is an example of a standard z/OS login and logout procedure for AIX.

```
# mvslogin mvshost1 smith
Password required
GFSA973I Enter MVS password: password
GFSA955I smith logged in ok.
# mount mvshost1:"/mvs/smith" /u/smith/mnt
mount: mvshost1:"/mvs/smith"
"/mvs/smith" was attached successfully.
# umount /u/smith/mnt
Unmounting '/u/smith/mnt' ...    successful
# mvslogout mvshost1
UID 200 logged out ok.
```

In this example:

Operand	Description
smith	Specifies a z/OS user ID and high level qualifier for MVS data sets.
mvshost1	Specifies the system name of the z/OS host.
/mvs/smith	Specifies the MVS prefix followed by the name of the high -level qualifier of the MVS data sets.
/u/smith/mnt	Specifies the name of the mount point.
GFSA*nnnt*	Messages starting with GFSA apply towards z/OS NFS requests. These messages are explained in Chapter 19, “Network File System messages,” on page 319.

You can use the **mount** command with no parameters specified to list the mount points on your client system.

Accessing z/OS UNIX file systems and z/OS conventional MVS files

To access z/OS UNIX files or z/OS conventional MVS data sets, enter both the **mvslogin** command to log in to the z/OS host system and the **mount** command to mount the files or data sets to your local system. The mvslogin command is only required when accessing data on systems where the z/OS NFS server site security attribute is set to *saf* or *safexp*. Once the files or data sets are mounted to a local directory, you can read, write, create, delete, and treat the mounted files as part of your workstation's local file system. When you are finished with your work, use the **umount** and **mvslogout** commands to break the connection. The mvslogout command is only required when the mvslogin command was used to begin the connection.

z/OS UNIX file systems and MVS data sets are very different and require different management techniques. Prior to z/OS V1R11, the NFS server distinguished between the two by the fact that z/OS UNIX file system paths are prefixed with an hfs prefix value (/hfs/pathname) and MVS data sets are not, The hfs prefix is not actually part of the path name. It is only intended as a trigger to tell the z/OS NFS Server that the specified path is a z/OS UNIX path, not an MVS data set. Based on the presence, or absence, of a prefix, the NFS Server invokes its appropriate data management functions.

Starting in V1R11, the z/OS NFS file system type management function has been expanded by adding an mvs prefix and a customer-configurable path resolution heuristic. The new mvs prefix provides the capability for the customer to explicitly specify a prefix for identifying MVS data sets, like the hfs prefix does for z/OS Unix files. The customer-configurable heuristic allows the customer to specify how to interpret absolute path names that do not have a prefix specified.

Note: If the underlying z/OS UNIX file system structure should change due to the mount of a new HFS or zFS file system into the space accessible by an existing remote NFS Client mount, this change and new directory structure and contents will not be visible to the remote NFS Client until the NFS mount is unmounted and remounted or the z/OS NFS Server is restarted.

To access files on z/OS systems where the z/OS NFS server site security attribute is set to *saf*, *exports*, or *safexp*, you need a z/OS user ID and password, and authorization to access the files that you need. You can only establish one z/OS session for each z/OS user ID. If you do not already have a z/OS user ID, a z/OS password, and access authorization, request them from your z/OS system administrator.

Note: If you cannot use the mvslogin, mvslogout, or **showattr** commands, they might be installed incorrectly or in another directory. Ensure that your system administrator has made the executable code for these three commands available to your workstation and that you have been given the correct path name to find the commands. Also, make sure that your version of these commands matches the release of the z/OS NFS that you are using. Otherwise, the commands might not function properly.

Mvslogin command examples

Use the **mvslogin** command to log in to z/OS from your workstation. The mvslogin command can be issued multiple times and the last one overrides the previous ones. The mvslogin command is only required when accessing data on systems where the z/OS NFS server site ***security*** attribute is set to *saf* or *safexp*.

Table 8 shows examples of the mvslogin command where `mvshost1` is the name of the z/OS host and `smith` is the user's ID on z/OS.

Table 8. Examples of the mvslogin command for clients

mvslogin -p mvshost1 smith
mvslogin -P smithspw -g finance -a 5278 mvshost1 smith
mvslogin -P "Smith's password phrase" mvshost1 smith
mvslogin -n mvshost1 smith
mvslogin -pn -a 5278 mvshost1 smith
mvslogin mvshost1
mvslogin mvshost1 smith

In the example where the user enters mvslogin mvshost1, the current login client user ID is used as the z/OS user ID.

In the example where the user enters mvslogin mvshost1 smith, the system then prompts for Smith's z/OS password. If Smith logs in successfully, this message appears:

```
GFSA955I smith logged in ok.
```

Otherwise, an appropriate error message appears.

Note: Messages with the prefix of GFSA and GFSC apply to NFS requests. These messages are further explained in Chapter 19, "Network File System messages," on page 319.

In the example where the user enters mvslogin -P "Smith's password phrase" mvshost1 smith, double-quotation marks are required around the password phrase because of the apostrophe within the phrase (smith's). Otherwise, only single-quotation marks are required around the password phrase. See *z/OS Security Server RACF Command Language Reference* for more information on syntax rules for password phrases.

When an z/OS UNIX UID or GID segment is defined with the user identification, an additional check is done to compare the z/OS UNIX UID or GID with the client UID or GID during the login processing. An informational message is returned when the server and the client UID or GID do not match. This informational message contains the z/OS UNIX UID and GID for the z/OS user identification.

Note: The authentication is considered successful even if the UID and GID do not match. The message is issued for the user's information only.

For the PCNFSD authentication request, the server UID and GID is returned to the client user if the UID and GID are defined. Otherwise, an arbitrary number is generated and returned to the client user.

“Permission denied” message

If you have successfully logged in and get the “Permission denied” message while trying to access data, that can be due to one of the following cases:

- An **mvslogout** command for the same z/OS host has been entered from the same client platform. See the description of **mvslogout** in “Using commands on the z/OS NFS client” on page 77 for details.
- Your z/OS user ID has been automatically logged out because the **logout** attribute value has been exceeded. This can happen when you leave the client idle for too long. Enter the **mvslogin** command again, and start your processes again. To find out how many seconds you can stay logged in while your client is idle, issue the **showattr** command and look at the logout attribute.
- Another mvslogin to the same z/OS host using the same z/OS ID has been entered from the same UID and the same client platform. If this is the case, retry your access.
- In multi-homed environments where a system has more than one network interface, the remote IP address specified in the mount command should match the remote IP address specified in the mvslogin and mvslogout. Note that a loopback IP address and the real IP address for the same system are considered separate IP addresses and therefore require the mount command and mvslogin/mvslogout to have matching IP addresses.
- The z/OS NFS server has been re-started. Enter the mvslogin command again, and start your processes again.

For more information, including additional causes of this message, see the explanation of the "Permission denied" message in “Messages from the client platform (AIX)” on page 412.

Note: Some clients give a somewhat different message such as “Access is Denied”.

Mount command examples using an MVS prefix

Use the **mount** command to make a connection between a mount point on your local file system and one or more files in the z/OS file system.

```
# mount mvshost1:"/mvs/smith" /u/smith/mnt
```

In this example:

Operand	**Description**
mvshost1	Specifies the name of the host server.
/mvs	The MVS prefix.
/smith	Specifies the name of the high-level qualifier of the MVS data sets.

/u/smith/mnt
Specifies the name of the mount point (preferably an empty directory).

At the same time, you can specify attributes for the mount point, overriding the default attributes.

Overriding default attributes

To override the default attributes, specify different attributes with the **mount** command or in a file access or creation command (such as **vi** in AIX or UNIX).

There are two kinds of attributes that you can modify:

Data set creation attributes provide information about an MVS data set to the z/OS NFS server, such as:

- The type of data set
- How the data set is allocated

Note: Data set creation attributes do not apply to z/OS UNIX files.

Processing attributes provide information to the z/OS NFS server about how to handle the file. For example:

- How long the files remain open
- Whether the file contents are sent and received in text form, or in binary form to avoid ASCII/EBCDIC conversion

Use the **showattr** command to display the default data set creation and processing attributes. For descriptions of the attributes, see Chapter 9, “Initialization attributes for the z/OS NFS server,” on page 115.

Files are created and processed using the mount point attributes that were in effect when the files were last mounted. If your installation's default attributes have been changed (by way of the *exportfs* operand of the **modify** system operator command or restart of the server) and you want to apply these new default attributes, you can unmount and remount (using the **umount** and **mount** commands).

When you access the file with a data access or creation command, you can override some of the attributes that were set by a mount command or the server default settings.

Mount command example: overriding server default attributes without an MVS prefix

In this example, the **mount** command is used to modify two processing attributes, specifying **binary** (rather than **text**), and **readtimeout(30)** (rather than the server default **readtimeout** value):

```
# mount mvshost1:"smith,binary,readtimeout(30)" /u/smith/mnt
```

In this example:

Operand	Description
mvshost1	Specifies the name of the host server.
smith	Specifies the name of the high-level qualifier of the MVS data sets or z/OS UNIX directory (depending on the IMPPREFIX( ) site attribute setting).

binary
: Specifies the processing mode for file contents sent between z/OS and the client (binary mode avoids ASCII/EBCDIC conversion).

readtimeout(30)
: Specifies the amount of time (30 seconds) allowed since the last read access before the file is closed.

/u/smith/mnt
: Specifies the name of the mount point (preferably an empty directory).

At the same time, you can specify the MVS prefix for the mount point.

Note: AIX 6.1 NFSv4 clients now require that all remote mount points must include a leading slash.

Getting authorization to access files

If the mount fails, check with your system administrator to ensure that you are authorized to access the MVS data sets or z/OS UNIX files and that the data sets or files are listed in the exports data set. The privilege level required to enter **mount** and **umount** commands varies among client operating system implementations. Many UNIX implementations limit these commands to the root user or superuser mode. If the MVS system operator issues the *freeze=on* operand of the z/OS NFS server **modify** command, all new tries to mount an MVS or z/OS UNIX file system fail until the z/OS system operator issues the *freeze=off* operand. If the z/OS system operator issues the *freeze=onhfs* operand of the z/OS NFS server **modify** command, z/OS conventional MVS data sets can still be mounted, but all new tries to mount z/OS UNIX file systems fail until the system operator issues the *freeze=offhfs* operand.

Saving of mount points

Once the **mount** command is issued successfully and a mount point is established between a local directory and the MVS or z/OS UNIX file system, the mount point information is saved in the mount handle data set by the z/OS NFS server. This information is used to automatically reestablish active mount points when the server is started. When accessing z/OS conventional MVS data sets, a mount point is active if the last activity against this mount point is less than the **restimeout** attribute value set by the system administrator.

The **mount** command does not need to be reissued for the same mount point in further sessions unless the **umount** command has been used to disconnect the mount point or the mount point has been disconnected by the cleanup activity of the **restimeout** site attribute. For more information about the **restimeout** site attribute see Chapter 9, “Initialization attributes for the z/OS NFS server,” on page 115.

Automatic timed logout - logout attribute

If there is no activity on the client within the period specified in the **logout** attribute of the attributes file, or the server stops, the connection between the server and the client workstation is logged out automatically. You must issue the **mvslogin** command again to get access to the z/OS files.

Displaying default and mount point attributes - showattr

Use the **showattr** command to display the default attributes or the attributes that have been set for a specific mount point. If you specify a mount point, showattr shows the attributes for the mount point, including the overriding values. For descriptions of the attributes, see Chapter 9, “Initialization attributes for the z/OS NFS server,” on page 115.

If you omit the hostname, you must specify the `/localpath`.

Table 9 shows examples of the showattr command for some clients.

Table 9. Examples of the showattr command for clients

Client Environments	Command Examples
AIX, UNIX	`showattr mvshost1 /u/smith/mnt`
z/OS	`showattr mvshost1 /u/smith/mnt`
Solaris	`showattr mvshost1 /u/smith/mnt`
Linux	`showattr mvshost1 /u/smith/mnt`

Make sure that your version of the showattr command matches the release of the z/OS NFS that you are using. Otherwise, the z/OS NFS server attributes do not display.

These examples show different ways you can use the **showattr** and **mount** commands.

Figure 5 shows a showattr command with just the host name (`mvshost1` in this example) specified. The attributes for the server are displayed.

```
# showattr  mvshost1

GFSA988I Remote host does not have AF_INET6 interface.

FMID HDZ1C1N , last APAR HDZ1C1N, last changed module: GFSARXDR
Compiled at May  5 2010 09:15:39

z/OS Network File System Server Data Set Creation Attributes:

lrecl(8196)             recfm(vb)               blksize(0)
space(100,10)           blks                    dsorg(ps)
dir(27)                 unit()                  volume()
recordsize(512,4K)      keys(64,0)              nonspanned
shareoptions(1,3)       mgmtclas()              dsntype(pds)
norlse                  dataclas()              storclas()

z/OS Network File System Server Processing Attributes:

binary                  lf                      blankstrip
nofastfilesize          retrieve                maplower
mapleaddot              executebitoff           setownerroot
attrtimeout(120)        readtimeout(90)         writetimeout(30,120)
sync                    nofileextmap            xlat()
srv_ccsid(1047)         cln_ccsid(819)          notag
convserv(lre)           nordrverf               sidefile()

z/OS Network File System Server Site Attributes:

mintimeout(1)           nomaxtimeout            logout(1800)
nfstasks(8,16,8,4,4)    restimeout(48,0)
hfsprefix(/hfs)         mvsprefix(/mvs)         impprefix(mvs)
bufhigh(32M, 80%)       readaheadmax(16K)       cachewindow(112)
percentsteal(20)        maxrdforszleft(32)      logicalcache(16M)
smf(none,off)           nopcnfsd                security(safexp,safexp,safexp)
leadswitch              sfmax(0)                nochecklist
fn_delimiter(,)         readdirtimeout(30)      hfsfbtimeout(60)
upcase                  rec878                  mintasks(4,4,1)
noremount               fileidsize(64)          denyrw
nonlm                   nodhcp                  nostringprep
leasetime(120)          nodelegation            DlyDTimeout(10)
setgid(posix)
nfsv4domain()
public()                        mvssec(sys,krb5,krb5i,krb5p)
hfssec(sys,krb5,krb5i,krb5p)    pubsec(sys,krb5,krb5i,krb5p)
```

Figure 5. Displaying server attributes

If you use the terse (-t) option, the attributes display like this:

```
# showattr  -t  mvshost1

GFSA988I Remote host does not have AF_INET6 interface.
lrecl(8196),recfm(vb),blksize(0),space(100,10),blks,dsorg(ps),dir(27),unit(),
volume(),recordsize(512,4K),keys(64,0),nonspanned,shareoptions(1,3),mgmtclas(),
dsntype(pds),norlse,dataclas(),storclas()
binary,lf,blankstrip,nofastfilesize,retrieve,maplower,mapleaddot,executebitoff,
setownerroot,attrtimeout(120),readtimeout(90),writetimeout(30,120),sync,
nofileextmap,xlat(),srv_ccsid(1047),cln_ccsid(819),notag,convserv(lre),nordrverf,
sidefile()
mintimeout(1),nomaxtimeout,logout(1800),nfstasks(8,16,8,4,4),restimeout(48,0),
hfsprefix(/hfs),mvsprefix(/mvs),impprefix(mvs),bufhigh(32M,80%),
readaheadmax(16K),cachewindow(112),percentsteal(20),maxrdforszleft(32),
logicalcache(16M),smf(none,off),nopcnfsd,security(safexp,safexp,safexp),
leadswitch,sfmax(0),nochecklist,fn_delimiter(,),readdirtimeout(30),
hfsfbtimeout(60),upcase,rec878,mintasks(4,4,1),noremount,fileidsize(64),denyrw,
nonlm,nodhcp,nostringprep,leasetime(120),nodelegation,DlyDTimeout(10),
setgid(posix),nfsv4domain(),public(),mvssec(sys,krb5,krb5i,krb5p),
hfssec(sys,krb5,krb5i,krb5p),pubsec(sys,krb5,krb5i,krb5p)
hfssec(sys,krb5,krb5i,krb5p),pubsec(sys,krb5,krb5i,krb5p)
```

Figure 6 illustrates the **showattr** command being used to display the attributes for the z/OS host named `mvshost1` as well as the mount point, `/u/smith/mnt`.

Figure 6 also illustrates the specified options. In the second showattr command, the client user specifies `/u/smith/mnt` in addition to `mvshost1`. This shows the user's specified settings at that mount point, rather than the settings in the attributes data set.

```
# mount mvshost1:"smith,text,space(5,0),trks"  /u/smith/mnt

# showattr  mvshost1

GFSA988I Remote host does not have AF_INET6 interface.

FMID HDZ1C1N , last APAR HDZ1C1N, last changed module: GFSARXDR
Compiled at May  5 2010 09:15:39

z/OS Network File System Server Data Set Creation Attributes:

lrecl(8196)             recfm(vb)               blksize(0)
space(100,10)           blks                    dsorg(ps)
dir(27)                 unit()                  volume()
recordsize(512,4K)      keys(64,0)              nonspanned
shareoptions(1,3)       mgmtclas()              dsntype(pds)
norlse                  dataclas()              storclas()

z/OS Network File System Server Processing Attributes:

binary                  lf                      blankstrip
nofastfilesize          retrieve                maplower
mapleaddot              executebitoff           setownerroot
attrtimeout(120)        readtimeout(90)         writetimeout(30,120)
sync                    nofileextmap            xlat()
srv_ccsid(1047)         cln_ccsid(819)          notag
convserv(lre)           nordrverf               sidefile()


z/OS Network File System Server Site Attributes:

mintimeout(1)           nomaxtimeout            logout(1800)
nfstasks(8,16,8,4,4)    restimeout(48,0)
hfsprefix(/hfs)         mvsprefix(/mvs)         impprefix(mvs)
bufhigh(32M, 80%)       readaheadmax(16K)       cachewindow(112)
percentsteal(20)        maxrdforszleft(32)      logicalcache(16M)
smf(none,off)           nopcnfsd                security(safexp,safexp,safexp)
leadswitch              sfmax(0)                nochecklist
fn_delimiter(,)         readdirtimeout(30)      hfsfbtimeout(60)
upcase                  rec878                  mintasks(4,4,1)
noremount               fileidsize(64)          denyrw
nonlm                   nodhcp                  nostringprep
leasetime(120)          nodelegation            DlyDTimeout(10)
setgid(posix)
nfsv4domain()
public()                        mvssec(sys,krb5,krb5i,krb5p)
hfssec(sys,krb5,krb5i,krb5p)    pubsec(sys,krb5,krb5i,krb5p)
```

Figure 6. Displaying mount point attributes (Part 1 of 2). The client user changed the space(100,10), blks, and binary attributes to space(5,0), trks, and text for the mount point `/u/smith/mnt`, and then specified that mount point in the second showattr command.

```
# showattr  mvshost1     /u/smith/mnt

GFSA988I Remote host does not have AF_INET6 interface.
server = mvshost1, serverbuf = mvshost1

FFMID HDZ1C1N , last APAR HDZ1C1N, last changed module: GFSARXDR
Compiled at May  5 2010 09:15:39

z/OS Network File System Server Data Set Creation Attributes:

lrecl(8196)             recfm(vb)               blksize(0)
space(100,10)           blks                    dsorg(ps)
dir(27)                 unit()                  volume()
recordsize(512,4K)      keys(64,0)              nonspanned
shareoptions(1,3)       mgmtclas()              dsntype(pds)
norlse                  dataclas()              storclas()


z/OS Network File System Server Processing Attributes:

binary                  lf                      blankstrip
nofastfilesize          retrieve                maplower
mapleaddot              executebitoff           setownerroot
attrtimeout(120)        readtimeout(90)         writetimeout(30,120)
sync                    nofileextmap            xlat()
srv_ccsid(1047)         cln_ccsid(819)          notag
convserv(lre)           nordrverf               znfsclient
sidefile()

NFSv4 mount point; may be unmounted.


z/OS Network File System Server Site Attributes:

mintimeout(1)           nomaxtimeout            logout(1800)
nfstasks(8,16,8,4,4)    restimeout(48,0)
hfsprefix(/hfs)         mvsprefix(/mvs)         impprefix(mvs)
bufhigh(32M, 80%)       readaheadmax(16K)       cachewindow(112)
percentsteal(20)        maxrdforszleft(32)      logicalcache(16M)
smf(none,off)           nopcnfsd                security(safexp,safexp,safexp)
leadswitch              sfmax(0)                nochecklist
fn_delimiter(,)         readdirtimeout(30)      hfsfbtimeout(60)
upcase                  rec878                  mintasks(4,4,1)
noremount               fileidsize(64)          denyrw
nonlm                   nodhcp                  nostringprep
leasetime(120)          nodelegation            DlyDTimeout(10)
setgid(posix)
nfsv4domain()
public()                        mvssec(sys,krb5,krb5i,krb5p)
hfssec(sys,krb5,krb5i,krb5p)    pubsec(sys,krb5,krb5i,krb5p)
```

Figure 6. Displaying mount point attributes (Part 2 of 2). The client user changed the space(100,10), blks, and binary attributes to space(5,0), trks, and text for the mount point `/u/smith/mnt`, and then specified that mount point in the second showattr command.

Unmounting and logging out of z/OS

This section describes the **umount** and **mvslogout** commands.

Disconnecting your mount point - umount

Use the **umount** command to break the connection between the mount point on your client and the server. When you issue this client command, the file you were editing is released (written to DASD). You do not need to unmount after each session, unmount only when you no longer have a need to access the MVS file system. Check the documentation for your client operating system to ensure that you enter the umount command correctly.

Table 10 shows examples of the umount command for some clients.

Table 10. Examples of the umount command for clients

Client Environments	Umount Command Examples
AIX, UNIX	umount /u/smith/mnt
Solaris	umount /u/smith/mnt
Windows	nfs unlink z: (where z: is the NFS mounted drive)
Linux	umount /u/smith/mnt

In this example:

Operand	Description
u/smith/mnt	Specifies the mount point on the local file system.
mvshost1	Specifies the name of the z/OS host system.

For example, suppose that you want to unmount from the server, and the mount point on your workstation is named `/u/smith/mnt`. You could enter the **umount** command as follows:

```
# umount /u/smith/mnt
```

"No Such File or Directory" Message - The z/OS system operator can also unmount your workstation from the server. If this happens before you try to unmount, you get a "No such file or directory" error message.

Ending your z/OS session - mvslogout

Use the **mvslogout** command to disconnect from the z/OS host. The mvslogout command is only required when the **mvslogin** command was used to begin the connection.

An mvslogout to an z/OS user ID cancels a prior mvslogin to the same z/OS user ID from the same local host.

Your account is automatically logged out if it is inactive for the period of time specified in the **logout** site attribute.

Table 11 shows an example of the mvslogout command for some clients, in which the name of the z/OS host is `mvshost1`.

Table 11. Example of the mvslogout command for clients

Client Environments	Mvslogout Command Examples
AIX, UNIX	mvslogout mvshost1
Solaris	mvslogout mvshost1
Linux	mvslogout mvshost1

If you log out successfully, a message like this appears:

```
GFSA958I uid 215 logged out ok.
```

Chapter 7. Commands and examples for z/OS NFS clients

This topic gives the syntax and examples of commands that NFS users need to know to access AIX, UNIX, , and other remote files using the NFS client. This topic shows how to perform the following tasks.

- Log on to a remote z/OS system from the NFS client if the target server is a remote NFS server
- Access NFS files from the NFS client
- Display NFS client statistical data
- Query mount points
- Display default mount point attributes
- Mount and unmount remote file systems from the NFS client
- Log out of z/OS, if the target server was a remote NFS server

The command programs are intended to run in an shell environment and are not implemented as TSO/E commands, with the exception of **mount** and **unmount**.

The **mount** and **unmount** commands are not part of NFS. See *z/OS UNIX System Services Command Reference* for additional details. An example of the syntax and usage is shown here for your information. You can use the TSO HELP MOUNT and TSO HELP UNMOUNT commands to see the syntax that is applicable to your system.

For information about the NFS attributes see Chapter 8, "Initialization attributes for the z/OS NFS client," on page 103 and Chapter 9, "Initialization attributes for the z/OS NFS server," on page 115.

Using commands on the z/OS NFS client

The following is a summary of the syntax for the commands described in this topic. See "Mount command syntax and examples" on page 85 and "Unmount command syntax and examples" on page 92 for information about the **mount** and **unmount** commands.

The **mvslogin** command is used to log in to z/OS from your workstation. The mvslogin command can be issued multiple times, and the last one overrides the previous one. The mvslogin command is required only when accessing data on systems where the z/OS NFS server site **security** attribute is set to *saf* or *safexp*.

Note: When the z/OS NFS server site attributes hfssec, mvssec, or pubsec specify any of the Kerberos security flavors (krb5, krb5i, or krb5p):

- Before initiating any RPCSEC_GSS request, a client user issuing mvslogin with a RACF user id whose Kerberos principal is different from the Kerberos principal in the client's credential cache (issue klist to view) should:
  1. Issue kdestroy to destroy the existing Kerberos credentials and
  2. Issue kinit to acquire the new Kerberos credentials for the principal that corresponds to the RACF user id that is being logged into.

© Copyright IBM Corp. 1991, 2010

On some platforms, the NFS clients cache the credentials and a kdestroy does not immediately cause the Kerberos credentials to be discarded. Refer to the documentation for your specific client platform for more information.

The following is the mvslogin command syntax.

```
►►──mvslogin──┬────────────────────┬──┬────────────┬──┬─────────────┬────────────►
              ├─-p─────────────────┤  └─-g──group──┘  └─norpcbind───┘
              ├─-n─────────────────┤
              ├─-pn────────────────┤
              ├─-P──mvs_passwd─────┤
              └─-P──'passphrase'───┘

►──┬──────────────┬──hostname──┬────────────┬─────────────────────────────────►◄
   └─-a──account──┘            └─mvs_userid─┘
```

where

Operand **Description**

-p
Causes a prompt for the user's z/OS password. The password is passed to z/OS to validate the user logging in. Your security procedures determine whether you should use this parameter.

-n
Causes a prompt for a new password.

-pn
Causes a prompt for the user's current password and then causes two prompts for the user's new password.

-P *mvs_passwd*
No prompt for your z/OS password; just type your z/OS password after the -P. This enables you to automate your MVS login.

-P '*passphrase***'**
No prompt for your z/OS password phrase, a text string of 14 to 100 characters. This enables you to automate your MVS login. See *z/OS Security Server RACF Command Language Reference* for more information on the z/OS password phrase and its syntax rules.

-g *group*
A group name string passed to z/OS for accounting purposes. The maximum length is 8 characters.

-norpcbind
Specifies that mvslogin should not look for the RPCBIND protocol on the NFS server system. The default is that mvslogin will first look for the RPCBIND protocol. If that request fails, or times out, it will then look for the PORTMAPPER. If this keyword is used, mvslogin will immediately use the PORTMAPPER protocol. Using this keyword, when it is known that the NFS server system does not support RPCBIND, can improve the performance of mvslogin, because it does not look for RPCBIND first. This keyword has no effect if the client system is not enabled for IP version 6 (IPv6).

-a *account*
An account string passed to z/OS for accounting purposes. The maximum length is 16 characters.

hostname
The name of the z/OS host (for example, `mvshost1`). The default is the local host.

mvs_userid
A user ID that z/OS recognizes as valid. If you do not specify this

parameter, your workstation user name is used. The NFS server does not support the use of an alias user ID or a mixed case user ID with the mvslogin command.

The **showattr** command is used to display the default attributes or the attributes that have been set for a specific mount point. If you specify a mount point, showattr shows the attributes for the mount point, including the overriding values. For descriptions of the attributes, see Chapter 9, “Initialization attributes for the z/OS NFS server,” on page 115.

The following is the showattr command syntax.

```
►►──showattr──┬────┬──┬───────────┬──hostname──┬────────────┬──────────────►◄
              └─-t─┘  └─norpcbind─┘            └─/localpath─┘
```

where

Operand — **Description**

-t — Used to specify tersed output.

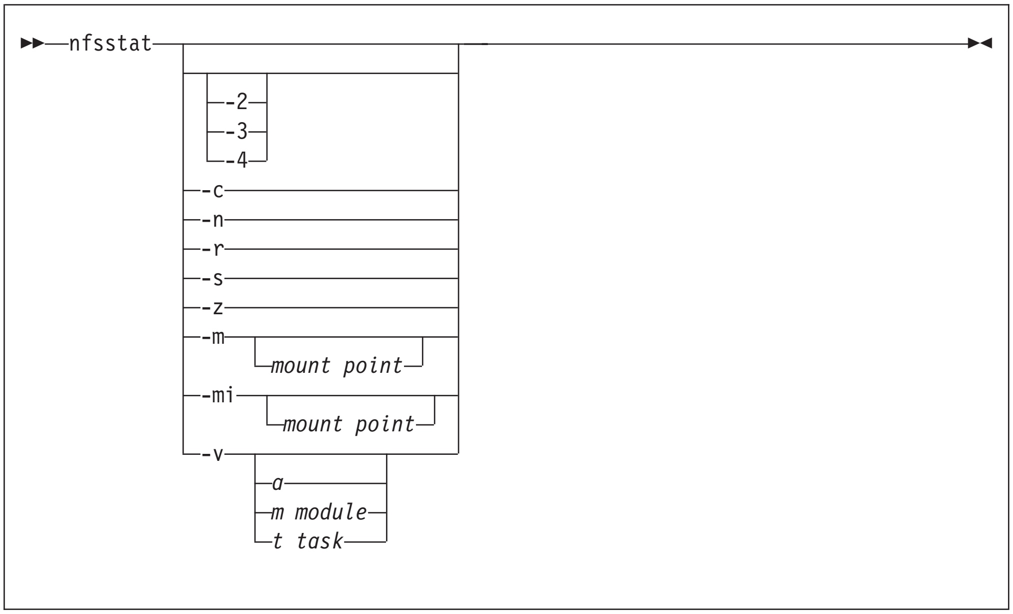
-norpcbind — Specifies that showattr should not look for the RPCBIND protocol on the NFS server system. The default is that showattr will first look for the RPCBIND protocol. If that request fails, or times out, it will then look for the PORTMAPPER. If this keyword is used, showattr will immediately use the PORTMAPPER protocol. Using this keyword, when it is known that the NFS server system does not support RPCBIND, can improve the performance of showattr, because it does not look for RPCBIND first. This keyword has no effect if the client system is not enabled for IP version 6 (IPv6).

hostname — The name of the z/OS host (for example, `mvshost1`). The default is the local host.

/localpath — The mount point on your client system (for example, `/u/smith/mnt`). This should be an empty directory.

The **nfsstat** command is used to display the NFS client and server statistical information, to reset the statistical information to zero, to display NFS mount point information, or to set the debug status.

The following is the nfsstat command syntax.



where

Operand **Description**

-2
Displays statistics for Version 2 of the z/OS NFS server and z/OS NFS client.

-3
Displays statistics for Version 3 of the z/OS NFS server and z/OS NFS client.

-4
Displays statistics for Version 4 of the z/OS NFS server and z/OS NFS client.

-c
Displays both NFS and RPC statistics about the NFS client. This is the default option on the nfsstat command.

-n
Displays NFS statistics about the NFS client and NFS servers.

- To display the NFS statistics of only the client, use -cn
- To display the NFS statistics of only the server(s), use -sn

-r
Displays RPC statistics about the NFS client and NFS servers.

- To display the RPC statistics of only the client, use -cr
- To display the RPC statistics of only the server(s), use -sr

-s
Displays statistics about the NFS servers.

-z
Initializes statistics to zero for the NFS client and NFS servers. Used by root user only. This option can be combined with options *-c*, *-n*, and *-r* on the nfsstat command. When combined with these nfsstat options, each particular set of statistics is set to zero after the statistics are printed.

- To only reset the client side statistics, use -cz
- To reset the server(s) side statistics , use -sn

-m
Displays the name of each NFS mounted file system.

-mi
Displays the name of each NFS mounted file system and checks the server's IP address validity.

-m *mount point*
Displays information for the NFS mounted file system on the specified mount point.

-mi *mount point*
Displays information for the NFS mounted file system on the specified mount point and checks the server's IP address validity.

-v
Returns information about the latest APAR installed on the z/OS NFS client.

-v a
Returns a list of all the modules in the z/OS NFS client with their current level information. At the end of the list is information about the latest APAR installed.

-v m *module*
Returns information about the APAR level of the specified *module*.

-v t *task*
Returns information about the APAR level of the specified *task*.

Any of the options specified above in the nfsstat command can be used in conjunction with each other to display the desired statistics. The following are some examples of such usage:

- To display the RPC and NFS statistics of the z/OS NFS client and the z/OS NFS server(s): `nfsstat` (that is, with no options, which is equivalent to `nfsstat –cs` or `–csrn` or `–rn` or `–csr234`)
- To display the RPC statistics of the z/OS NFS client and the z/OS NFS server(s): `nfsstat –csr`
- To display the NFS version 2 statistics of the z/OS NFS client: `nfststat –c2`
- To display the NFS version 3 statistics of the z/OS NFS Client and the z/OS NFS server(s): `nfsstat -cs3`
- To display the NFS version 4 statistics of the z/OS NFS server(s): `nfsstat -s4`

Note: The examples listed above are not a complete list, but a sampling of the combinational usage of the nfsstat command's options. As stated above, any of the nfsstat command's options can be combined with each other to achieve the desired level of filtering to display the desired statistics.

The **showmount** command is used to display the remote NFS server mount information. If you omit the options, the default option displays hostnames of all remote mounts from the *hostname* NFS server. If you omit the *hostname* parameter, then the local hostname is used.

The following is the showmount command syntax.

```
►►──showmount──┬──────┬──┬─────────────┬──hostname──────────────────────────►◄
               ├─-a───┤  └─norpcbind───┘
               ├─-d───┤
               └─-e───┘
```

If you omit the options, the default option displays hostnames of all remote mounts from the *hostname* NFS server.

where

Operand **Description**

-a Displays all mounts in the format hostname:directory from the hostname specified in the showmount command.

-d Displays only directory names of all mounts from the hostname specified in the showmount command.

-e Displays the list of exported directories from the hostname specified in the showmount command. Directory entries are shown as they appear in the exports data set.

-norpcbind Specifies that showmount should not look for the RPCBIND protocol on the NFS server system. The default is that showmount will first look for the RPCBIND protocol. If that request fails, or times out, it will then look for the PORTMAPPER. If this keyword is used, showmount will immediately use the PORTMAPPER protocol. Using this keyword, when it is known that the NFS server system does not support RPCBIND, can improve the performance of showmount, because it does not look for RPCBIND first. This keyword has no effect if the client system is not enabled for IP version 6 (IPv6).

hostname The name of the NFS server host (for example, `mvshost1`). The default is the local host.

The **crnl2nl** command will remove the carriage-return and end-of-file mark (EOF) from the input file. No other conversions are performed. Either, or both the input and output file can be a local or remote file.

The following is the crnl2nl command syntax.

```
►►──crnl2nl──input──output──────────────────────────────────────────────►◄
```

The data is assumed to be in EBCDIC format.

where

Operand	Description
input	Absolute path name of the input file to be converted.
output	Absolute path name of the output file.

The **nl2crnl** converts the line delimiter from MVS format to carriage-return newline (CRNL) format; the newline (NL) in the input file is converted to the carriage-return newline pairs (CR, NL). No other conversions are performed. Both input and/or output file can be local or remote file.

The following is the nl2crnl command syntax.

```
►►──nl2crnl──input──output──────────────────────────────────────────────►◄
```

where

Operand **Description**

input Absolute path name of the input file to be converted.

output Absolute path name of the output file.

Notes:

1. The size of the write buffer is double the size of the read buffer.
2. The data is assumed to be in EBCDIC format.

The **mvslogout** command is used to disconnect from the remote z/OS NFS server host. The mvslogout command is only required when the mvslogin command was used to begin the connection.

The following is the mvslogout command syntax.

```
►►──mvslogout──┬───────────┬──hostname──────────────────────────────────►◄
               └─norpcbind─┘
```

where

Operand **Description**

-norpcbind Specifies that mvslogout should not look for the RPCBIND protocol on the NFS server system. The default is that mvslogout will first look for the RPCBIND protocol. If that request fails, or times out, it will then look for the PORTMAPPER. If this keyword is used, mvslogout will immediately use the PORTMAPPER protocol. Using this keyword, when it is known that the NFS server system does not support RPCBIND, can improve the performance of mvslogout, because it does not look for RPCBIND first. This keyword has no effect if the client system is not enabled for IP version 6 (IPv6).

hostname The name of the z/OS host (for example, `mvshost1`). The default is the local host.

Accessing z/OS

To access remote z/OS files, enter both the **mvslogin** command to log in to the NFS server's host system and the **mount** command to mount the files or data sets to your local system. The mvslogin command is only required when accessing data on systems where the NFS server site *security* attribute is set to saf or safexp. Once the files or data sets are mounted to a local mount point, you can read, write, create, delete and treat the mounted files as part of your local UNIX files. When you are finished with your work, use the **unmount** and **mvslogout** commands to break the connection. The mvslogout command is only required when the mvslogin command was used to begin the connection.

Notes:

1. Issuing the mount and unmount commands, as well as creation of the mount point, must be performed by someone with superuser authority.
2. If the underlying z/OS UNIX file system structure should change due to the mount of a new HFS or zFS file system into the space accessible by an existing remote NFS Client mount, this change and new directory structure and contents will not be visible to the remote NFS Client until the NFS mount is unmounted and remounted or the z/OS NFS Server is restarted.

To access files on z/OS systems where the NFS server site *security* attribute is set to saf or safexp, you need a z/OS user ID and password, and authorization to access the files that you need. You can only establish one z/OS session for each z/OS user ID. If you do not already have an z/OS user ID, a z/OS password, and access authorization on the z/OS system from which you require NFS services, request them from your z/OS system administrator.

Note: If you cannot use the mvslogin, mvslogout, or **showattr** commands, they might be installed incorrectly. Ensure that your system administrator has made the executable code for these three commands available to your z/OS user ID and that you have been given the correct access authority to them.

Mvslogin command examples

Use the **mvslogin** command to log in to the remote z/OS system. The mvslogin command can be issued multiple times and the last one overrides the previous one. The mvslogin command is only required when accessing data on systems where the NFS server site *security* attribute is set to *saf* or *safexp*.

Table 12 shows examples of the mvslogin command where `mvshost1` is the name of the z/OS host and `smith` is the user's ID on z/OS.

Table 12. Examples of the mvslogin command for clients

Command Examples
mvslogin -p mvshost1 smith
mvslogin -P smithspw -g finance -a 5278 mvshost1 smith
mvslogin -P "Smith's password phrase" mvshost1 smith
mvslogin -n mvshost1 smith
mvslogin -pn -a 5278 mvshost1 smith
mvslogin mvshost1
mvslogin mvshost1 smith

In the example where the user enters **mvslogin mvshost1**, the current login client user ID is used as the z/OS user ID.

In the example where the user enters **mvslogin mvshost1 smith**, the system then prompts for Smith's z/OS password. If Smith logs in successfully, this message displays.

```
GFSA955I smith logged in ok.
```

Otherwise, an appropriate error message displays.

Note: Messages that start with GFSA and GFSC apply to NFS requests. See Chapter 19, "Network File System messages," on page 319.

"Permission denied" message

If you have successfully logged in and get the "Permission denied" message while trying to access data, that can be due to one of these cases:

- An **mvslogout** command for the same z/OS host has been entered from the same client platform. See mvslogout for details.
- Your z/OS user ID has been automatically logged out because the **logout** attribute value has been exceeded. This can happen when you leave the client idle for too long. Enter the **mvslogin** command again, and start your processes

again. To find out how many seconds you can stay logged in while your client is idle, issue the **showattr** command and look at the logout attribute.

- Another mvslogin to the same z/OS host using the same z/OS ID has been entered from the same UID and the same client platform. If this is the case, retry your access.

Mount command syntax and examples

Use the TSO MOUNT command to make a connection between a mount point on your local file system and one or more files on a remote AIX, UNIX, , or other file system. The MOUNT command can only be used by a z/OS superuser.

Note: The same mount function can also be performed using the UNIX automount facility or /etc/rc shell scripts support. When the automount facility is used to manage remote NFS mount points, the NFS client user could experience ESTALE/EIO errors if the automounter unmounts the accessed mount point when the time limits specified by the automount DURATION and DELAY parameters have been exceeded. The automount default, DURATION=NOLIMIT, disables the DURATION timeout period. The DELAY for unmounting file systems should be longer than the time NFS clients are likely to keep NFS mounts to the files and directories active. For more information see *z/OS UNIX System Services Command Reference*.

Figure 7 illustrates the syntax of the TSO MOUNT command. For more information about the **mount** command, see *z/OS UNIX System Services Command Reference*.

```
MOUNT FILESYSTEM(file_system_name)
      TYPE(NFS)
      MOUNTPOINT(local_mountpoint)
      MODE(RDWR|READ)
      PARM('hostname:"path_name,server_attributes", client_options')
      SETUID|NOSETUID
      TAG(TEXT,CCSID)
      WAIT|NOWAIT
```

Figure 7. TSO MOUNT command syntax operands

where

FILESYSTEM(*file_system_name*)
Specifies the name of the file system to be added to the file system hierarchy. This operand is required. The file system name specified must be unique among previously mounted file systems. It may be an arbitrary name up to 44 characters in length of a filesystem. You can enclose file_system_name in single quotes, but they are not required.

TYPE(*NFS*)
Specifies the type of file system that performs the logical mount request. The NFS parameter must be used.

MOUNTPOINT(*local_mountpoint*)
Specifies the path name of the mount point directory, the place within the file hierarchy where the file system is to be mounted. The local_mountpoint specifies the mount point path name. The name can be a relative path name or an absolute path name. The relative path name is relative to the current working directory of the TSO/E session (usually the home directory).

Therefore, you should usually specify an absolute path name. A path name is case-sensitive, so enter it exactly as it is to appear. Enclose the path name in single quotes.

Note: The mount point must be a directory. Any files in that directory are inaccessible while the file system is mounted. Only one file system can be mounted to a mount point at any time.

MODE(*RDWR*|*READ*)
Specifies the type of access for which the file system is to be opened.

RDWR specifies that the file system is to be mounted for read and write access. *RDWR* is the default option.

READ specifies that the file system is to be mounted for read-only access.

PARM(*'hostname:"path_name,server_attributes", client_attributes'*)
Specifies the hostname of the remote NFS server, the path name of the UNIX file or MVS data set to be mounted, the server attributes, and the client attributes.

See Chapter 9, "Initialization attributes for the z/OS NFS server," on page 115 for descriptions of the server attributes, including the **mvsmnt** attribute to be used with mount commands using the NFS version 4 protocol. The "client_attributes" used on the PARM parameter are the same as the NFS client attributes, although some attributes cannot be used on the mount. See Table 15 on page 111 for a list of the parameters that can be used to specify client attributes on the MOUNT command.

If the path_name has no colon and if no server attributes are specified then the double quotes can be omitted. Enclose the entire string in single quotes. If the automount facility is being used, the single quotes should not be specified. You can specify lower case or upper case characters.

SETUID|**NOSETUID**
See *z/OS UNIX System Services Command Reference* for details about **SETUID** and **NOSETUID**.

TAG(*TEXT,CCSID*)
See *z/OS UNIX System Services Command Reference* for details about the **TAG** keyword.

Note: When TAG is specified, **xlat(Y)** must not be specified; otherwise, mount will fail.

WAIT|**NOWAIT**
See *z/OS UNIX System Services Command Reference* for details on **WAIT** and **NOWAIT**.

Data conversion

The NFS client supports data conversion defined by the universal character encoding standard known as the Unicode Standard on V1R2 (and above) when reading data from a remote NFS server or writing data to a remote NFS server. The Unicode Standard offers character conversion as well as basic case conversion. Within character conversion, characters are converted from one coded character set identifier (CCSID) to another. CCSID information is obtained from the *cln_ccsid* and *srv_ccsid* parameters.

Only single byte to single byte data conversion is supported. For example, if a client file has a CCSID of 437 and a server file has a CCSID of 297, data conversion will occur between USA ASCII format (CCSID 437) and French EBCDIC format

(CCSID 297). Single byte to multiple byte conversion (including double byte character set (DBCS)) is not supported and will result in NFS terminating with an error message. NFS version 4 protocol (NFSv4) differs from NFSv2 and NFSv3 protocol in handling single to multiple byte conversion. Therefore, the technique-search-order specified in the **convserv()** attribute should consider the effects of the NFS protocol being used. See“Creating the conversion environment for Unicode Services” on page 164 for further details.

The **cln_ccsid**, **srv_ccsid**, **xlat** , **tag/notag**, and **convserv** attributes identify whether data conversion is performed, and how data conversion is done. The **cln_ccsid**, **srv_ccsid**, **xlat** , **tag/notag**, and **convserv** attributes are supported by the z/OS NFS client installation parameter and TSO MOUNT command. The parameters on a TSO MOUNT command override the parameters specified as a NFS client installation parameter.

BSAM, QSAM, and VSAM ESDS access to remote files

BSAM, QSAM, and VSAM ESDS applications can access files stored on remote NFS servers through the NFS client. This will allow existing MVS application programs access to data on other systems using BSAM, QSAM, and VSAM ESDS interfaces. The BSAM, QSAM, and VSAM ESDS access methods assume that all text files are EBCDIC. When using these access methods, the **delim** parameter indicates whether the remote files contain text or binary data. Text data consists of records that are separated by a delimiter. If the delim parameter is not binary, the EBCDIC text delimiter is used by the access methods when processing the remote files. The **delim** parameter is supported on the NFS client installation parameter and TSO MOUNT command.

Note: The z/OS NFS client can also access VSAM key-sequenced (KSDS) and relative record (RRDS) data sets.

All the remote file objects under the same mount point have the same **delim** value. The **delim** parameter cannot be set on a file basis under the mount point. The **delim** parameter in the TSO MOUNT command overrides the **delim** parameter specified in NFS client installation parameter. However, you can override the **delim** parameter on the TSO MOUNT command with the **filedata** parameter on a JCL DD statement, SVC 99, or TSO ALLOCATE command. The **filedata** parameter can be either **text** or **binary**.

For BSAM, QSAM, and VSAM ESDS applications accessing files stored on remote NFS servers, the NFS client provides data conversion when the *xlat=Y* parameter is specified. It uses the cln_ccsid and srv_ccsid settings. When *xlat=N*, the NFS client will not perform data conversion. The filedata parameter on a JCL DD statement is also used to specify if the data consists of text records separated by delimiters or if the data is binary and does not contain record delimiters. To avoid undesirable data conversions, care should be taken to insure the specification of the xlat and delim parameters are not in conflict with the data type specified by the filedata parameter on a JCL DD statement. The filedata and delim parameters only affect BSAM, QSAM, and VSAM ESDS data access and have no affect on the NFS client data conversion. The NFS client data conversion is only controlled by the xlat parameter. The significance of different filedata and delim combinations are described in the following information.

Note: In each case, ensure the NFS client *xlat=Y*, *cln_ccsid*, and *srv_ccsid* parameter settings are correct for the filedata and delim combination.

FILEDATA=TEXT, *delim*=notBINARY
Specifies that the data is to be accessed as text. The access method appends a

record delimiter on output and expects delimiters on input. The delimiter used is that specified on the ***delim*** parameter.

FILEDATA=TEXT, *delim*=BINARY
Specifies that the data is to be accessed as text. The access method appends a record delimiter on output and expects delimiters on input. The delimiter used is the default of the EBCDIC new line character (x'15') since the delim parameter does not specify a valid text delimiter.

FILEDATA=BINARY, *delim*=notBINARY
Specifies that the data is to be accessed as binary. The access method does not append record delimiters on output, does not recognize record delimiters on input, and it treats all characters as data on input.

FILEDATA=BINARY, *delim*=BINARY
Specifies that the data is to be accessed as binary. The access method does not append record delimiters on output, does not recognize record delimiters on input, and it treats all characters as data on input.

FILEDATA=not specified, *delim*=specified
Means that the data is to be accessed according to the value specified in the delim parameter.

FILEDATA=not specified, *delim*=not specified
Means that the data is to be accessed as binary. The access method does not append record delimiters on output, does not recognize record delimiters on input, and it treats all characters as data on input.

The NFS client also provides UNIX authentication for security and provides the UNIX client's user ID (UID), group ID (GID), and a list of users GIDs to the remote NFS server for authorization checking. When the remote NFS server is the NFS server, the **mvslogin** command can be used to provide additional security checking through RACF authentication. MVS application programs which require access to data on remote systems may be required to perform an mvslogin.

For information on BSAM, QSAM, and VSAM ESDS applications access to z/OS UNIX or remote files and their restrictions, refer to *z/OS DFSMS Using Data Sets*.

Invoking the Mount command on the z/OS platform

Use the TSO MOUNT command to make a connection between a local mount point on the NFS client and an NFS server.

The Mount command can be invoked on the z/OS platform from different locations. Assume the following values are to be used for the mount point:

NFS	file system type	indicates that this file system is to mounted as an NFS file system
mynfs	file system	is the name of the file system to be mounted.
tcpj701:	server host name	is the NFS Server host name
nfstest	exported path	is the name of the directory to be mounted.
text,mvsmnt	server attributes	are the z/OS NFS Server mount attributes to be used for this mount point. **Note:** Other NFS servers, such as AIX, Linux, and SUN, do not have server attributes.
,soft,timeo(0),xlat(y)	client attributes	are the z/OS NFS Client attributes to be used for this mount point
/nfstest	mountpoint	is the directory in the local client file system where the file system is to be mounted.

The different styles of mount command are:

1. **From TSO.** The mount command can be issued from the TSO Ready prompt or from the ISPF TSO commands prompt. (This is the version of the command that is shown in the examples in "Additional mount command examples.")

   ```
   mount type(NFS) filesystem(mynfs) mountpoint('/nfstest')
     parm('tcpj701:"nfstest,text,mvsmnt",soft,timeo(100),xlat(y)')
   ```

2. **From the OMVS shell.** The mount command can also be invoked from within the OMVS shell or a shell script. This version of the command looks more like it would be when issued on other Unix platforms:

   ```
   /usr/sbin/mount -tnfs -fmynfs -w0
            -o'tcpj701:"nfstest,text,mvsmnt",soft,timeo(100),xlat(y)' /nfstest
   ```

3. **TSO command from OMVS shell.** The mount command can also be invoked as a TSO command from within the OMVS shell or a shell script. In this case, the command looks like the TSO version of the command:

   ```
   tso -t "mount type(NFS) filesystem(mynfs) mountpoint('/nfstest')
                parm('tcpj701:\"nfstest,text,mvsmnt\",soft,timeo(100),xlat(y)')"
   ```

   Note: The inner double quote must be entered with an escape character (\"). If only a double quote is entered , the double quote will be stripped by the shell TSO command parser, causing the z/OS NFS server mount attributes to be misinterpreted as z/OS NFS client attributes, which will lead to unpredictable results.

Additional mount command examples

In this example, the **mount** command is used to mount a set of MVS files. The PARM operand contains the NFS server **text** processing attribute and requires the use of double quotes around the string user,text.

```
mount filesystem(nfs00) type(nfs) mountpoint('/u/nfsdir')
parm('stlmvs3:"user,text",soft,timeo(100)')
```

In this example:

Operand	**Description**
nfs00	Specifies the name of the file system to be added to the file system hierarchy.
nfs	Specifies the required file system type.
/u/nfsdir	Specifies the name of the mount point (preferably an empty directory).
stlmvs3	Specifies the name of the host NFS server.
user	Specifies the name of the high-level qualifier of the MVS files on the NFS server. **Note:** Based on the implicit prefix heuristic specified with the IMPPREFIX( ) site attribute, you may need to specify the MVS path type prefix to identify the file system being mounted as an MVS data set path. For example: parm('stlmvs3:"/mvs/user,text",soft,timeo(100)')
text	Specifies the processing attribute for the NFS server.
soft	Specifies the PARM operand option for the NFS client.
timeo(100)	Specifies the PARM operand option for the NFS client.

In this example, the mount command is used to mount a z/OS UNIX directory. The PARM operand contains the NFS server *binary* processing attribute and requires the use of double quotes around the string */hfs/u/user,binary*.

```
mount filesystem(nfs01) type(nfs) mountpoint('/u/nfsdir1')
parm('stlmvs3:"/hfs/u/user,binary",soft')
```

In this example:

Operand	**Description**
nfs01	Specifies the name of the file system to be added to the file system hierarchy.
nfs	Specifies the required file system type.
/u/nfsdir1	Specifies the name of the mount point (preferably an empty directory).
stlmvs3	Specifies the name of the host NFS server.
/hfs/u/user	Specifies the name of the z/OS UNIX directory on the NFS server. The '/hfs' prefix is optional based on the implicit prefix selection setting.
binary	Specifies the processing attribute for the NFS server.
soft	Specifies the PARM operand option for the NFS client.

In this example, the mount command is used to mount an AIX home directory.

```
mount filesystem(nfs02) type(nfs) mountpoint('/u/nfsdir2')
parm('aix6000:/home/user,xlat(y)')
```

In this example:

Operand	**Description**
nfs02	Specifies the name of the file system to be added to the file system hierarchy.
nfs	Specifies the required file system type.
/u/nfsdir2	Specifies the name of the mount point (preferably an empty directory).
aix6000	Specifies the name of the host AIX NFS server.
/home/user	Specifies the name of the home directory on the AIX NFS server.
xlat(y)	Specifies the PARM operand option for the NFS client.

In this example, the mount command is used to mount an AIX home directory using the NFS version 4 protocol.

```
mount filesystem(nfs03) type(nfs) mountpoint('/u/nfsdir2')
parm('aix6000:/home/user,xlat(y),vers(4)')
```

In this example:

Operand	**Description**
nfs03	Specifies the name of the file system to be added to the file system hierarchy.
nfs	Specifies the required file system type.

/u/nfsdir2 Specifies the name of the mount point (preferably an empty directory).

aix6000 Specifies the name of the host AIX NFS server.

/home/user Specifies the name of the home directory on the AIX NFS server.

xlat(y) Specifies the PARM operand option for the NFS client.

vers(4) Specifies the version of NFS protocol that is being used.

In this example, the mount command is used to mount a Windows Share using the NFS version 4 protocol.

```
 mount filesystem(nfs04) type(nfs) mountpoint('/u/nfsdir4')
     parm('windowshost:"D:/",xlat(y),vers(4)')
```

In this example:

Operand **Description**

nfs04 Specifies the name of the file system to be added to the file system hierarchy.

nfs Specifies the required file system type.

/u/nfsdir4 Specifies the name of the mount point (preferably an empty directory).

windowshost Specifies the hostname of the Windows NFS server.

D:/ Specifies the name of the share on the Windows NFS server.

xlat(y) Specifies the PARM operand option for the NFS client.

vers(4) Specifies the version of NFS protocol that is being used.

Automount facility mount: In this example, the mount command in the automount policy file is used to do ASCII to EBCDIC conversion.

```
name  *
type  NFS
filesystem shared.<asis_name>
mode  rdwr
duration 15
delay 10
parm  mvshost<asis_name>",convserv(LE)",xlat(Y)
setuid no
```

In this example:

Operand **Description**

type NFS Identifies the automount file system type as NFS.

parm mvshost<asis_name>",convserv(LE)",xlat(Y)
Specifies the parameter information to be supplied to NFS. mvshost specifies the name of the NFS Server. /<asis_name> is the remote file system. convserv(LE) is a z/OS NFS server option convserv to conversion technique. The string should be enclosed in quotes. xlat(Y) is a z/OS NFS client option xlat to allow the client to perform ascii to ebcdic conversion. The string should not be enclosed in quotes.

For more information about the z/OS UNIX automount facility, including the other parameters in the automount policy, such as filesystem, mode, duration, delay, and setuid, see *z/OS UNIX System Services Planning* and *z/OS UNIX System Services Command Reference*.

Getting authorization to access files

If the mount fails, check with your system administrator to ensure that you are authorized to access the AIX or UNIX file system listed in the exports control file. The NFS client also provides UNIX authentication for security and provides the UNIX client's UID, GID, and a list of the GIDs from the UNIX client's groups to the remote NFS server for authorization checking. When the remote NFS server is the NFS server, the **mvslogin** command can be used to provide additional security checking through RACF authentication. The privilege level required to enter **mount** and **unmount** commands is superuser. When requesting service from the NFS server, if the z/OS system operator issues the *freeze=on* operand of the **modify** command, all new tries to mount a z/OS UNIX file system fail until the z/OS system operator issues the *freeze=off* operand. If the UNIX system operator issues the *freeze=onhfs* operand of the modify command, z/OS conventional MVS data sets can still be mounted, but all new tries to mount UNIX files fail until the system operator issues the *freeze=offhfs* operand.

Saving of mount points

Once the **mount** command is issued successfully and a mount point is established between a remote directory and the file system, the mount point information is not saved by the NFS client. The NFS client does not maintain mount persistence across restart. If UNIX or the NFS client is restarted, all prior session's mount point information is lost and all mount points must be reestablished.

Automatic timed logout - logout attribute

When using Network File System services from the NFS server, if there is no activity on the client within the period specified in the **logout** attribute of the attributes file, or if the server stops, the connection between the server and the client workstation is logged out automatically. You must issue the **mvslogin** command again to get access to the z/OS files.

Unmount command syntax and examples

This section describes the **unmount** command.

Disconnecting your mount point - unmount

Use the **unmount** command to break the connection between the mount point on your client and the server (that is, to unmount). You must have superuser authority to issue the unmount command.

Note: The same unmount function can also be performed using the UNIX **automount** facility. When the automount facility is used to manage remote NFS mount points, the NFS client user could experience `ESTALE/EIO` errors if the automounter unmounts the accessed mount point when the time limits specified by the automount ***duration*** and ***delay*** parameters have been exceeded. For additional information on the UNIX automount facility, see *z/OS UNIX System Services Planning* and *z/OS UNIX System Services Command Reference*.

Figure 8 on page 93 illustrates the syntax of the TSO UNMOUNT command. For more information about the UNMOUNT command, see *z/OS UNIX System Services Command Reference*.

```
UNMOUNT  FILESYSTEM(file_system_name)
NORMAL | DRAIN | IMMEDIATE | FORCE | RESET
```

Figure 8. TSO UNMOUNT command syntax operands

where

FILESYSTEM(*file_system_name*)
Specifies the name of the file system to be removed from the file system hierarchy. file_system_name specifies the file_system_name exactly as it was specified when the file system was originally mounted. You can enclose file_system_name in single quotes, but they are not required.

NORMAL|DRAIN|IMMEDIATE|FORCE|RESET
NORMAL: Specifies that if no user is accessing any of the files in the specified file system, the unmount request is processed. Otherwise, the system rejects the unmount request. **NORMAL** is the default option.

DRAIN: Specifies that the system is to wait until all uses of the file system have ended normally before the unmount request is processed or until another UNMOUNT command is issued.

Note: UNMOUNT can be specified with IMMEDIATE to override a previous unmount DRAIN request for a file system. If this is used in the foreground, your TSO/E session waits until the unmount request has completed. The attention request key (usually ATTN or PA1) will not end the command.

IMMEDIATE: Specifies that the system is to unmount the file system immediately. Any users accessing files in the specified system receive failing return codes. All data changes to files in the specified file system are saved. If the data changes to files cannot be saved, the unmount request fails.

Note: UNMOUNT of an NFS mount point (regardless of soft or hard mount option) with NORMAL, DRAIN, or IMMEDIATE may fail with the return code of EBUSY if the z/OS NFS client determines that there are ongoing NFS requests to the NFS server. The UNMOUNT receiving EBUSY can have the file system unmounted immediately with FORCE, at the risk of data loss.

FORCE: Specifies that the system is to unmount the file system immediately. Any users accessing files in the specified file system receive failing return codes. All data changes to files in the specified file are saved, if possible. If the data changes cannot be saved to the files, the unmount request continues and data is lost.

Note: You must issue an UNMOUNT IMMEDIATE request before issuing UNMOUNT FORCE. Otherwise, UNMOUNT FORCE fails.

RESET: A reset request stops a previous unmount DRAIN request.

The following example unmounts the file system NFSC_001 normally:

```
UNMOUNT FILESYSTEM('NFSC_001')
```

The following example forces an **unmount** of the file system NFSC_001. You must issue an UNMOUNT IMMEDIATE before you can issue an unmount *FORCE* command.

```
UNMOUNT FILESYSTEM('NFSC_001') IMMEDIATE
UNMOUNT FILESYSTEM('NFSC_001') FORCE
```

If you receive a **“No Such File or Directory”** message, the z/OS system operator can also unmount your workstation from the server. If this happens before you try to unmount, you get a “No such file or directory” error message.

Displaying client and server statistical information–nfsstat

Use the **nfsstat** command to display the NFS client and server statistical information, to reset the statistical information to zero, to display NFS mount point information, or to set the debug status.

The following nfsstat command displays the NFS and RPC statistics for the NFS client:

```
nfsstat -c
```

The following nfsstat command resets the NFS and RPC statistics for the NFS server and client to zero. Only the root user may use this option:

```
nfsstat -z
```

You may also use this command as follows:

- To only reset the client side statistics, use `nfsstat -cz`
- To only reset the z/OS NFS server(s) side statistics, use `nfsstat -sz`

The following nfsstat command displays the RPC statistics of both the NFS Client and the NFS server(s):

```
nfsstat -r
```

You may also use this command as follows:

- To only display the client side RPC statistics, use `nfsstat -cr`
- To only display the server(s) side RPC statistics , use `nfsstat -sr`

The following nfsstat command displays the NFS statistics of both the NFS Client and the NFS server(s):

```
nfsstat -n
```

You may also use this command as follows:

- To display the NFS statistics of only the client, use `nfsstat -cn`
- To display the NFS statistics of only the server(s), use `nfsstat -sn`

To display NFS version-specific information, use the following values with an nfsstat command option:

2 NFS version 2

3 NFS version 3

4 NFS version 4

For example, to display the NFS version 3 statistics of the z/OS NFS Client and the z/OS NFS server(s), enter:

```
nfsstat  -cs3
```

Figure 9 on page 95 displays both z/OS NFS Server and Client NFSv3 statistics, assuming that the LPAR has one z/OS NFS Server (with the StartUpProcedure name of MVSNFS) and one z/OS NFS Client.

```
$ /usr/lpp/NFS/nfsstat -3

GFSC857I z/OS Network File System Server : (MVSNFS)

Server NFSv3:
calls           badcalls
20019           0
null            getattr         setattr         lookup          access
0           0%  0           0%  753         4%  9123       45%  2424       12%
readlink        read            write           create          mkdir
208         1%  1058        5%  941         5%  488         2%  118         0%
symlink         mknod           remove          rmdir           rename
312         1%  0           0%  1214        6%  119         0%  725         4%
link            readdir         readdirplus     fsstat          fsinfo
412         2%  419         2%  0           0%  1508        7%  1           0%
pathconf        commit
1           0%  195         1%

GFSC857I z/OS Network File System Client :

Client NFSv3:
calls           badcalls
20035           0
null            getattr         setattr         lookup          access
0           0%  0           0%  754         4%  9124       45%  2427       12%
readlink        read            write           create          mkdir
208         1%  1058        5%  948         5%  488         2%  118         0%
symlink         mknod           remove          rmdir           rename
312         1%  0           0%  1214        6%  119         0%  725         4%
link            readdir         readdirplus     fsstat          fsinfo
412         2%  419         2%  0           0%  1509        7%  2           0%
pathconf        commit
2           0%  196         1%
Unicode Support service is used.
```

Figure 9. Displaying NFS Server and Client NFSv3 statistical information

Figure 10 shows the output from the nfsstat command using the -r option to display the remote procedure call (RPC) statistics for the NFS client.

```
USER1:/u/user1:> nfsstat -r
Client rpc:
calls           badcalls        retrans         timeout         qfull
107             0               0               0               0
lossconn
0

Unicode Support service is used.
```

Figure 10. Displaying NFS client rpc statistical information

In this example:

Operand **Description**

calls
Specifies the total number of RPC calls sent.

badcalls
Specifies the total of RPC calls rejected by a server.

retrans
Specifies the number of times an RPC call had to be retransmitted.

timeout
Specifies the number of times an RPC call timed out.

qfull
specifies the number of times an RPC call had to be delayed due to insufficient resources.

losscconn specifies the number of times an RPC call had to be retransmitted on a new TCPIP connection.

Figure 11 shows the output from the **nfsstat** command using the -nc option to display the NFS statistics for the NFS client.

```
USER1:/u/user1:> nfsstat -nc

Client NFSv2:
calls           badcalls
71              0
null            getattr         setattr         root            lookup
0           0%  1           1%  0           0%  0           0%  68         96%
readlink        read            writecache      write           create
0           0%  0           0%  0           0%  0           0%  0           0%
remove          rename          link            symlink         mkdir
0           0%  0           0%  0           0%  0           0%  0           0%
rmdir           readdir         fsstat
0           0%  1           1%  1           1%

Client NFSv3:
calls           badcalls
10              0
null            getattr         setattr         lookup          access
0           0%  0           0%  0           0%  0           0%  0           0%
readlink        read            write           create          mkdir
0           0%  0           0%  0           0%  0           0%  0           0%
symlink         mknod           remove          rmdir           rename
0           0%  0           0%  0           0%  0           0%  0           0%
link            readdir         readdirplus     fsstat          fsinfo
0           0%  0           0%  5          50%  3          30%  1          10%
pathconf        commit
1          10%  0           0%

Client NFSv4:
calls           badcalls
18              0
null            access          close           commit          create
0           0%  2           5%  0           0%  0           0%  0           0%
delegpurge      delegreturn     getattr         getfh           link
0           0%  0           0%  9          21%  4           9%  0           0%
lock            lockt           locku           lookup          lookupp
0           0%  0           0%  0           0%  4           9%  0           0%
nverify         open            openattr        open_cfm        downgrade
2           5%  0           0%  0           0%  0           0%  0           0%
putfh           putpubfh        putrootfh       read            readdir
12         28%  0           0%  2           5%  0           0%  4           9%
readlink        remove          rename          renew           restorefh
0           0%  0           0%  0           0%  0           0%  0           0%
savefh          secinfo         setattr         setclid         clid_cfm
0           0%  0           0%  0           0%  2           5%  2           5%
verify          write           rlse_lockowner
0           0%  0           0%  0           0%

Unicode Support service is used.
```

Figure 11. Displaying NFS client NFS statistical information

In this example:

Operand	**Description**
calls	Specifies the total number of NFS calls sent.
badcalls	Specifies the total of NFS calls rejected by a server.

Figure 12 on page 97 shows the output from **nfsstat** command using the **-s** option to display ALL of the z/OS NFS Server (RPC, NFSv2, NFSv3, NFSv4) statistical information.

```
# /usr/lpp/NFS/nfsstat -s

GFSC857I z/OS Network File System Server : (MVSNFS)
Server RPC:
calls           badcalls        nullrecv        badlen          xdrcalls
69130           0               0               0               0
dupreqs
0

Server NFSv2:
calls           badcalls
26678           0
null            getattr         setattr         root            lookup
0           0%  3519       13%  264         1%  0           0%  9616       36%
readlink        read            writecache      write           create
208         1%  4148       15%  0           0%  3647       14%  487         2%
remove          rename          link            symlink         mkdir
1212        4%  724         3%  412         1%  312         1%  118         0%
rmdir           readdir         fsstat
119         0%  388         1%  1504        6%

Server NFSv3:
calls           badcalls
20019           0
null            getattr         setattr         lookup          access
0           0%  0           0%  753         4%  9123       45%  2424       12%
readlink        read            write           create          mkdir
208         1%  1058        5%  941         5%  488         2%  118         0%
symlink         mknod           remove          rmdir           rename
312         1%  0           0%  1214        6%  119         0%  725         4%
link            readdir         readdirplus     fsstat          fsinfo
412         2%  419         2%  0           0%  1508        7%  1           0%
pathconf        commit
1           0%  195         1%

Server NFSv4:
calls           badcalls
22433           0
null            access          close           commit          create
0           0%  8109        8%  1039        1%  189         0%  430         0%
delegpurge      delegreturn     getattr         getfh           link
0           0%  0           0%  28423      30%  9559       10%  412         0%
lock            lockt           locku           lookup          lookupp
1028        1%  59          0%  1038        1%  7878        8%  145         0%
nverify         open            openattr        open_cfm        downgrade
2707        3%  1041        1%  0           0%  6           0%  0           0%
putfh           putpubfh        putrootfh       read            readdir
22843      24%  0           0%  1           0%  1138        1%  1686        2%
readlink        remove          rename          renew           restorefh
208         0%  1331        1%  724         1%  1           0%  1334        1%
savefh          secinfo         setattr         setclid         clid_cfm
2054        2%  0           0%  752         1%  1           0%  1           0%
verify          write           rlse_lockowner
0           0%  1180        1%  15          0%
cb_null         cb_compound     cb_getattr      cb_recall
0           0%  0           0%  0           0%  0           0%
```

Figure 12. Displaying all of the z/OS NFS Server (RPC, NFSv2, NFSv3, NFSv4) statistical information

Figure 13 on page 98 shows the output from the nfsstat command using the -m option to display the server and path name of each NFS mounted file system.

```
# nfsstat   -m

mvshost1:"/hfs/sj/sjpl" is mounted on /sj/sjpl/host1 filesystem NFS_MNT1
mvshost1:"/hfs/sj/sjpl2" is mounted on /sj/sjpl/host2 filesystem NFS_MNT2
```

Figure 13. Displaying NFS mounted file system information

Figure 14 shows the output from the nfsstat command using the -m option to display the server name, path name, and attributes of mount point */mnt* using the version 3 protocol with secure(upd).

```
# tso -t "mount type (NFS) filesystem(nfs1) mountpoint('/mnt') parm('mvshost1:/hfs/home/hain,secure(udp),vers(3)')"

mount type (NFS) filesystem(nfs1) mountpoint('/mnt') parm('mvshost1:/hfs/home/hain,secure(udp),vers(3)')

# nfsstat   -m  /mnt

server  mvshost1
path    /hfs/home/hain,secure(udp),vers(3)

hard            vers(3)         proto(udp)      secure(udp)
timeo(7)        retrans(3)      rpcbind(y)      accesschk(y)
delim(NA)       xlat(n)         cln_ccsid(1047) srv_ccsid(819)
convserv(LRE)   stringprep(n)   llock(y)
rsize(32768)    wsize(32768)    readahead(8)    delaywrite(16)
acregmin(3)     acregmax(60)    acdirmin(30)    acdirmax(60)
datacaching(y)  attrcaching(y)  retry(10)       dynamicsizeadj(y)
```

Figure 14. Displaying NFS mounted file system information with secure(upd) (Versions 2 and 3 protocol only)

Figure 15 shows the output from the nfsstat command using the -m option to display the server name, path name, and attributes of mount point */mnt* using the version 4 protocol with a public mountpoint.

```
# tso -t "mount type(NFS) filesystem(nfs1) mountpoint('/mnt') parm('sjvm5151:/hfs/u/public,public')"
mount type(NFS) filesystem(nfs1) mountpoint('/mnt') parm('sjvm5151:/hfs/u/public,public')

# /usr/lpp/NFS/nfsstat -m /mnt
server  sjvm5151
path    /hfs/u/public,public

hard    public  vers(4)         proto(tcp)
timeo(7)        retrans(3)      rpcbind(y)      accesschk(y)
delim(NA)       xlat(n)         cln_ccsid(1047) srv_ccsid(819)
convserv(LRE)   stringprep(n)   llock(n)
rsize(32768)    wsize(32768)    readahead(8)    delaywrite(16)
acregmin(3)     acregmax(60)    acdirmin(30)    acdirmax(60)
datacaching(y)  attrcaching(y)  retry(10)       dynamicsizeadj(y)
```

Figure 15. Displaying NFS mounted file system information with public mountpoint (Version 4 protocol only)

Displaying server mount information–showmount

Use the **showmount** command to display the remote NFS server mount information. If you omit the options, the default option displays hostnames of all remote mounts from the `hostname` NFS server. If you omit the *hostname* parameter, then the local hostname is used.

The following showmount command displays all remote mounts in the format `hostname:directory` from the local `hostname` NFS server.

```
showmount -a
```

The following showmount command displays only the directory names of all the remote mounts from the local *hostname* NFS server.

```
showmount -d
```

The following example shows the output from the showmount command using the -a option to display all mounts in the format hostname:directory from the hostname mvshost.

```
# showmount -a mvshost
mvshost.sanjose.ibm.com:/IBMUSER
usera.sanjose.ibm.com:/USER2
```

The following example shows the output from the showmount command using the -d option to display only the directory names of all mounts from the hostname mvshost.

```
# showmount -d mvshost
/IBMUSER
/USER2
```

The following example shows the output from the showmount command with no option specified To only display the hostnames of all remote mounts from the hostname mvshost.

```
# showmount mvshost
mvshost.sanjose.ibm.com
usera.sanjose.ibm.com
```

The following example shows the output from the showmount command using the -e option to display the exported directories from the hostname aix_server1.

```
USER1:/u/user1:>showmount -e aix_server1
Export list for host aix_server1:
/home/u/guest/test (everyone)
/usr/lpp/info      (everyone)
/tmp               (everyone)
```

The following example shows the output from the showmount command using the -e option to display the exported directories from the hostname mvshost. In this case, mvshost has the site attribute set to ***security***(*none*).

```
# showmount -e mvshost
No exported file systems for host MVSHOST
```

The following examples shows the output from the showmount command using the -e option to display the exported directories from the hostname mvshost. In this case, mvshost has the site attribute set to ***security***(*safexp*).

```
# showmount -e mvshost
Export list for host MVSHOST:
/IBMUSER             user1
```

Displaying default and mount point attributes–showattr

Use the **showattr** command to display the default attributes or the attributes that have been set for a specific mount point of the z/OS NFS server. If you specify a mount point, showattr shows the attributes for the mount point, including the overriding values. For descriptions of the attributes, see Chapter 9, “Initialization attributes for the z/OS NFS server,” on page 115 and “Mount command syntax and examples” on page 85.

If you omit the *hostname*, you must specify the /localpath.

The following is an example of the showattr command.

```
showattr mvshost1 /u/smith/mnt
```

Make sure that your version of the showattr command matches the release of NFS that you are using. Otherwise, the NFS server attributes will not display.

These examples show different ways you can use the showattr command.

Figure 16 shows a showattr command with just the hostname (`mvshost1` in this example) specified. The attributes for the server are displayed.

```
# showattr  mvshost1

GFSA988I Remote host does not have AF_INET6 interface.

FMID HDZ1C1N , last APAR HDZ1C1N, last changed module: GFSARXDR
Compiled at May  5 2010 09:15:39

z/OS Network File System Server Data Set Creation Attributes:

lrecl(8196)             recfm(vb)               blksize(0)
space(100,10)           blks                    dsorg(ps)
dir(27)                 unit()                  volume()
recordsize(512,4K)      keys(64,0)              nonspanned
shareoptions(1,3)       mgmtclas()              dsntype(pds)
norlse                  dataclas()              storclas()


z/OS Network File System Server Processing Attributes:

binary                  lf                      blankstrip
nofastfilesize          retrieve                maplower
mapleaddot              executebitoff           setownerroot
attrtimeout(120)        readtimeout(90)         writetimeout(30,120)
sync                    nofileextmap            xlat()
srv_ccsid(1047)         cln_ccsid(819)          notag
convserv(lre)           nordrverf               sidefile()


z/OS Network File System Server Site Attributes:

mintimeout(1)           nomaxtimeout            logout(1800)
nfstasks(8,16,8,4,4)    restimeout(48,0)
hfsprefix(/hfs)         mvsprefix(/mvs)         impprefix(mvs)
bufhigh(32M, 80%)       readaheadmax(16K)       cachewindow(112)
percentsteal(20)        maxrdforszleft(32)      logicalcache(16M)
smf(none,off)           nopcnfsd                security(safexp,safexp,safexp)
leadswitch              sfmax(0)                nochecklist
fn_delimiter(,)         readdirtimeout(30)      hfsfbtimeout(60)
upcase                  rec878                  mintasks(4,4,1)
noremount               fileidsize(64)          denyrw
nonlm                   nodhcp                  nostringprep
leasetime(120)          nodelegation            DlyDTimeout(10)
setgid(posix)
nfsv4domain()
public()                        mvssec(sys,krb5,krb5i,krb5p)
hfssec(sys,krb5,krb5i,krb5p)    pubsec(sys,krb5,krb5i,krb5p)
```

Figure 16. Displaying attributes

If you use the terse (-t) option, the following attributes display.

```
# showattr  -t  mvshost1

GFSA988I Remote host does not have AF_INET6 interface.
lrecl(8196),recfm(vb),blksize(0),space(100,10),blks,dsorg(ps),dir(27),unit(),
volume(),recordsize(512,4K),keys(64,0),nonspanned,shareoptions(1,3),mgmtclas(),
dsntype(pds),norlse,dataclas(),storclas()
binary,lf,blankstrip,nofastfilesize,retrieve,maplower,mapleaddot,executebitoff,
setownerroot,attrtimeout(120),readtimeout(90),writetimeout(30,120),sync,
nofileextmap,xlat(),srv_ccsid(1047),cln_ccsid(819),notag,convserv(lre),nordrverf,
sidefile()
mintimeout(1),nomaxtimeout,logout(1800),nfstasks(8,16,8,4,4),restimeout(48,0),
hfsprefix(/hfs),mvsprefix(/mvs),impprefix(mvs),bufhigh(32M,80%),
readaheadmax(16K),cachewindow(112),percentsteal(20),maxrdforszleft(32),
logicalcache(16M),smf(none,off),nopcnfsd,security(safexp,safexp,safexp),
leadswitch,sfmax(0),nochecklist,fn_delimiter(,),readdirtimeout(30),
hfsfbtimeout(60),upcase,rec878,mintasks(4,4,1),noremount,fileidsize(64),denyrw,
nonlm,nodhcp,nostringprep,leasetime(120),nodelegation,DlyDTimeout(10),
setgid(posix),nfsv4domain(),public(),mvssec(sys,krb5,krb5i,krb5p),
hfssec(sys,krb5,krb5i,krb5p),pubsec(sys,krb5,krb5i,krb5p)
hfssec(sys,krb5,krb5i,krb5p),pubsec(sys,krb5,krb5i,krb5p)
```

Ending your z/OS session - mvslogout

Use the **mvslogout** command to disconnect from the remote NFS server host. The mvslogout command is only required when the **mvslogin** command was used to begin the connection.

An mvslogout to an z/OS user ID cancels a prior mvslogin to the same z/OS user ID from the same local host.

Your account is automatically logged out if it is inactive for the period of time specified in the **logout** site attribute.

The following example disconnects the client from the remote NFS server machine, *mvshost1*.

```
mvslogout mvshost1
```

Chapter 8. Initialization attributes for the z/OS NFS client

This topic contains information about the attributes that are used by the z/OS NFS client.

Table 13 contains directive information about this topic's contents:

Table 13. Attributes - z/OS NFS client

Section	Page
Client attribute syntax	103
Datacaching attribute	110
Mount processing parameters and installation parameters	111

Client attribute syntax

Client attributes are described in Table 14.

Table 14. Client attributes

Attribute	Description

accesschk(**Y|N**)
Specifies whether the z/OS NFS client or NFS server is to check that the user has the requested access to the file or directory. If accesschk(Y) is specified, the z/OS NFS client performs the access check. If accesschk(N) is specified, the NFS server performs the access check.

The accesschk attribute default value is:

Y For mounts established with system authentication (sys)

N For mounts established with RPCSEC_GSS authentication (krb5, krb5i, or krb5p)

acdirmax(*n***)**
Specifies the maximum lifetime in seconds of cached directory attributes.

The **acdirmax** attribute default value is **60**.

acdirmin(*n***)**
Specifies the minimum lifetime in seconds of cached directory attributes.

The **acdirmin** attribute default value is **30**.

acregmax(*n***)**
Specifies the maximum lifetime in seconds of cached file attributes.

The **acregmax** attribute default value is **60**.

Table 14. Client attributes (continued)

Attribute **Description**

acregmin(*n*)
Specifies the minimum lifetime in seconds of cached file attributes.

The **acregmin** attribute default value is **3**.

attrcaching(Y|N)
Specifies whether to process attributes and data caching.

If attribute caching is in effect, the z/OS NFS client maintains cache consistency with the copy of the file on the NFS server by performing the consistency check with the cached file attributes. When a file's data is read, it remains valid on the z/OS NFS client until the attribute cache is timed out or negated. If **attrcaching(N)** is specified, it will automatically set **datacaching(N)**.

The attrcaching attribute default value is:

Y For mounts established with system authentication (sys)

N For mounts established with RPCSEC_GSS authentication (krb5, krb5i, or krb5p)

Note: attrcaching is not supported for Kerberos mounts (for example, krb5, krb5i, or krb5p). It is only supported for system authentication (sys) mounts. Specifying attrcaching(Y) is ignored with RPCSEC_GSS mounts.

biod(*n*)
Specifies the number of asynchronous block input/output (I/O) daemons.

The BIOD daemon runs on all NFS client systems. When a user on a client wants to read or write to a file on a server, the BIOD daemon sends this request to the server. The BIOD daemon is activated during system startup and runs continuously.

The number of daemons is based on the load the client can handle. Six to eight daemons can handle an average load. You must run at least one daemon for NFS to work.

The valid range is 1 to 32.

The **biod** attribute default value is **6**.

cln_ccsid(*x*)
Specifies the coded character set identifier (CCSID) for the local mounted file system.

The **cln_ccsid** attribute default is **1047** (LATIN OPEN SYSTEM EBCDIC).

convserv(*technique*)
Specifies the conversion technique-search-order that Unicode Services will use for specified **srv_ccsid(*x*)** and **cln_ccsid(*x*)** code pages. *Technique* consists of up to five technique-characters corresponding to the available techniques: R, E, C, L and M. See *z/OS Unicode Services User's Guide and Reference* for detailed descriptions on these conversion techniques.

NFS version 4 protocol (NFSv4) differs from NFSv2 and NFSv3 protocol in handling single to multiple byte conversion. Therefore, the technique-search-order specified in the **convserv()** attribute should consider the effects of the NFS protocol being used. See"Creating the conversion environment for Unicode Services" on page 164 for further details.

The **convserv** default is **LRE**.

Table 14. Client attributes (continued)

Attribute **Description**

datacaching(Y|N)

Specifies whether to perform data caching.

The **datacaching** attribute provides finer granularity in controlling whether file data should be cached by the z/OS NFS client. By caching the file data, all subsequent references to the cached data is done locally thus avoiding the network overhead. This has additional significance when obtaining data from NFS server systems which do not use UNIX access permissions for security as there is a potential security exposure allowing unauthorized users to access file data.

The datacaching attribute default value is:

Y For mounts established with system authentication (sys)

N For mounts established with RPCSEC_GSS authentication (krb5, krb5i, or krb5p)

Notes:

1. datacaching is not supported for Kerberos mounts (for example, krb5, krb5i, or krb5p). It is only supported for system authentication (sys) mounts. Specifying datacaching(Y) is ignored with RPCSEC_GSS mounts.
2. During mount , datacaching is turned off when security negotiation was attempted such that security was negotiated from sys to any of the krb flavors.
3. If attrcaching(N) is specified, it will automatically set datacaching(N).

delaywrite(*n*)

Specifies the maximum number of disk blocks for delay write.

The valid range is 0 to 32. The **delaywrite** attribute default value is **16**. The blocksize is 8192. This option is valid only when **datacaching=Y**.

Table 14. Client attributes (continued)

Attribute **Description**

delim (na | binary | nl | cr | lf | crlf | lfcr)

Specifies the line delimiter for record access to remote files through the basic sequential access method (BSAM), queued sequential access method (QSAM), and virtual storage access method (VSAM).

na Not specified. This value applies when the **delim** attribute is omitted. Note that this value must **not** be specified on the **delim** attribute. **na** can be specified only by omitting the **delim** attribute from the parameter list.

binary Specifies the data does not have record delimiters. The access method does not add a delimiter for each record on output and treats any delimiters on input as data.

The following text options can be specified:

cr Specifies that records are delimited by the EBCDIC carriage return character (x'0D').

crlf Specifies that records are delimited by the EBCDIC carriage return character followed by the EBCDIC line feed character (x'0D25').

crnl Specifies that records are delimited by the EBCDIC carriage return character followed by the EBCDIC new line character (x'0D15').

lf Specifies that records are delimited by the EBCDIC line feed character (x'25').

lfcr Specifies that records are delimited by the EBCDIC line feed character followed by the EBCDIC carriage return character (x'250D').

nl Specifies that records are delimited by the EBCDIC new line character (x'15').

disablella(Y | N)

Specifies the disabling or enabling of Lookup Look-Aside (LLA) caching.

The enabling of LLA cache with disablella(N) provides better performance, but may cause data integrity issues in a Shared File System sysplex or a network environment if file objects are altered by more than one system in the sysplex or network.

The disabling of LLA cache with disablella(Y) causes more LOOKUP requests from z/OS NFS client to z/OS NFS Server, thus affecting performance but guaranteeing data integrity. This is the recommended setting in a Shared File System sysplex environment or a network environment.

See *z/OS UNIX System Services File System Interface Reference* for further details on the use of disablella.

dynamicsizeadj(Y | N)

Specifies whether to perform the packet size adjustment for remote procedure call (RPC).

The **dynamicsizeadj** attribute default value is **Y**.

Table 14. Client attributes (continued)

Attribute | **Description**

llock(Y | N)

Specifies whether file locking requests are managed on your local z/OS UNIX file system or remotely on the NFS Server.

Y the local Byte Range Lock Manager (BRLM) manages the lock.

N The z/OS NFS Client sends various NFS Version 4 locking operations to the remote NFS Server to manage the file lock requests. The z/OS NFS Client and other participating NFS Clients can perform Byte Range Locking on the remote files at the supported NFS Server.

The **llock** attribute default value is **N**.
Note: This attribute is valid only for the NFS Version 4 protocol. If the NFS mounted file system is NFS Version 2 or 3 or if the NFS Server does not support the NFS Version 4 Locking then /usr/lpp/NFS/nfsstat will report **llock(y)**.

nfsv4domain(*NFSv4_default_domain***)**

Specifies the default domain for the NFS v4 protocol (NFSv4) name mapping.

The **nfsv4domain** attribute serves for redefinition of a name of this unique domain. In accordance with RFC3530 NFSv4 attributes "owner"and "owner_group"are transferred between the client and server in the form of "user_name@domain"and "group_name@domain". The client provides the mapping of names to id's and vice versa. *NFSv4_default_domain* identifies the user/group name space with one to one correspondence between the names and their numeric identifiers (uid's and gid's).

z/OS NFS Client will accept as valid a set of users and groups for default domain. The client will treat other domains as having no valid translations. If the nfsv4domain attribute is not used, the client uses the system-defined domain. The *NFSv4_default_domain* will be converted internally to lower case.

For further details on NFSv4 name mapping, see “NFS v4 protocol name mapping” on page 207.

proto(tcp | udp)

Specifies the transport protocol for the NFS client to communicate with the NFS server. By default, the NFS client selects the **proto** and **vers** with the following priorities:

1. **proto(tcp)** and **vers(4)**
2. **proto(tcp)** and **vers(3)**
3. **proto(udp)** and **vers(3)**
4. **proto(tcp)** and **vers(2)**
5. **proto(udp)** and **vers(2)**

Notes:

1. **proto(udp)** is functionally equivalent to **secure(udp)**
2. **proto(udp)** is mutually exclusive with the **vers(4)** parameter. **proto(udp)** is valid only for the NFS Version 2 and Version 3 protocols.
3. If **proto(tcp)** and **secure(udp)** are both in effect as mount parameters, **proto(tcp)** is ignored.

public

Forces the use of the public file handle when connecting to the NFS server.

This option is valid only during mount processing. The public keyword is valid only for the NFS version 4 protocol.

Table 14. Client attributes (continued)

Attribute	Description
readahead(*n***)**	Specifies the maximum number of disk blocks to read ahead. The block size is 8192 bytes. The valid range is 0 to 16. The **readahead** attribute default value is **1**.
retrans(*n***)**	Specifies the number of times to retransmit the NFS remote procedure calls (RPC). The valid range is 0 to 1000. The **retrans** attribute default value is **3**. This option is valid only when **soft** and **proto(upd)** are specified.
retry(*n***)**	Specifies the number of times to retry the mount operation. The valid range is 0 to 20,000. The **retry** attribute default value is **10000**. This option is valid only during mount processing.
rsize(*n*)	Specifies the read buffer size in *n* bytes. The valid range is 1 to 8192. The **rsize** attribute default value is **8192**.
rpcbind(Y|N)	Specifies whether the target NFS server platform supports the RPCBIND protocol, so the NFS client will not have to attempt to use the RPCBIND protocol if that protocol is not supported. The default is **rpcbind(Y)**, to indicate that RPCBIND is supported. If **N** is specified, the z/OS NFS Client will immediately use the PORTMAPPER protocol instead. This keyword has no effect if the client system is not enabled for IP version 6 (IPv6). The **rpcbind** default value is **Y**.

Table 14. Client attributes (continued)

Attribute	Description
secure(sys | krb5 | krb5i | krb5p | udp)	Specifies the transport protocol for the NFS client to use to bind reserved (privileged) ports when communicating to the NFS server. **Notes:** 1. **secure(sys)** uses the system authentication. 2. **secure(krb5)** provides Kerberos V5 based integrity on the RPC credentials (but not data) using the DES_MAC_MD5 integrity algorithm and uses the RPCSEC_GSS service of rpc_gss_svc_none. **secure(krb5)** is valid only for the NFS Version 4 protocol. 3. **secure(krb5i)** provides Kerberos V5 based integrity on both the RPC credentials and data using the DES_MAC_MD5 integrity algorithm and uses the RPCSEC_GSS service of rpc_gss_svc_integrity. **secure(krb5i)** is valid only for the NFS Version 4 protocol. 4. **secure(krb5p)** provides Kerberos V5 based integrity and privacy on both the RPC credentials and data using the DES_MAC_MD5 algorithm for integrity and 56 bit DES for privacy. It uses the RPCSEC_GSS service of rpc_gss_svc_privacy. **secure(krb5p)** is valid only for the NFS Version 4 protocol. 5. **secure(udp)** is functionally equivalent to **proto(udp)**. 6. If **secure(udp)** is specified, **proto(tcp)** is ignored and the NFS client uses **udp** as the transport protocol to communicate with the NFS server. 7. **secure(udp)** is mutually exclusive with the **vers(4)** parameter. **secure(udp)** is valid only for the NFS Version 2 and Version 3 protocols. During mount when sys,krb5,krb5i or krb5p is specified in the secure keyword, the client does not attempt a security negotiation.
soft|hard	Returns an error if the NFS server does not respond or continues to retry the NFS remote procedure call (RPC) until the NFS server responds. • If **hard** is specified, the NFS remote procedure call (RPC) is retried until the NFS server responds. • If **soft** is specified, an error is returned if the NFS server does not respond. The maximum number of retries is specified with the **retrans** option. This option is valid for all NFS RPCs under the mount point.
srv_ccsid(*x*)	Specifies the coded character set identifier (CCSID) for the remote mounted file system. The **srv_ccsid** attribute default value is **819** (ISO 8859–1 ASCII).
stringprep(Y|N)	Specifies whether z/OS NFS Client is to enable or disable stringprep normalization. Stringprep normalization is the NFS version 4 globalization function for converting inbound strings to UTF-8 format. The **stringprep** attribute default value is **N**.

Table 14. Client attributes (continued)

Attribute	Description
timeo(*n*)	Sets the remote procedure call (RPC) timeout to *n* tenths of a second. The **timeo** attribute default value is **7**.
vers(2|3|4)	Specifies the NFS protocol version that the client will use to communicate with the NFS server. If no version is specified, the z/OS NFS client will communicate with the NFS server at the highest protocol level that is supported by the server.
wsize(*n*)	Sets the write buffer size to *n* bytes. The **wsize** attribute default value is **8192**.
xlat(Y|N)	If **Y** is specified, the data in all the files are text and the NFS client will perform data conversion according to the **cln_ccsid** and **srv_ccsid** parameters. The **xlat** attribute default value is **N** and should be used for binary data.

Datacaching attribute

Security checking is done on the Network File System server to determine whether the requesting client user is authorized to access the data. On UNIX systems, this is done by validating the client's user ID and group ID against the file's permission codes. If the authorization checking is successful, the file data is returned to the z/OS NFS client system. Further authorization checking for subsequent access to the cached data or for other client users is done on the z/OS NFS client system.

For z/OS conventional MVS data set access through the z/OS NFS server, the user is required to present their z/OS credentials which are checked by the z/OS security system, such as RACF, before file data is returned. Since the z/OS system does not maintain UNIX style permission codes for MVS data sets, the z/OS NFS server returns a code indicating that anyone can access the file. This is done since passing any lesser access code to the client would result in the client user not being allowed to use the cached data which was already read. When the file data is cached on the z/OS NFS client system and another client user on this system attempts to access the same file data, the z/OS NFS client checks the returned permission codes to validate access. Since the z/OS NFS server has passed a code which allows anyone access to the file, all users on the client system can access the cached data without further restrictions. If data caching is turned off, no client caching takes place and each user must pass the server security check.

Based on the installation time out values, the file data cached by the client is flushed and further attempts to access the file data again requires passing server authorization.

The installation **datacaching** parameter can be set and it can be overridden for each mount point so that different mount points can be handled as required for the files under that mount point.

Notes:

1. attrcaching and datacaching are not supported for Kerberos mounts (for example, krb5, krb5i, or krb5p). It is only supported for system authentication (sys) mounts.
2. datacaching is turned off whenever there is a security negotiation from sys to any of the krb flavors during mount.
3. If attrcaching(N) is specified, it will automatically set datacaching(N).

If the potential security exposure can not be tolerated for sensitive file data, the **datacaching** should not be used so that no file data is cached by the z/OS NFS client.

Mount processing parameters and installation parameters

Table 15 shows the client attributes that can be modified when used as parameters on the MOUNT command.

Table 15. Mount processing parameters

acdirmax(n)
accesschk(Y|N)
acdirmin(n)
acregmax(n)
acregmin(n)
attrcaching(Y|N)
cln_ccsid(n)
convserv(UNICODE technique)
datacaching(Y|N)
delaywrite(n)
delim (na|binary|nl|cr|lf|crlf|lfcr)
dynamicsizeadj(Y|N)
hard|soft
llock(Y|N)
nfsv4domain(*NFSv4_default_domain***)**
proto(tcp|udp)
public
readahead(n)
retrans(n)
retry(n)
rpcbind(Y|N)
rsize(n)
secure(krb5|krb5i|krb5p|udp)
stringprep(Y|N)
srv_ccsid(n)
timeo(n)
vers(2|3|4)
wsize(n)
xlat(Y|N)

Table 16 shows installation parameters.

Table 16. Installation parameters

attrcaching(Y|N)
biod(n)
bufhigh(n)
cln_ccsid(n)
convserv(UNICODE technique)
datacaching(Y|N)
delaywrite(n)
delim (binary|nl|cr|lf|crlf|lfcr)
disablella(Y|N)
dynamicsizeadj(Y|N)
llock(Y|N)
readahead(n)
rpcbind(Y|N)
secure(krb5|krb5i|krb5p|udp)
srv_ccsid(n)
stringprep(Y|N)
xlat(Y|N)

The following conditions may cause the NFS client to fail its initialization:

- The NFS client is not started in a standalone colony address space.
- The NFS client is already started; multiple instances of the NFS client on a single z/OS system is not supported.

- Invalid parameter is specified in the installation parameters.
- If Unicode exists, then Unicode is used. If Unicode does not exist and Character Data Representation Architecture (CDRA) exists, then CDRA is used. If both Unicode and CDRA do not exist, then initialization fails.

A WTO message is issued to the operator console if the NFS client fails to initialize.

NFS client translation support

Table 17 contains NFS client attributes. See Table 22 on page 131 for more information about considerations for native ASCII environment support.

Table 17. z/OS NFS clients with non-z/OS based NFS servers

Client Attributes Specified	Mount Option	Read	Write
xlat(Y), cln_ccsid,srv_ccsid	No TAG specified	NFS client does translation	NFS client does translation
xlat(N)	TAG(TEXT,CCSID)	Logical file system does translation	Logical file system does translation
xlat(Y)	TAG(TEXT,CCSID)	Mount will fail	Mount will fail

Notes:

1. The logical file system will do translation when the **mount tag** option is specified. It will do translation based on the process tag (calling application) and file tag (if the file tag is not zeros or untagged). Otherwise, the system will do translation based on the process tag and **mount tag** for the CCSID information.
2. It is assumed that the user doing the mount knows the files being accessed from the remote non-z/OS file systems. So the CCSID needs to be set accordingly. Data written to the server will be stored in a specific CCSID format. To read it back correctly, the correct CCSID must be specified (for example, without it being translated with the wrong CCSID).

   For more information about client mount options, see Chapter 6, “Commands and examples for AIX and UNIX clients,” on page 63 and Chapter 7, “Commands and examples for z/OS NFS clients,” on page 77.

z/OS NFS client with z/OS NFS server

Both the client and server operate as described in “NFS client translation support” and “NFS servers with non-z/OS based NFS clients” on page 131.

In order to avoid double translation, the mount to the server must specify the correct **cln_ccsid** (server option) and the client TSO MOUNT command should not have the **tag** option. The client mount option **xlat(N)** should be specified so that only the server will do translation (if needed) and return the data in the correct CCSID.

```
mount filesystem(NFS001) type(nfs) mountpoint('/u/nfsdir')
          parm('mvsnfs:"/hfs/u/user,text,cln_ccsid(2000)",xlat(N)')
          vi /u/nfsdir/file1

**  Translation will be done by the server only based on file1's file tag and
    cln_ccsid of 2000.
```

In all other cases, double translation may occur as the server will do translation based on its file tag and **cln_ccsid** settings and the logical file system will do translation based on the process tag and the CCSID in the **mount tag** option. Caution must be used as double translation may result in the data becoming garbage.

Chapter 9. Initialization attributes for the z/OS NFS server

This topic contains information about the attributes that are used to manipulate files in the z/OS NFS server.

Table 18 contains directive information about this topic's contents.

Table 18. Attributes - z/OS NFS server

Section	Modification	Description	Page
Data set creation attributes syntax	Data set creation attributes can be modified by the client	Data set creation attributes provide information about an MVS file to the z/OS NFS server, such as the type of file, or how the file is allocated (for example, blocks, cylinders, or tracks)	116
Processing attributes syntax	Processing attributes can be modified by the client	Processing attributes provide information to the z/OS NFS server about how to handle the file, such as how long the files remain open, or whether the files are processed in text or binary format	121
Site attributes syntax	Site attributes can only be modified by the system administrator	Site attributes control the z/OS NFS server resources	134

Attributes used for z/OS UNIX file access

These attributes are specific to the following z/OS UNIX file access.

- **hfs***(prefix)*
- **hfsprefix***(prefix)*
- **sync** and **async**
- **extlink**

These attributes are relevant to accessing the following z/OS UNIX files as well as z/OS conventional MVS data sets.

- **impprefix***(opt1, opt2)*
- **restimeout** - Resource timeout
- **logout** - User log time out
- **security** - Security checking
- **text** - ASCII to EBCDIC data conversion and vice versa
- **binary** - No ASCII and EBCDIC
- **xlat** - Customized translation table

This attribute is relevant to accessing z/OS conventional MVS data sets.

- **mvsprefix***(prefix)*

© Copyright IBM Corp. 1991, 2010

Multipliers

Instead of entering the entire numeric values for the attributes, you can use the multipliers K (1024), M (1024 × 1024), or G (1024 × 1024 × 1024). For example, entering 10M is the same as entering 10,485,760.

Duplicate attributes

Specifying an attribute several times on a line does not cause an error. The line is read from left to right, and the last of any duplicate attribute is used. For example:

```
$ vi "file,recfm(vb),recfm(fb)"
```

This results in a file created with a fixed-blocked format.

Data set creation attributes syntax

The data set creation attributes are used to define the structure of MVS data sets when creating a file. These attributes correspond to the data control block (DCB) or the job control language (JCL) parameters used to define an MVS data set when it is created. See *z/OS MVS JCL Reference*for more information about data set creation attributes.

The data set creation attributes do not apply for z/OS UNIX files.

Table 19 describes data set creation attributes. Defaults are underlined **in this format**. You can override these attributes by using the **mount** command or file creation command. For PDS and PDSE, members have the same attributes as the data set attributes, so the file creation attributes specified for members are ignored.

Table 19. Data set creation attributes

Data Set Creation Attribute	Description
blks	Specifies that disk space is allocated by blocks, except for VSAM data sets. See the **space** attribute in this table.
cyls	Specifies that disk space is allocated by cylinders.
recs	Specifies that disk space is allocated by records for VSAM data sets. The **blks** and **recs** attribute values are identical for VSAM data sets.
trks	Specifies that the disk space is allocated by tracks.
blksize(0 | *quan***)**	Specifies the maximum length, in bytes, of a physical block on disk. The value of *quan* can range from **0** (the default value) to 32,760. If blksize(0) is specified, the system determines an optimal block size to use.
dataclas(*class_name***)**	Specifies the data class associated with the file creation. The *class_name* must be defined to DFSMS before it can be used by the client. The system-managed storage automatic class selection (ACS) routine must also assign a storage class to the file being created. If a data class is not specified in the attribute file, **dataclas()** is displayed by the **showattr** client enabling command. For more information about data classes, see *z/OS DFSMSdfp Storage Administration*.

Table 19. Data set creation attributes (continued)

Data Set Creation Attribute	Description

dir(27 | *quan*)

Specifies the number of 256-byte records needed in the directory of a PDS. Use the **dir** attribute with the **mkdir** command when you create a PDS.

The value of *quan* can range from 1 to 16,777,215. The default value is **27**. The maximum number of PDS members is 14,562.

dsntype(library | pds)

Specifies whether a PDSE or a PDS is to be created when the **mkdir** client command is used.

The following options can be specified.

library	Specifies partitioned data set extended (PDSE)
pds	Specifies partitioned data set (PDS)

You cannot create a PDS (or PDSE) within another PDS (or PDSE).

For more information about creating a PDS or a PDSE, see *z/OS DFSMS Using Data Sets*.

dsorg(*org*)

Specifies the organization of a data set.

The following *org* values can be specified.

da	Direct data set
indexed	VSAM KSDS data set
nonindexed	VSAM ESDS data set
numbered	VSAM RRDS data set
ps	Physical sequential (ps) data set

The **dsorg** attribute is ignored for directory-oriented client commands.

If you are using VSAM data sets in binary mode, then **nonindexed** is recommended.

keys(*len*, *off*)

Specifies the length and offset of the keys for VSAM KSDS data sets. The **keys** attribute can only be specified when using **dsorg(indexed)**.

The *len* and *off* values are specified in bytes.

len	Specifies a value between 1 and 255. The default value is **64**.
off	Specifies a value between 0 and 32,760. The default value is **0**.

When you create a VSAM KSDS data set, the records you are loading into it must be keyed-sequenced or the write fails. Each write of the data set is treated like a first load, and requires that the records being loaded are in ascending key sequence.

Table 19. Data set creation attributes (continued)

Data Set Creation Attribute	Description

lrecl(8196 | *quan*)
The value of *quan* specifies a record length.

1. Length, in bytes, for fixed-length records.
2. Maximum length, in bytes, for variable-length records. If the **blksize** attribute is specified, the value must be at least 4 bytes less than the **blksize** quantity.

The value of *quan* can range from 1 to 32,760. The default value is **8196**.

mgmtclas(*mgmt_class_name*)
Specifies the management class associated with the file creation. The *mgmt_class_name* must be defined to DFSMS/MVS before it can be used by the client. The system-managed storage automatic class selection (ACS) routine must also assign a storage class to the file being created. If a management class is not specified in the attribute file, **mgmtclas()** is displayed by the **showattr** client enabling command.

For more information about management classes, see *z/OS DFSMSdfp Storage Administration*.

recfm(*cccc*)
Specifies the format and characteristics of the records in the data set. The value of *cccc* can be 1 to 4 characters, in one of the following combinations.

```
[f | fb | fs | fbs]  [a|m]

u [a|m]

[v | vb | vs | vbs]  [a|m]
```

The following are valid record format characters.

a	ANSI control codes
b	Blocked
f	Fixed-length records
m	Machine control codes
s	Spanned for variable records, standard format for fixed records
u	Undefined-length records
v	Variable-length records

The **recfm** format characters `v`, `f` and `u` are mutually exclusive. The **recfm** format characters `a` and `m` are mutually exclusive.The format character `s` is not allowed for a PDS or PDSE.

recordsize(*avg*,*max*)
Specifies the average and maximum record size for VSAM data sets. The *avg* and *max* values are specified in bytes. They can each range from 1 to 32,760.

The default values are **512** and **4K**, respectively. **These values must be equal for VSAM RRDS.**

Table 19. Data set creation attributes (continued)

Data Set Creation Attribute **Description**

rlse
Specifies that unused space should be released from the data set the first time a new data set is closed. For slow clients with long pauses between writes, the **rlse** attribute causes space to be released from the primary extent prematurely. Any additional writes will cause secondary space to be allocated.

norlse
Specifies that unused space should not be released from the data set.

---

shareoptions(*xreg***,***xsys***)**
Specifies the cross-region and cross-system share options for a VSAM data set.

The value of *xreg* ranges from 1 to 4.

The value of *xsys* is either 3 or 4.

The default values are **1** and **3**, respectively.

For more information, see "Sharing VSAM Data Sets" in *z/OS DFSMS Using Data Sets*

This applies to VSAM data sets only. For spanned records of non-VSAM data sets, see the entry for the **recfm** attribute in this table.

spanned
Specifies that VSAM KSDS or ESDS data sets can contain records that span control intervals (spanned records).

nonspanned
Specifies that data sets do not have spanned records.

---

space(*prim[,aux]***)**
Specifies the amount of primary and auxiliary space allocated for a new data set on a direct access volume.

The value of *prim* is the number (from 0 to 16,777,215) of primary tracks, cylinders, or data blocks in the data set.

The value of *aux* (optional) is the number (from 0 to 16,777,215) of additional tracks, cylinders, or blocks allocated if more space is needed.

If the **space** attribute is not specified, the default is used. The default values are **100** and **10**, respectively.

---

storclas(*class_name***)**
Specifies the storage class associated with the file creation. The *class_name* must be defined to the DFSMS efore it can be used by the client. If a storage class is not specified in the attribute file, **storclas()** is displayed by the **showattr** client enabling command.

For more information about storage classes, see *z/OS DFSMSdfp Storage Administration*.

Table 19. Data set creation attributes (continued)

Data Set Creation Attribute	Description
unit(*unit_name***)**	Specifies the unit on which to create a data set. The *unit_name* is a generic or symbolic name of a group of DASD devices. The *unit_name* must be specified as 3390 for extended format data sets. If a side file name is not specified in the attribute file, **unit()** is displayed by the **showattr** client enabling command. **Notes:** • You cannot create or access tape data sets on an z/OS host using the z/OS NFS server. • You cannot create extended format data sets with the z/OS NFS server, except using ACS routines.
vol(*volser***)** or **volume(***volser***)**	Specifies the name of the DASD volume to use to store the created data set. **volume** or **vol** is the keyword, and the value of *volser* represents the volume name. If a volume is not specified in the attribute file, **volume()** is displayed by the **showattr** client enabling command. If a data set is system-managed, as determined by the DFSMS automatic class selection (ACS) routines, you can omit this attribute.

Processing attributes syntax

Processing attributes are used to control how files are accessed by the client.

Table 20 describes processing attributes. Defaults are underlined **in this format**. You can override the default processing attributes on the mount command or file processing commands.

Table 20. Processing attributes

Processing Attribute | **Description**

attrtimeout(*n*)

The time (in seconds) that the data set remains allocated after a **lookup** or **getattr** server operation.

The default value of *n* is **120**. The value of *n* can range from 1 to 32,767 (9 hours, 6 minutes, and 7 seconds).

Notes:

1. The **attrtimeout** value is normally greater than the **readtimeout** or **writetimeout** values.
2. With NFS version 2 and version 3 protocols, the lookup operation searches for a file in the current directory. If it finds the file, **lookup** returns information on the file's attributes and a file handle pointing to the file. With NFS version 4, neither the file's attributes nor the file handle are returned. The file handle is saved by the server and used as an anchor for accessing the file.
3. When using the NFS version 4 protocol, the **attrtimeout** value should be set to a value less than or equal to the lease time. Otherwise, it is possible for performance problems to occur when attempting to access MVS data sets.

noattrtimeout

The data set is not deallocated after a **lookup** or **getattr** operation.

For more information, see “Timeout attributes” on page 128.

binary

Indicates that the data set is processed between the client and server using binary format and no data conversion occurs between ASCII and EBCDIC formats.

text

Converts the contents in the data set between EBCDIC and ASCII formats. Use this format to share text data between clients and z/OS applications.

In **text** mode, the following attributes apply only to z/OS MVS data sets:

- **blankstrip** and **noblankstrip**. (See the entry for **blankstrip** in this table.)
- End-of-line terminators (**cr**, **crlf**, **lf**, **lfcr**, or **noeol**) are used to indicate the MVS logical record boundary. (See the entry for **lf** in this table. See “Text processing mode” on page 34 for rules of coding EOL terminators by the z/OS NFS server. See the **xlat** attribute in this table for customized EBCDIC-ASCII tables.)

Table 20. Processing attributes (continued)

Processing Attribute	Description
blankstrip With text mode, strips trailing blanks from the end of each record of a fixed-length text file when the file is read. Trailing blanks pad the end of each file or record when a text file is written. **noblankstrip** Does not strip trailing blanks from the end of fixed-length records when a fixed-length text file is read. Does not pad records when writing a text file. The file must be of the correct size or an I/O error is reported to the client. For information about the **text** attribute, see the entry for **binary** in this table. This attribute does not apply to z/OS UNIX files.	
With **text** mode, use one of the following end-of-line specifiers. **cr** Carriage Return is the end-of-line terminator. **crlf** Carriage Return followed by Line Feed is the end-of-line terminator (standard DOS). **lf** Line Feed is the end-of-line terminator (standard AIX or UNIX). **lfcr** Line Feed followed by Carriage Return is the end-of-line terminator. **noeol** No end-of-line terminator. For information about the **text** attribute, see the entry for **binary** in this table. This attribute does not apply to z/OS UNIX files.	
cln_ccsid(*n*) Specifies the coded character set identifier (CCSID) for the remote mounted file system (NFS client) when text is being translated. The default value of *n* is **819** (ISO 8859-1 ASCII).	
convserv(*technique***)** Specifies the conversion technique-search-order that Unicode Services will use for specified **srv_ccsid(***x***)** and **cln_ccsid(***x***)** code pages. *Technique* consists of up to five technique-characters corresponding to the available techniques: R, E, C, L and M. See *z/OS Unicode Services User's Guide and Reference* for detailed descriptions on these conversion techniques. NFS version 4 protocol (NFSv4) differs from NFSv2 and NFSv3 protocol in handling single to multiple byte conversion. Therefore, the technique-search-order specified in the **convserv()** attribute should consider the effects of the NFS protocol being used. See"Creating the conversion environment for Unicode Services" on page 164 for further details. The default value of *technique* is **LRE**.	

Table 20. Processing attributes (continued)

Processing Attribute	Description
executebiton	Sets the execute permission bits in user, group, and other (as reported with the **ls** AIX or UNIX command) for a mount point's files. Use when storing executable or shell scripts on the z/OS system. This option can only be overridden on a mount point basis — not at a command level. The **executebiton** attribute does not apply to z/OS UNIX files and can only be used with the **mount** command.
executebitoff	Does not set execute bits in user, group, and other for the mount point's files. This value is normally used in the site file.
extlink	Specifies the use of the external link command to create, process, and delete a symbolic link to an MVS data set. The **extlink** attribute is used with the following commands. **ln -s** Create a symbolic link to an MVS data set. **ls -l** Display the attributes and contents of the symbolic link. **rm** Delete the symbolic link. The **extlink** attribute only applies to z/OS UNIX file objects. If **extlink** is not specified then it is not displayed by the **showattr** client enabling command.
fastfilesize	Specifies to get the file size from SPF statistics, if it exists, for direct data sets, PDSs, and non-system-managed data sets.
nofastfilesize	Specifies to read the entire file or member to get the file size for direct data sets, PDSs, PDSEs, and non-system-managed data sets. Using the **nofastfilesize** attribute might cause a noticeable delay when first accessing very large data sets. These attributes apply to MVS data sets, but do not apply to z/OS UNIX files. For more information, see "Using fastfilesize to avoid read-for-size" on page 450.
fileextmap	Enables file extension mapping. The **fileextmap** attribute can be specified at the file command level for the client platforms that support passing of attributes. See "File extension mapping" on page 40 for related information.
nofileextmap	Disables file extension mapping.

Table 20. Processing attributes (continued)

Processing Attribute **Description**

mapleaddot
Enables mapping of a single leading "." from a client file name to a legal leading "$" on z/OS. The **mapleaddot** attribute should normally be enabled for access by AIX and UNIX clients.

nomapleaddot
Disables mapping of a single leading "." from a client to a leading "$" on z/OS.

These attributes do not apply to z/OS UNIX files.

maplower
Enables mapping of lower case file names to upper case when accessing files on z/OS, and back to lower case when sending to the network. This option should normally be enabled for access by AIX or UNIX clients. This option only affects file names (high-level qualifiers and user catalog aliases).

nomaplower
Disables mapping of lower case file names to upper case and back to lower case when using files on z/OS.
Note: These attributes do not apply to z/OS UNIX files.

mapped
The **mapped** attribute should be specified at the mount or site level when a mixed set of data types is to be processed under a single mount point. The determination of whether the data is to be processed as text or binary depends on the rules that are established in the specified side file. See “File extension mapping” on page 40 for related information.

If a file extension is not mapped to text or to binary using the side files, then the data will be processed according to what has been specified at the mount or site level (**binary** or **text**).

If **binary** or **text** is specified at the file command level, that overrides the **mapped** specification. If **mapped** is not specified then it is not displayed by the **showattr** client enabling command.

mvsmnt
With NFS Version 4, the **mvsmnt** attribute should be specified on user mount commands. In NFS Version 4, mount requests are passed to the server in the form of a PUTROOTFH operation followed by a sequence of **lookup** operations. The **mvsmnt** attribute indicates to the z/OS NFS server that the associated **lookup** operation is emulating a mount procedure and causes the z/OS NFS server to write the mount point to the mount handle database (MHDB), so the server can automatically recover the mount point during a server restart. With the mvsmnt attribute specified, z/OS NFS server site security attributes **saf** and **safexp** are used to control access to z/OS UNIX file systems. **mvsmnt** is not displayed by the **showattr** client enabling command.

Table 20. Processing attributes (continued)

Processing Attribute	Description

nordrcache
Specifies that the server should not stale the legacy (MVS z/OS conventional data) internal readdir cache if an addition is made to the directory. This causes the next READDIR operation to access the directory information from the Physical File System (PFS) rather than the internal readdir cache.

| The default value is **nordrcache** and is not displayed by the **showattr** client enabling
| command.

The **nordrcache** attribute does not apply to z/OS UNIX files.

When **nordrcache** is not specified, the addition of an entry to the legacy internal readdir cache will not be visible to the client until the next readdir cache timeout or a remove from that directory. When **nordrcache** is specified, the addition will be visible to the client by the subsequent READDIR, whether the readdir cache timeout has expired or not. This may impair performance because the directory list must be read from the Physical File System after any addition to the cached directory. When **nordrcache** is specified, if no changes are made to the internal readdir cache, the cache does remain available until the readdir cache timeout expires.

nordrverf
Specifies not to perform cookie verifier checking for the NFS version 3 **readdir** and **readdirplus** procedures, and the NFS version 4 **readdir** procedure.
Note: nordrverf does not provide consistency in the listing of a directory's content and may cause duplicate or omit entries when the directory is changing during the listing.

rdrverf
Specifies to perform cookie verifier checking for the NFS version 3 **readdir** and **readdirplus** procedures, and the NFS version 4 **readdir** procedure.
Note: rdrverf provides consistency in the listing of a directory content and, as a result, may impact performance.

readtimeout(*n*)
The **readtimeout** attribute specifies the amount of time in seconds before a data set is released after a read operation.

The value of *n* can range from 1 to 32,767 (9 hours, 6 minutes, and 7 seconds). The default value of *n* is **90**. The server closes the file when the file times out.

The **readtimeout** attribute does not apply to z/OS UNIX files.
Note: When using the NFS version 4 protocol, the **readtimeout** value should be set to a value less than or equal to the lease time. Otherwise, it is possible for performance problems to occur when attempting to access MVS data sets.

noreadtimeout
Specifies the data set is not deallocated after a read operation.

For more information, see “Timeout attributes” on page 128.

Table 20. Processing attributes (continued)

Processing Attribute **Description**

The z/OS NFS server uses DFSMShsm to recall or delete migrated files. The action that the server takes against migrated files depends on which of the **retrieve** or **noretrieve** attributes is active.

The **retrieve** and **noretrieve** attributes do not apply to z/OS UNIX files.

retrieve

When the **retrieve** attribute is active, the server will recall the migrated file if necessary, upon an NFS_LOOKUP request for the file, depending on the files status.

The server may be able to obtain the migrated files attributes without recall (see “Retrieve attributes” on page 129 for additional information). If not, the recall operation is started by the server. The server waits for the recall operation to complete if the file resides on DASD. If the file does not reside on DASD, the server does not wait for the recall operation to complete and returns a “device not available” message. You can attempt accessing the file again later when the recall is complete.

retrieve(*wait*)

When the **retrieve(wait)** attribute is active, the server waits for the recall to finish.

retrieve(*nowait*)

When the **retrieve(nowait)** attribute is active, the server does not wait for the recall to finish, and immediately returns a “device not available” message. You can attempt accessing the file again later when the recall is complete.

noretrieve

When the **noretrieve** attribute is active, the server does not recall the file, and returns “device not available” upon an NFS_LOOKUP, NFS_READ, or NFS_CREATE request for a migrated data set.

For more information, see “Retrieve attributes” on page 129.

---

setownerroot

Specifies that z/OS NFS server return root user authority as the owner of a z/OS MVS data set when the client is logged on as superuser. setownerroot does not grant root authority for a UID=0 user, instead see the mvslogin command or the root keyword in the exports file.

The **setownerroot** attribute can only be used with the **mount** command and does not apply to z/OS UNIX files.

setownernobody

Specifies setting the user ID in a file's attributes to nobody (65534), for a superuser.

---

sidefile(*dsname***)**

Specifies the name of the data set that contains the rules for file extension mapping purposes. If a side file name is specified in the attributes data set, then it is used as the default side file for the NFS server. A user can also specify an additional side file name during a mount operation to be used along with the default. The mapping rules will first be searched in the side file specified during the **mount** command and then the default side file is searched. To allow file extension mapping, a side file name must be specified either as a default or in the **mount** command. The value of *dsname* is a fully-qualified MVS data set name without quotation marks. See “File extension mapping” on page 40 for related information.

If a side file name is not specified in the attribute file or in the **mount** command, **sidefile**() is displayed by the **showattr** client enabling command.

Table 20. Processing attributes (continued)

Processing Attribute **Description**

srv_ccsid(*n*)
Specifies the coded character set identifier (CCSID) for the local mounted file system (NFS server) when a new file is being created.

The **srv_ccsid** attribute has no effect on the translation of an existing file's data.

The default value of *n* (if specified) is **1047** (`LATIN OPEN SYSTEM EBCDIC`).

If the **srv_ccsid** attribute is not specified, new z/OS UNIX files will continue to be created as untagged.

sync
Specifies that data transmitted with the **write** request should be committed to nonvolatile media (for example, DASD) by the server immediately when received.

async
The user can alternatively specify the **async** processing attribute to get improved performance. When this attribute is specified, the data will be committed asynchronously.

The **sync**|**async** attribute only applies to z/OS UNIX file objects and only for the NFS version 2 protocol.

tag
Specifies that the newly created files should receive a file tag.

notag
Specifies that the newly created files should not receive a file tag. The tag is set to 0x0000.

Table 20. Processing attributes (continued)

Processing Attribute — **Description**

writetimeout(*n*,*o*)

Specifies the amount of time *n*, in seconds, before a data set is released after a write operation and the amount of time *o*, in seconds, that the server will wait for data to arrive to complete a partial record before closing the data set.

The value of *n* can range from 1 to 32,767 (9 hours, 6 minutes, and 7 seconds). The default value of *n* is **30**.

The value of *o* can range from *n* to 255 * *n*. The default value of *o* is **120**.

The server closes the file when the file times out. All cached buffers are forced to disk. Normally **writetimeout** values are kept short because write operations result in exclusive locking. However, for slow client machines with long pauses between writes, you should increase the **writetimeout** value.

The server will use the *o* value to extend the **writetimeout** value for a data set processed in text or binary mode in the case of a partial record (no end-of-line terminator discovered in the record or RPC WRITE data was lost in the network) on a WRITE operation to delay file closing and wait for the record completion data to arrive, so that the server is able to correctly process the partial record.

The **writetimeout** attribute does not apply to z/OS UNIX files.

Note: When using the NFS version 4 protocol, the **writetimeout** value should be set to a value less than or equal to the lease time. Otherwise, it is possible for performance problems to occur when attempting to access MVS data sets.

nowritetimeout

Specifies that the data set is not deallocated after a write operation.

For more information, see "Timeout attributes."

xlat(*member_name*)

Specifies how to override the installation default translation table during file processing. The *member_name* is the member name of the PDS or PDSE that contains the customized translation table.

The system administrator defines this member name in the attribute data set, and PDS or PDSE in the startup procedure. The **xlat** attribute is ignored if specified on the command line.

If a customized translation table is not specified in the attribute file or in the **mount** command, **xlat()** is displayed by the **showattr** client enabling command.

znfsclient

Specifies that the NFS client is a z/OS NFS client. The z/OS NFS server uses this attribute to customize its response to the NFS client.
Note: Do **not** specify this attribute. This attribute is automatically appended to the processing attribute string by the z/OS NFS client when it detects that it is sending the mount request (or the last LOOKUP for an NFS version 4 mount) to a z/OS NFS Server. **znfsclient** is not displayed by the **showattr** client enabling command.

Timeout attributes

The values of the following attributes depend on the settings of the associated site attributes:

attrtimeout, **readtimeout**, and **writetimeout** – These attributes must be within the ranges specified by the **maxtimeout** and **mintimeout** site attributes.

Note: When using the NFS version 4 protocol, these timeout values should be set to a value less than or equal to the lease time. Otherwise, it is possible for performance problems to occur when attempting to access MVS data sets.

noattrtimeout, **noreadtimeout** and **nowritetimeout** – These attributes are valid only when **nomaxtimeout** is specified in the site attributes.

There are three processing attributes which control when files are timed out: **attrtimeout**, **readtimeout**, and **writetimeout**. The server determines which of these timeouts are in effect based on the last file operation. Thus when an existing file is appended, the file cannot be accessed before it times out in the time specified for **writetimeout** and is released by the server, because write operations result in exclusive locking. Similarly, if a file is read, it is not released before it times out in the time specified for **readtimeout** seconds.

The **readtimeout** and **writetimeout** attributes do not apply to the MVS data set or member being opened by NFS version 4 OPEN operation because there is a stateful CLOSE operation that closes and releases the data set or member.

The **readdirtimeout** site attribute controls the internal **readdir** cache used by directory lookups of MVS z/OS conventional data sets to be timed out or discarded based on a customizable value. The default is 30 seconds.

Retrieve attributes

The server deletes the migrated file upon an NFS_REMOVE request for a file, regardless if the **retrieve** or the **noretrieve** attribute is active. Typically, an NFS_REMOVE request is preceded by an NFS_LOOKUP request. If the data set was migrated with DFSMS/MVS 1.2 or below, the retrieve attribute causes a recall because NFS_LOOKUP processing needs to open the data set and read for size. If the data set was migrated under DFSMS/MVS 1.3 and DFSMShsm 1.3, and is SMS managed, its attributes were saved on DASD; therefore it is not always necessary to recall the data set to read for size and the data set may be deleted without recall. If the **noretrieve** attribute is active, the NFS_LOOKUP can return a "device not available" message. If the client code decides to ignore the error and issue the NFS_REMOVE, the migrated file is then deleted.

The UNIX command **ls mvshost** does not issue requests for individual files under the **mvshos**t directory. Migrated files under the **mvshost** directory are displayed, but are not recalled. However, the UNIX command **ls -l mvshost** issues NFS_LOOKUP requests for individual files under the **mvshost** directory.

Mapped keyword processing attribute

Table 21 contains mapped and existing keyword information.

Table 21. The mapped keyword and existing keywords

SFMAX	SIDEFILE(NAME)	MAPPED	ACTION
=0	X	Don't care	Data processed using existing rules for binary/text
	I		Server won't come up
	M		MOUNT will fail

Table 21. The mapped keyword and existing keywords (continued)

SFMAX	SIDEFILE(NAME)	MAPPED	ACTION
1–2000	I + M	SET	File extension used in the MOUNT-specified side file and then the site-specified side file. If an extension is not found, the existing rules for binary/text will be used.
		NOT SET	Data processed using existing rules for binary/text
	I	SET	File extension used in the site-specified side file. If an extension is not found, the existing rules for binary/text will be used.
		NOT SET	Data processed using existing rules for binary/text
	X	Don't care	Data processed using existing rules for binary/text
	M	SET	File extension used in the mount-specified side file. If an extension is not found, the existing rules for binary/text will be used.
		NOT SET	Data processed using existing rules for binary/text

Legend:

```
I = side file specified in installation table
M = side file specified in mount command
X = no side file specified
```

Native ASCII processing attributes

The **cln_ccsid(n)** and **srv_ccsid(n)** attributes can be specified either as installation defaults or at mount time for more granularity between different mount points. Unless **srv_ccsid** is specified either as an installation default or at mount time, newly created files will not have any file tag set (that is, the file tag is all zeros). These two attributes affect translation only when text processing is involved and only when an existing file has a nonzero or a nonbinary file tag.

Special attention must be paid to the different server attributes specified. See Table 17 on page 112 and Table 22 on page 131.

Considerations for native ASCII environment support

For applications running on z/OS V1R2 (and higher), a native ASCII environment is provided for z/OS UNIX file processing.

In this environment, applications can operate on files in either EBCDIC or ASCII format as well as other data formats defined with a coded character set identifier (CCSID) without translation, provided the data is already defined and stored in the data format wanted.

For the z/OS NFS server to operate properly on z/OS UNIX files in this environment, consider the following important factors:

- Unicode Services must be installed and set up on the system to let the NFS server use it for text translation. With the NFS version 4 protocol, z/OS NFS conversion of UTF-8 text data and metadata requires setting up a conversion environment using the z/OS Unicode Services by creating a Unicode conversion image that defines conversion tables with UTF-8 [CCSID 1208].
- Two processing attributes, **cln_ccsid** and **srv_ccsid**, are available for the NFS server for translation purposes as well as for the creation of new files. The

srv_ccsid attribute determines the CCSID of newly created z/OS UNIX files. If **srv_ccsid** is not specified as an installation default or at mount time, then new files continue to be created as untagged, or with a tag of 0x0000 and the old translation method of using translation tables specified by the **xlat** keyword applies.

- Processing (read/write) of tagged files depends on the different server options specified.

NFS servers with non-z/OS based NFS clients

Table 22 contains NFS server options (file tagging with Unicode Services active).

Table 22. File tagging with Unicode Services active

Server Options Specified	File Tag	Read	Write	Create
text,notag[4]	Untagged or Tag=0x0000 or Tag=0xFFFF	Translation using the current xlat tables	Translation using the current xlat tables	New file created with Tag=0x0000
text,notag	Yes	xlate using Src=FileTag and Tgt=cln_ccsid	xlate using Src=cln_ccsid and Tgt=FileTag	N/A (file exists)
text,cln_ccsid,srv_ccsid,notag[12]	Untagged or Tag=0x0000 or Tag=0xFFFF	xlate using Src=srv_ccsid and Tgt=cln_ccsid	xlate using Src=cln_ccsid and Tgt=srv_ccsid	New file created with Tag=0
text,cln_ccsid,srv_ccsid,notag	Yes[3]	xlate using Src=FileTag and Tgt=cln_ccsid	xlate using Src=cln_ccsid and Tgt=FileTag	N/A (file exists)
text,tag	Untagged or Tag=0x0000 or Tag-0xFFFF	xlate using Src=site attribute srv_ccsid and Tgt=cln_ccsid[10]	xlate using Src cln=ccsid and Tgt=site attribute srv_ccsid[10]	New file created with Tag=srv_ccsid[2]
text,tag	Yes[3]	xlate using Src=FileTag and Tgt=cln_ccsid	xlate using Src=cln_ccsid and Tgt=FileTag	N/A (file exists)
binary,notag[6]	Untagged or Tag=0x0000 or Tag-0xFFFF	No translation	No translation	New file created with Tag=0x0000
binary,notag	Yes[13]	No translation	Fail operation	N/A (file exists)
binary,tag	Untagged or Tag=0x0000 or Tag=0xFFFF	No translation	No translation	If ccsid on mount, Tag=srv_ccsid else Tag=0xFFFF
binary,tag	Yes[13]	No translation	If FileTag!=srv_ccsid fail op else no xlate	N/A (file exists)

Table 22. File tagging with Unicode Services active (continued)

Server Options Specified	File Tag	Read	Write	Create

Notes:

1. Writing to a file that has a tag that is different form the **srv_ccsid** (regardless whether the file is empty or not) will result in the file tag overriding the specified srv_ccsid when text is specified.
2. If srv_ccsid is specified (as an installation default or at mount), then the file is created with the srv_ccsid tag. Otherwise an untagged file is created.
3. xlat is ignored when the file being accessed is tagged.
4. xlat is optional. For untagged files, translation is done using default xlat tables, or custom **xlat** tables (if specified).
5. There is no facility in the NFS server to change an existing file tag. This must be done outside the NFS server.
6. Specifying the binary option overrides any **cln_ccsid** and srv_ccsid specified.
7. All files created by the server when text and srv_ccsid are specified will also have the TXTFLAG set to ON.
8. The NFS file tagging function assumes that Unicode Services is installed and activated on the system (available as of OS/390 V2.8).
9. If Unicode Services is not activated, only the NOTAG option is valid
   a. If the TAG option is specified in the site attributes, the NFS server start-up will fail.
   b. If the TAG option is specified on the **mount** command, the mount will fail.
10. File create and write are atomic operations. A file is created before it can be written. Thus, a file always already exists when it is written, and the attributes are used accordingly.
11. For reading or writing untagged files when in text, TAG mode, NFS uses the default server CCSID from the site attribute file. Any **srv_ccsid** values specified on the mount command will be ignored for reading or writing files in this case. The mount srv_ccsid will still be used for file creation however.
12. If TAG is specified in site attribute file, the site attribute **srv_ccsid** and **cln_ccsid** are always used for translating file names.

    If TAG is specified in site attribute file but Unicode Services is not active, the NFS server will shut down.

    If NOTAG is specified in the site attribute file, the site attribute **xlate(table)** is always used for translating file names.
13. These CCSIDs must be specified on the mount.

Table 23 contains NFS server options (file tagging with Unicode Services not active).

Table 23. File tagging with Unicode Services not active

Server Options Specified	File Tag	Read	Write	Create
text,notag	Untagged or Tag=0x0000 or Tag=0xFFFF	Translation using the current xlat tables	Translation using the current xlat tables	New file created with Tag=0x0000
text	Yes	Translation using the current xlat tables	Fail operation	N/A (file exists)
binary,notag	Untagged or Tag=0x0000 or Tag=0xFFFF	No translation	No translation	New file created with Tag=0x0000

Note: The TAG option is not valid if the Unicode Services are not active.

To support the correct use of Unicode Services the CONVSERV processing attribute is added. The values of this attribute defines the technique search order, or how Unicode Services processes specified code pages. See “Creating the conversion environment for Unicode Services” on page 164 for descriptions of the values.

The value of this attribute should exactly concur with the value of the technique search order, that was used during the current Unicode Image generation.

Site attributes syntax

You can use the site attributes to control z/OS NFS server resources.

Table 24 describes the site attributes. Defaults are underlined **in this format**. Some initial settings are shown, but the system administrator might have changed these settings, so use the **showattr** command to show the actual settings being used. The site attributes cannot be modified by client users.

Table 24. Site attributes

Site Attribute **Description**

bufhigh(*xx*, *yy*)

Specifies the storage limit for data buffers for the NFS client. If the combined total specified in the bufhigh *xx* and logicalcache attributes is greater than the available storage in the extended private area (implied by the REGION parameter in your procedure) at startup, the server shuts down immediately. A higher number of bufhigh *xx* means more caching and potentially better read performance.

xx the storage limit (in bytes). The valid range is 1 byte to 2047 MB.

yy the watermark in percent of the storage limit (*xx*) for printing a data buffer utilization alert message. The valid values are 0 (which turns off the data buffer utilization reporting mechanism) and from 50 to 90. Note that an invalid value of *yy* is converted to the default of 80 percent.

At z/OS Server startup, the actual value (*xx*) specified with bufhigh may be adjusted by the z/OS NFS Server internally depending on the available region size and other z/OS NFS Server memory requirements to enable the z/OS NFS Server to execute properly.

The **bufhigh** attribute default storage limit (*xx*) is **32** MB. If no percentage (*yy*) is specified, the default is 80 percent.

cachewindow(*n*)

Specifies the window size used in logical I/O to buffer client block writes received out of order. The value of *n* is a number from 1 to 2048 (the default is **112**). The **cachewindow** attribute does not apply to z/OS UNIX files. The suggested value is some small multiple of the number of BIODs running on a client. The general rule in setting the *n* value of **cachewindow(n)** is

```
n = (( num of BIOD + 1 ) * (client_max_IO_buffer_size/transfer_size))
```

- *num of BIOD* is the number of blocked I/O daemons set by the client workstation. This value is usually set to defaults at the installation of the operating system or by your system administrator.
- *client_max_IO_buffer_size* is the amount of I/O data requested by the client (for example, client writes 8192 bytes of data to the remote file system). This value is determined by your application programs.
- *transfer_size* is the actual size of data being sent across the network (for example, the 8192 bytes of data can be broken down to 16 smaller packets of 512 bytes (16x512=8192)). This value is determined dynamically by your client workstation.

Table 24. Site attributes (continued)

Site Attribute	Description
checklist	When specified, the server bypasses **saf** checking (even when **saf** or **safexp** is specified) for the list of files and directories underneath mount points which either matches a mount point entry or is a child of a mount point entry specified on the **dirsuf** parameter in the exports data set. CHECKLIST is only valid if SAF checking is the security option for the particular data access; otherwise, it is ignored even if it is specified. See GFSAPEXP in NFSSAMP library for a sample exports data set.
nochecklist	When specified, the server operates as before and ignores the information that is specified on the **dirsuf** parameter in the exports data set.
delegation	When specified, the server temporarily delegates management of a file's resources to an NFS client for NFS Version 4. When a file's management is delegated to an NFS client, all file access requests can be managed locally by the NFS client while the file is delegated.
nodelegation	When specified, the server does not delegate management of a file's resources to an NFS client for NFS Version 4.
denyrw	When specified, the server honors deny requests for file share reservations (the Windows Share_Deny value) from the NFS client. The deny requests may be specified on an NFS V4 Open operation or an NLM_share RPC.
nodenyrw	When specified, the server ignores deny requests from NFS clients (the Windows Share_Deny value), and treats the requests as if deny_none were specified.
dhcp	When specified, the server accepts dynamic IP addresses for the NFS client, using the dynamic host configuration protocol (dhcp). The client system must have a static host name and must dynamically update the DNS server with their IP address changes.
nodhcp	When specified, the server supports only NFS clients that use a static IP address.
DlyDTimeout(*n*)	Specifies the minimum delay detection time value in sec before the delay detection mechanism observes a delay in an external call/function and prints message GFSA1030E on the console. Valid values are 0 and a range of 5 to 60 seconds. Any value of DlyDtimeout from 1 to 4 seconds is converted to 5 seconds. If DlyDTimeout is set to 0 the delay detection mechanism is turned off. The default value is 10 seconds.
fileidsize(*n*)	Specifies how to control the handling of fileid sizes by the NFS server in NFS objects. Fileids may be recognized either as 32-bit or 64-bit addresses. Valid values are 32 and 64. The default value is **fileidsize(64)**.

Table 24. Site attributes (continued)

Site Attribute	Description

fn_delimiter

Specifies a character 'c' to be used as a delimiter between the file name and the attributes that follow it. This capability allows those sites that have UNIX data sets containing commas to copy and store their data on the NFS server. The following example specifies the default delimiter as a semicolon:

```
fn_delimiter(;)
```

So a user can process a file called 'comma,in-name' by entering:

```
vi "comma,in-name;text,lf"
```

Notes:

1. If the comment symbol was set as "**;**" with the *altsym* keyword and the fn_delimiter(;) attribute uses semicolon then the fn_delimiter semicolon will be treated as a delimiter between the file name and the attributes that follow, not as a comment symbol.
2. It is admissible to use the semicolon as the comment symbol after the right parenthesis if the altseq keyword is used.
3. The following example shows allowable multi-line syntax:

   ```
   fn_delimiter   \
    (;) ; the second semicolon is the comment symbol if the altseq keyword is used
   ```

   (;) must be located on one line.

A user can also include a default file name delimiter as a comma as follows:

```
fn_delimiter(,)
```

fn_delimiter(,)

The default file name delimiter is a comma.

hfs(*prefix***)** or **hfsprefix(***prefix***)**

Specifies a z/OS UNIX file system *prefix* to be imbedded in the mount directory path name. The default value of the z/OS UNIX file system prefix is **/hfs**. Mount requests received by the z/OS NFS server beginning with the z/OS UNIX file system prefix value are identified as mount requests for z/OS UNIX. The z/OS UNIX file system prefix value is not part of the path name.

Notes:

1. hfsprefix is preferred and should be used in future updates, but hfs is still accepted.
2. The z/OS UNIX file system must already be mounted locally by z/OS UNIX. Otherwise, the client mount request will fail.
3. The prefix value can only be 7 characters or less including the beginning "/"
4. The prefix value is case insensitive. It is always folded to upper case.

Table 24. Site attributes (continued)

Site Attribute	Description
hfsfbtimeout(*n***)**	Specifies how to control the timeout of the z/OS UNIX vnode token used by the NFS server. The timeout value controls how long before *vnode tokens* saved in file blocks are released. The valid range is 1 to 32,767 seconds. • The value of *n* can go as low as 1 second but to avoid the possibility of the client hanging (because of network delays). The value of *n* is not recommended to be lower than 5 second. • The value of *n* may need to be increased if the network is slow and the accessed directory has a lot of entries. The **hfsfbtimeout** attribute default value is **60** seconds.
hfssec(*krb5***,***krb5i***,***krb5p***,***sys***)**	Specifies the acceptable network transmission levels of security which can be used as the authentication flavor on NFS version 4 requests for accesses to z/OS UNIX files. This attribute is only used when not overridden by authentication specifications in the exports file. Multiple values for this attribute can be specified using the comma as delimiter. The following are the supported values: **krb5** Provides Kerberos V5 based integrity on the RPC credentials (but not data), when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 integrity algorithm and the RPCSEC_GSS service of rpc_gss_svc_none. **krb5i** Provides Kerberos V5 based integrity on both the RPC credentials and data, when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 integrity algorithm and the RPCSEC_GSS service of rpc_gss_svc_integrity. **krb5p** Provides Kerberos V5 based integrity and privacy on both the RPC credentials and data, when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 algorithm for integrity and 56 bit DES for privacy. The RPCSEC_GSS service used here is rpc_gss_svc_privacy. **sys** Specifies that the AUTH_SYS authentication flavor can also be used to access this file system. Note that the AUTH_SYS authentication flavor does not provide any integrity or privacy protection. The **hfssec** attribute default is **hfssec(sys,krb5,krb5i,krb5p)**. **Note:** File systems that require integrity or privacy protection over network transmissions of data should explicitly specify the desired settings. Do not rely on the default settings, because the default settings allow for RPC accesses using the AUTH_SYS authentication flavor, which does not provide any integrity or privacy protection.

Table 24. Site attributes (continued)

Site Attribute **Description**

impprefix(*impprefix*)
Specifies how to interpret a mount path name that does not have a path type prefix, where *immprefix* is one of the following:

NONE An explicit prefix must always be specified for an absolute path. Implicit prefix resolution is not valid in this case.

HFS If no explicit prefix is present, assume the path is a z/OS UNIX file system.

MVS If no explicit prefix is present, assume the path is an MVS high-level qualifier. This is the default.

HFS, MVS
If no explicit prefix is present, first assume the path is a z/OS UNIX file system. If no matching z/OS UNIX file system can be found, assume that it is an MVS high-level qualifier.

MVS, HFS
If no explicit prefix is present, first assume the path is an MVS high-level qualifier. If no matching high level qualifier can be found, assume that it is a z/OS UNIX file system.

Notes:

1. For the "MVS,HFS" setting, MVS selection requires that at least one MVS data set exists in the system catalog with the specified High Level Qualifier (HLQ).
2. The"MVS" setting, results in equivalent implicit prefix processing to releases prior to V1R11; that is, z/OS UNIX requires a prefix and MVS does not. However, an MVS prefix can still be specified on path names.
3. Since NFS v4 mount processing is performed one qualifier at a time, when 2 options are specified, the object existence test for determining whether to move on to the second option is based on the first path name qualifier only. If it exists, then the first option is selected and that cannot change if a later qualifier is not found.

leadswitch
Specifies that the server returns '/' as the first character in each export entry.

noleadswitch
Specifies that the server will not return '/' as the first character in each export entry.

The **leadswitch** attribute is ignored for z/OS UNIX file objects.

Table 24. Site attributes (continued)

Site Attribute **Description**

leasetime(*n*)

Specifies the length of time (the lease interval) in seconds that the z/OS NFS server allows clients to:

- Reclaim locks and share reservations following an NFS server restart. During this grace period, clients can reclaim locks on behalf of their users.
- Remain active without communicating with the NFS server. If an NFS V4 client does not communicate with the z/OS NFS server for the length of the lease interval, its client id will expire.

The value of *n* can range from 5 to 3600. The specified value must be smaller than the value of the **logout** attribute, if logout is not set to zero. The default value is **120**.
Note: When using the NFS version 4 protocol, the **leasetime** value should be set to a value larger than or equal to the attrtimeout, writetimeout and readtimeout attributes. Otherwise, it is possible for performance problems to occur when attempting to access MVS data sets.

logicalcache(*n*)

Specifies the maximum storage (in bytes) for allocated buffers in the logical I/O processing for all the cache windows combined.

The value of *n* is an integer from 1M to 2047M. The default value is **16M** . The recommended value for this attribute is 50% of the **bufhigh** attribute.

The **logicalcache** attribute does not apply to z/OS UNIX files.

logout(*n*)

Specifies the time limit for inactivity in seconds for a given user on a client. The default value is **1800**. When the limit is reached, the user is automatically logged out. The client user must enter the **mvslogin** command again to reestablish the client's z/OS session. This value should normally be the same as the value defined for TSO/E logout at your site. The value of *n* can range from 61 seconds to 20 megaseconds (approximately 243 days).

maxrdforszleft(*n*)

Specifies the number of physical block buffers left after determining a file's size. This operation is done for later server read requests to the same file. The buffers left are subject to trimming during a “buffer steal” operation. The value of *n* is an integer from 1 to 1024.

The default value is **32**.

maxtimeout(*n*)

Specifies the maximum timeout allowed. This attribute and the **mintimeout** attribute define the range of values that client users can specify for **attrtimeout**, **readtimeout**, and **writetimeout**. The value of *n* is the number of seconds from 1 to 32,767 (9 hours, 6 minutes, and 7 seconds). This attribute does not affect the **logout** attribute.

nomaxtimeout

Allows client users to specify **noattrtimeout**, **noreadtimeout**, and **nowritetimeout**.

Table 24. Site attributes (continued)

Site Attribute **Description**

mintasks(*n*,*m*,*o*)

Defines the minimum number of NFS tasks or threads allowed to run. Tasks may be terminated for reasons such as 80A or 878 ABENDs.

n Specifies the minimum number of subtasks which handle the asynchronous I/O operations or short blocking operations. If the number of active 'short' tasks becomes less than *n* the shutdown process of the NFS server starts.

m Specifies the minimum number of subtasks which handle z/OS UNIX file requests. If the number of active z/OS UNIX tasks becomes less than *m* the shutdown process of the NFS server starts.

o Specifies the minimum number of subtasks which handle long blocking operations. If the number of active legacy long service tasks becomes less than *o* the shutdown process of the NFS server starts.

If *n*, *m*, or *o* are greater than the corresponding values in **nfstasks**, they are assigned to **half** the **nfstasks** values. If *n*, *m*, or *o* are not specified, they are assigned default values of 4, 4 and 1, respectively.

Valid range for *n* is from 4 to 99

Valid range for *m* is from 4 to 100

Valid range for *o* is less than or equal to 99

Valid range for *n* + *o* is less than or equal to 100

mintimeout(*n*)

Specifies the minimum timeout. This attribute and **maxtimeout** define the range of values that can be specified for **attrtimeout**, **readtimeout**, and **writetimeout**. The value of *n* is the number of seconds from 1 to 32,767.

The default value is **1**.

mixcase/upcase

Specifies messages display in mixed or upper case.
Note: Starting with z/OS NFS V1R10, this attribute is ignored.

mvsprefix(*prefix*)

Specifies an MVS data set prefix to be appended to the front of MVS data set name for a mount path directory. The default value of *prefix* is **/mvs**.

The mvs prefix enables you to explicitly specify a prefix for identifying MVS data sets, simliar to the way in which the hfs prefix does for z/OS Unix files.

Table 24. Site attributes (continued)

Site Attribute	Description
mvssec(*krb5*,*krb5i*,*krb5p*,*sys***)**	Specifies the acceptable network transmission levels of security which can be used as the authentication flavor on NFS version 4 requests for accesses to MVS data sets. This attribute is only used when not overridden by authentication specifications in the exports file. Multiple values for this attribute can be specified using the comma as delimiter. The following are the supported values:. **krb5** Provides Kerberos V5 based integrity on the RPC credentials (but not data), when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 integrity algorithm and the RPCSEC_GSS service of rpc_gss_svc_none. **krb5i** Provides Kerberos V5 based integrity on both the RPC credentials and data, when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 integrity algorithm and the RPCSEC_GSS service of rpc_gss_svc_integrity. **krb5p** Provides Kerberos V5 based integrity and privacy on both the RPC credentials and data, when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 algorithm for integrity and 56 bit DES for privacy. The RPCSEC_GSS service used here is rpc_gss_svc_privacy. **sys** Specifies that the AUTH_SYS authentication flavor can also be used to access this data set. Note that the AUTH_SYS authentication flavor does not provide any integrity or privacy protection. The **mvssec** attribute default is **mvssec(sys,krb5,krb5i,krb5p)**. **Note:** File systems that require integrity or privacy protection over network transmissions of data should explicitly specify the desired settings. Do not rely on the default settings, because the default settings allow for RPC accesses using the AUTH_SYS authentication flavor, which does not provide any integrity or privacy protection.

Table 24. Site attributes (continued)

Site Attribute	Description
nfstasks(*n*,*m*,*o*,*t*,*u*)	Specifies the number of server processes to initiate on startup. If **nfstasks**(*n*,*m*) is specified, then the following is true. • The value of *n* is the number of subtasks that handle the asynchronous input/output (I/O) operations or short blocking operations (the maximum number of concurrent NFS server requests) in the z/OS MVS data path. • The value of *m* is the number of subtasks that handle the long blocking operations (the maximum number of concurrent NFS server recall and z/OS UNIX requests). Increase this value if your server supports lots of active recall or z/OS UNIX clients. Based on system resources available below the 16 Mb line, the maximum *n* value may not be achievable. The precise maximum value will be system configuration dependent. If an 80A or 878 Abend is experienced during NFS server startup, use a smaller value for *n*. If **nfstasks**(*n*,*m*,*o*) or **nfstasks**(*n*,*m*,*o*,*t*,*u*) is specified, then the following is true. • The value of *n* is the number of subtasks that handle the asynchronous input/output (I/O) operations or short blocking operations (the maximum number of concurrent NFS server requests) in the z/OS MVS data path. • The value of *m* is the number of subtasks that handle z/OS UNIX requests. Increase this value if your server supports lots of active z/OS UNIX requests. • The value of *o* is the number of subtasks that handle the long blocking operations (the maximum number of concurrent NFS server recall requests). Increase this value if your server supports lots of active recall operations. • The value of *t* is the number of transport subtasks that handle TCP network requests. • The value of *u* is the number of transport subtasks that handle UDP network requests. Based on system resources available below the 16 Mb line, the maximum *n* + *o* value may not be achievable. The precise maximum value will be system configuration dependent. If an 80A or 878 Abend is experienced during NFS server startup, use a smaller value for *n* + *o*. The following are valid value ranges for *n*, *m*, *o*, *t*, and *u*. • Valid range for *n* is from 4 to 99. • Valid range for *m* is from 4 to 100. • Valid range for *o* is from 1 to 99. • Valid range for *n* + *o* is from 5 to 100 • Valid range for *t* is from 4 to 32. • Valid range for *u* is from 4 to 32. The **nfstasks** attribute default is **nfstasks(8,16,8,4,4)**.

Table 24. Site attributes (continued)

Site Attribute **Description**

nfsv4domain(*NFSv4_default_domain*)
specifies the default domain for the NFS v4 protocol (NFSv4) name mapping.

The NFSV4DOMAIN attribute serves for redefinition of a name of this unique domain. In accordance with RFC3530 NFSv4 attributes "owner" and "owner_group"are transferred between the client and server in the form of "user_name@domain" and "group_name@domain". The server provides the mapping of names to ids and vice versa. *NFSv4_default_domain* identifies the user/group name space with one to one correspondence between the names and their numeric identifiers (uids and gids).

z/OS NFS Server will accept as valid a set of users and groups for default domain. The server will treat other domains as having no valid translations. If the NFSV4DOMAIN attribute is not used, the server uses the system-defined domain. The *NFSv4_default_domain* will be converted internally to lower case.

For further details on NFSv4 name mapping, see “NFS v4 protocol name mapping” on page 207.

nlm
Specifies that the initialization of the z/OS NFS server should include starting the NLM and NSM daemons.

nonlm
Specifies that the initialization of the z/OS NFS server should not include starting the NLM and NSM daemons. The system will run without lockd and statd. Specifying **nonlm** does not affect the availability of byte range locking and share reservation support for NFS version 4 protocol access.

If **nonlm** is specified, the NLM may not be started after NFS has initialized. If NLM is desired, you must stop and restart NFS after specifying the **nlm** site attribute. The only way to stop NLM is to shut down the NFS server. It is no longer necessary to define the NLM and NSM startup procedures to a z/OS UNIX segment as UID(0) to RACF because the NLM and NSM startup procedures are no longer supported.

Notes:

1. The lock data sets must always be allocated, even if **nonlm** is specified in the site attributes.
2. The old startup procedures for NLM and NSM are no longer shipped with z/OS; these procedures are obsolete and old copies from previous releases should not be used on z/OS V1R7 or later releases.

overflow
Specifies x37 detection support for PS/PDS MVS data sets for NFS Version 2, Version 3, and Version 4 WRITE operations. It allows ENOSPACE errors to be reported to the NFS Client in a timely manner and to avoid situations when the z/OS NFS Server closes a data set on timeout expiration basis with x37 abend which can not be propagated to the NFS Client. This option may also be activated on the MOUNT level. Default mode is no overflow detection, and is not displayed by the **showattr** client enabling command..
Note: PDSE and VSAM data sets are not supported.

pcnfsd
Specifies that z/OS NFS server is to start the PCNFSD server.

nopcnfsd
Specifies that z/OS NFS server is not to start the PCNFSD server.

Table 24. Site attributes (continued)

Site Attribute	Description
percentsteal(*n***)**	Specifies the percent of data buffers that can be reclaimed for use when the **bufhigh**(*n*) limit has been reached. A higher value means a reclaim operation is performed less often, but the cached data is significantly trimmed on each reclaim. This can result in poor read performance, because readahead buffers might be stolen. Lower values result in more frequent reclaim operations, but the cached data normal water mark is higher, meaning possibly better performance by reading from cached data. The value of *n* is an integer from 1 to 99. The **percentsteal** attribute default value is **20**.
public(*legacy_path***,***hfs_path***)**	Specifies the legacy path (MVS z/OS conventional data) and HFS path (z/OS UNIX data) that is associated with the public file handle for WebNFS access. The first path, if specified, is the legacy path. The second path is the HFS path. If the first path is not present, a comma must precede the second path. If the **public** keyword is specified, then one of the paths must be specified. The **public** keyword must be specified after the **hfsprefix(), mvsprefix(),** and **impprefix()** keywords in the site attribute table. A **lookup** request with the public file handle determines which of the two paths is being referenced by the pathname that is specified. An absolute pathname will tell the server which of the paths is referenced by matching one of the paths specified. A **lookup** request with a relative pathname will be interpreted as a z/OS UNIX request if HFS is active (hfs_path has been provided); otherwise, it is treated as a MVS request. The **public** attribute default value is **no public path**.

Table 24. Site attributes (continued)

Site Attribute	Description
pubsec(*krb5*,*krb5i*,*krb5p*,*sys***)**	Specifies the acceptable network transmission levels of security for accesses to public file systems which can be specified as the authentication flavor of the RPC request. This attribute is only used when not overridden by authentication specifications in the exports file. Multiple values for this attribute can be specified using the comma as delimiter. The following are the supported values:. **krb5** Provides Kerberos V5 based integrity on the RPC credentials (but not data), when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 integrity algorithm and the RPCSEC_GSS service of rpc_gss_svc_none. **krb5i** Provides Kerberos V5 based integrity on both the RPC credentials and data, when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 integrity algorithm and the RPCSEC_GSS service of rpc_gss_svc_integrity. **krb5p** Provides Kerberos V5 based integrity and privacy on both the RPC credentials and data, when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 algorithm for integrity and 56 bit DES for privacy. The RPCSEC_GSS service used here is rpc_gss_svc_privacy. **sys** Specifies that the AUTH_SYS authentication flavor can also be used to access file systems. Note that the AUTH_SYS authentication flavor does not provide any integrity or privacy protection. The **pubsec** attribute default is **pubsec(sys,krb5,krb5i,krb5p)**. **Note:** File systems that require integrity or privacy protection over network transmissions of data should explicitly specify the desired settings. Do not rely on the default settings, because the default settings allow for RPC accesses using the AUTH_SYS authentication flavor, which does not provide any integrity or privacy protection.
readaheadmax(*n***)**	Specifies the number of bytes to be read to fill internal buffers during read processing to enhance satisfying read requests directly from cache. This reduces the amount of synchronous physical I/O performed for NFS read requests for sequential read file access. It also reduces context switching overhead on NFS read requests by allowing more read requests to be satisfied directly from cache. The value of *n* is an integer from 1 KB to 128 KB (normally 2 to 4 times the common block size used for file access, which is recommended at 8 KB for AIX file activity). The **readaheadmax** attribute default value is **16K**. Specifying zero (0) will deactivate **readahead**.
readdirtimeout(*n***)**	Specifies the amount of time, in seconds, before the internal readdir cache that is used for MVS z/OS conventional data sets is timed out or discarded. The valid range is from 1 to 32,767 (9 hours, 6 minutes, and 7 seconds). The value of *n* can go as low as 1 second, but to avoid the possibility of *client hanging* (due to network delays and staled cache), *n* is not recommended to be lower than 5 seconds. The value of *n* may need to be increased if the network is slow and the accessed directory has a lot of entries. The **readdirtimeout** attribute default value is **30** seconds.

Table 24. Site attributes (continued)

Site Attribute **Description**

rec878
Specifies that the recovery processing of 878 and 80A ABENDs will be turned on, and affected tasks will attempt to recover.

norec878
Specifies that the recovery processing of 878 and 80A ABENDs will be turned off. That is, if this type of ABEND occurs, the server will shutdown without recovery. It should only be used for debug.

remount
When specified, the server processes NFS requests after the NFS Server is restarted even though the HFS file system was remounted with a new HFS file system number (device number) after its last usage. Use of the remount attribute causes the NFS Server to automatically access a remounted HFS file system even though it may have been changed prior to remounting. Any active client mounts are re-established.

noremount
When specified, the server fails NFS requests (with return value NFSERR_STALE) if the HFS file system was remounted with a new HFS file system number (device number) after its last usage.

The **readdirtimeout** attribute default value is **30** seconds.

The remount/noremount attributes apply only to HFS file systems.

restimeout(*n*,*m*)
Specifies a retention period and a clock time for the removal of mount points and control blocks that have been inactive longer than the specified retention period.

n Specifies the resource retention period for mounts and associated resources. If they have been inactive for more than *n* hours, they are removed.

The valid range for *n* is 0 to 720 hours (30 days). The default is **48** hours. If *n* is set to 0, the z/OS NFS server does not remove any mount points or associated resources.

m Specifies the time of day to do the cleanup for mounts and associated resources that have been inactive more than *n* hours. The time of day is specified as a 24 hour local time value.

The valid range for *m* is 0 to 23. The default is **0** (that is, midnight). Because cleanup work slows down the server, set *m* so that cleanup work occurs when the server is lightly loaded. If a mount handle is removed by the cleanup activity, the user must do the **umount** and **mount** operations to access the mount point again. The resource cleanup is also done when the server is shutting down.

Table 24. Site attributes (continued)

Site Attribute	Description

security(*mvs*[,*hfs*,*public*]**)**

Specifies security options for MVS data sets, z/OS UNIX files, and data that is accessed using the public file handle.

mvs Specifies the security option for MVS z/OS conventional data sets. The *mvs* parameter is a required parameter.

hfs Specifies the security option for z/OS UNIX files. The *hfs* parameter is an optional parameter.

public Specifies the security option for data that is accessed with the public file handle. The *public* parameter is an optional parameter.

Note: When the optional parameters (*hfs* and *public*) are not specified, they are assigned the same security option as the first parameter.

You can specify the following security options:

exports
Exports list checking. For z/OS UNIX files, checks UNIX permission bits. The UID is obtained from the client RPC request. No SAF checking.

none
Neither SAF checking nor exports list checking. For z/OS UNIX files, checks UNIX permission bits. The UID is obtained from the client RPC request.

saf
SAF checking. No exports checking. For z/OS UNIX files, checks UNIX permission bits. The UID is obtained from the z/OS UNIX segment using **mvslogin**. There is no transparent access across z/OS NFS server restart. An mvslogin is required

safexp
SAF checking and exports list checking. For z/OS UNIX files, checks UNIX permission bits. The UID is obtained from the z/OS UNIX segment using **mvslogin**. There is no transparent access across z/OS NFS server restart. An mvslogin is required

The **security** attribute default is **security(safexp,safexp, safexp)**.

Table 24. Site attributes (continued)

Site Attribute **Description**

setgid(POSIX | ZOSUNIX)

z/OS NFS Server uses POSIX rules in GID inheritance for new z/OS UNIX objects. If the S_ISGID bit of the parent directory is on, the new GID is set to the GID of the parent directory. Otherwise, it is set from the GID of the process. A new directory inherits the S_ISGID bit value of the parent directory.

POSIX

z/OS NFS Server uses POSIX rules in GID inheritance for new z/OS UNIX objects. If the S_ISGID bit of the parent directory is on, the new GID is set to the GID of the parent directory. Otherwise, it is set from the GID of the process. A new directory inherits the S_ISGID bit value of the parent directory.

This is the default value.

ZOSUNIX

z/OS NFS Server provides compatibility with z/OS UNIX. When the RACF profile FILE.GROUPOWNER.SETGID in the UNIXPRIV class is set, z/OS NFS Server uses POSIX rules, as stated above. Otherwise, a new GID is always set to the GID of the parent directory, and for a new directory, the S_ISGID bit is always set off.
Note: Some NFS clients (such as, SUN and AIX) force GID setting after object creation and prevent compatibility with z/OS UNIX even though the setgid(ZOSUNIX) attribute is set.

sfmax(*n*)

Specifies the maximum size (in kilobytes) of allocated storage for all of the side files. The value of *n* is an integer from 0 to 2000. The default value is **0** and it signifies that no mapping is allowed on the NFS server. If **sfmax** is set to **0**, specifying the **sidefile** keyword in the attributes data set will cause the server to shut down and specifying the **sidefile** keyword in any subsequent **mount** commands causes the mount to fail because mapping is not allowed on the NFS server. If the amount of storage specified cannot be obtained during server initialization then the server will shut down immediately.

Table 24. Site attributes (continued)

Site Attribute	Description

smf(*level*[,*switch*])

Specifies the level of SMF support and defines whether or not to start SMF record collection at the NFS server startup.

The following *level* options can be specified:

none
No SMF records are to be produced.

all
All SMF NFS type 42 records are to be produced.

userfile
Both user session and file usage SMF records are to be produced.

Alternately a list of levels (*subtype_list*) delimited by commas, can be specified. In this mode of specification, at least one of the subtype levels (user, file, audit) must be specified, and the remaining levels are optional.

file
Produces file usages SMF records (subtype 7).

user
Produces user session SMF records (subtype 8).

audit
Produces file creation, removal, and rename records (subtype 26).

The following *switch* options can be specified:

off
Activation of SMF records collection can be done manually by issuing the **modify** command. The *switch* parameter is optional.

on
Activates SMF records collection at the NFS server startup.

The full syntax of the **smf** attribute follows:

```
smf(none|all||userfile | subtype_list[,on|off])
```

An example of the **smf** attribute follows:

```
smf(user,on)
```

An example follows that shows the **smf** attribute when the value of *switch* is **off**:

```
smf(user)
```

stringprep

Specifies that z/OS NFS server is to enable stringprep normalization. Stringprep normalization is the NFS version 4 globalization function for converting inbound strings to UTF-8 format.

nostringprep

Specifies that z/OS NFS server is to not enable stringprep normalization.

Part 2. Customization and Operations

Chapter 10. Customization 153
- Protecting your programs and files 153
  - Protecting the server control files 153
  - Setting up the z/OS NFS authorization 153
  - Protecting the file system on z/OS with the NFS V4 protocol 154
    - GSS credential acquisition 155
    - Security context acceptance 156
    - Security negotiation 156
    - NFS4ERR_WRONGSEC handling. 158
  - Protecting the file system on z/OS with the Security site attribute 158
    - Unrestricted data access–security(none). . . 158
    - Exports list checking–security(exports) . . . 159
    - SAF checking–security(saf) 160
    - SAF and exports list checking–security(safexp) 160
    - SAF checking with checklist processing. . . 161
    - File system export 162
    - Authorization of file operations 163
  - Customizing installation security exits 163
  - Using UNIX style credentials for authentication 163
- Converting data 164
- Creating the conversion environment for Unicode Services 164
- Collecting NFS usage data 165
- Configuring the z/OS NFS client 167
  - Creating the PARMLIB statement for the client 167
  - Updating z/OS system data sets for the client 167
    - PARMLIB updates. 167
    - PROCLIB updates 168
  - Allocating client log data sets 168
  - NFS Client with Multiple TCPIP stacks. . . . 168
  - Mounting remote file systems 168
  - Setting up reserved (privileged) ports 169
  - Configuring a secure z/OS NFS client 169
- Configuring the z/OS NFS server 171
  - Attributes data set. 171
  - Exports data set 172
    - When the modified exports data set takes effect 172
    - Directory statement 173
    - Client id specification 179
  - Checklist data set 181
  - Mount handle data sets 182
  - Lock data sets 182
  - Converting data between ASCII and EBCDIC - NFS V2 and V3 only 183
    - Customizing the translation table. 183
    - Enabling the xlat processing attribute . . . 184
  - Updating z/OS system data sets for the server 185
  - Allocating the z/OS NFS server log data sets 185
  - Side file data set 185
  - Modifying tcpip.ETC.RPC and etc/rpc 185
  - Setting up a user-specified port range 186
  - Configuring a secure z/OS NFS server 188
  - Using dynamic client IP addressing 191
  - Terminal ID based restricted MVSLOGIN . . . 191
  - SERVAUTH based restricted MVSLOGIN . . . 192
  - Data Labeling 194
  - Using multiple TCP/IP stacks 195
    - Configuring multiple NFS servers with multiple TCP/IP stacks 195
    - Configuring a single NFS server with multiple TCP/IP stacks 197
- Installing the client enabling commands 198
  - Retrieving commands for AIX, Sun Solaris, and Linux 200
  - Porting the mvslogin, mvslogout, and showattr commands 203
  - Porting on different compilers and operating systems 204
    - Compiling 204
    - Linking 206
    - Running commands 206
- NFS v4 protocol name mapping 207
  - Inbound owner/owner_group processing . . . 208
  - Outbound owner and owner_group attributes processing 209
  - Domain processing in NFSv4 Name Mapping 210

Chapter 11. Network File System operation . . 211
- Starting the z/OS NFS client 211
- Stopping the z/OS NFS client 212
- Starting component tracing for the z/OS NFS client 212
- Starting the z/OS NFS server 215
  - Starting multiple servers 217
- Stopping the z/OS NFS server 217
- Starting the z/OS NFS NSM and z/OS NFS NLM 218
- Starting component tracing for the z/OS NFS server 218
- Entering operands of the modify command for the z/OS NFS server 222
  - Addds operand. 225
  - bufferusage operand 225
  - bufhigh operand 226
  - Cbsniff operand 226
  - DlyDTimeout operand 226
  - Exportfs operand 227
  - Freeds operand. 227
  - Freeze operand. 228
  - List operand. 228
  - Mapfile operand 229
  - racf_cache_reset operand 229
  - Release operand 230
  - smf operand. 231
  - Status operand 231
  - Swapldb operand 231
  - Swapmhdb operand 231
  - Unmount operand. 232
  - Unmntall operand. 232
  - Unmnthfs operand 232

© Copyright IBM Corp. 1991, 2010

Unmntmvs operand 233
v4delg operand. 233
Entering operands of the modify command for diagnosis 233
Debug operand. 233
Flushlog operand 234
Listlock operand 234
Log operand 235
Smf operand 236
Sockhang operand. 236
Switchlog operand 237
Version operand 237
Displaying NFS trace information 238

Chapter 12. Installation-wide exit routines for the z/OS NFS server 239
Requirements for NFS 239
Sample link-edit JCL 240
Storage blocks of the server exits 240
Global exit block (GXB) 240
User exit block (UXB) 241
Login installation-wide exit. 241
Requirements of the login exit. 243
Options of the login exit. 243
Structure of the login exit message 243
Contents of the login exit parameter list . . . 243
Login exit parameter list. 244
Request codes to the login exit 244
Return codes from the login exit 245
System initialization routine of the login exit 245
Start of new user session routine of the login exit 245
User login request routine of the login exit . . 246
User logout request routine of the login exit . . 247
System termination routine of the login exit . . 247
File security installation-wide exit 247
Requirements of the file security exit 250
Structure of the file security exit message . . . 250
Contents of the file security exit parameter list 250
File security exit parameter list 252
Request codes to the file security exit 252
Return codes from the file security exit. . . . 252
Validate allocate request routine of the file security exit 252
Validate write request routine of the file security exit 253
Validate read request routine of the file security exit 253
Return security permissions routine of the file security exit 254

Chapter 10. Customization

This topic describes how to configure NFS and how to make it available to users. You can perform these tasks to customize NFS:

Table 25. Customizing NFS

Section	Page
Protecting your programs and files	153
Converting data	164
Creating the conversion environment for Unicode Services	164
"Using multiple TCP/IP stacks" on page 195	195
Collecting NFS usage data	165
Configuring the z/OS NFS client	167
Configuring the z/OS NFS server	171
Installing the client enabling commands	198

Protecting your programs and files

This section describes security measures that you should take to protect your programs and files in preparation for customizing NFS. These security measures help you protect the server control files, the NFS server and client installations, and the MVS file system. You can customize the NFS security processing and use UNIX style credentials to verify the identity of a client system.

Protecting the server control files

You should protect the following server control data sets from unauthorized access with RACF, a component of the Security Server for z/OS.

- Attributes file (read by the server at initialization)
- Exports file (read by the server)
- Mount handle data set (read and updated by the server)
- Checklist data set (read by the server)
- Lock data sets (read and updated by the server)
- Kerberos configuration file (read by the server)
- Kerberos keytab (read by the server)

Setting up the z/OS NFS authorization

The following security measures should be addressed when you install the z/OS NFS server and client:

- All programs that come with the z/OS NFS server and client must reside in an APF-authorized program library.
- You need to define the z/OS NFS server and client to resource access control facility (RACF) and assign the necessary level of authority. You do this by defining a RACF user ID for the z/OS NFS server and client. Because the z/OS NFS server and client are run as started tasks, you also need to define an entry in the RACF-started procedures table which associates the z/OS NFS server and client startup procedure names with the previously defined user IDs. For more

© Copyright IBM Corp. 1991, 2010

information about coding and replacing the RACF-started procedure table, see *z/OS Security Server RACF Security Administrator's Guide* and *z/OS Security Server RACF System Programmer's Guide*.

The z/OS NFS server can now be set up with the **trusted** attribute as follows:

```
SETROPTS GENERIC(STARTED)
RDEFINE STARTED mvsnfs.* STDATA(USER(mvsnfs) GROUP(SYS1) TRUSTED(YES))
SETR CLASSACT(STARTED)
SETR RACLIST(STARTED)
ADDUSER mvsnfs OMVS(UID(0))
```

With trusted authority, the NFS server can perform the following tasks:

- Reconstruct the mount points (from the active mount handle data set) upon startup
- Handle mount requests from client prior to user login
- Handle **ls** or **nfsdir** list commands prior to user login
- Be a trusted user during normal operation

During actual remote client file access, the z/OS NFS server first RACROUTEs the remote client's user ID to determine if the remote client is authorized to access the file system. If the remote client is authorized, the z/OS NFS server switches to its own user ID, which has trusted authority, to access the file system.

- You need to define a z/OS UNIX segment for the z/OS NFS client in the RACF user profile as UID(0).

For TCP/IP security information, see *z/OS Communications Server: IP Configuration Guide*.

For z/OS UNIX security information, see *z/OS UNIX System Services Planning*.

Protecting the file system on z/OS with the NFS V4 protocol

The NFS version 4 protocol improves on the NFS version 2 and 3 protocols with stronger authentication and network transmission protection for NFS data. The NFS V4 protections include encryption algorithms for data privacy, multiple protections per file, and the means to negotiate security as NFS clients explore the file system. The NFS V4 protocol provides these protections through the required RPCSEC_GSS security authentication flavor and the SECINFO operation.

The z/OS NFS server enforces these protections, for client RPC requests that use RPCSEC_GSS, through the site attributes **mvssec**, **hfssec**, and **pubsec**. These attributes provide the Kerberos V5 Security Mechanism (RFC1964) subset of the V4 protocol, at the file-system level. The z/OS NFS server also continues to support NFS V2 and V3, as well as V4 protocol requests with the protections provided by the **security** site attribute, as described in "Protecting the file system on z/OS with the Security site attribute" on page 158.

The export data set also contains a security keyword, **sec**, that specifies the Kerberos authentication level that clients must have to access individual files and data sets on the z/OS NFS server. That is, specific export entries can be further constrained with different authentication flavors by using this security keyword. For example, an important export entry can be protected with **krb5p** level set by the security keyword, while other exported entries in the file system can be accessed by all authentication levels which are specified by the **mvssec**, **hfssec** and **pubsec** site attributes. The authentication flavors specified by the **sec** keyword in the export entries should be a subset of the authentication flavors of site attributes **mvssec**, **hfssec** and **pubsec**. In other words, an authentication level is effective only if it is specified by the site attribute logically AND'ed with the security keyword. For this reason, if the **sec** keyword is not specified, meaning all flavors are on, the authentication level is defaulted to the site attribute **mvssec**, **hfssec** and **pubsec**. For more information on the export security **sec** keyword, see "Exports data set" on page 172.

The **mvssec**, **hfssec**, and **pubsec** attributes let you specify the default network security flavors, and order, that can be used by requests accessing MVS, z/OS UNIX, and public file systems, respectively. These attributes only apply when the RPCSEC_GSS security mechanism is being used for communicating with the NFS client. These site attributes apply to all NFS versions. If you protect a data set with any one of these transmission attributes, NFS V2/V3 requests will get responses of AUTH TOOWEAK unless 'sys' is listed as a valid authentication flavor. NFS V4 requests that do not comply with these protections will get WRONG SEC. For more information on these site attributes, see "Site attributes syntax" on page 134.

Note: Since z/OS NFS only supports RPCSEC_GSS security for NFS version 4, if one of the site attributes is set to require RPCSEC_GSS, then clients using NFS versions 2 and 3, which only support AUTH_SYS security, cannot access those file systems. On the other hand, if the attribute is specified at a mount point, then only that mount point will be affected.

To use the NFS V4 RPCSEC_GSS security flavors, the following changes to the security infrastructure are required:

- Kerberos services must be activated on the z/OS system where the NFS server is running. This activation includes the definition of Realms, Inter-Realm relationships, and the Kerberos Principal for the z/OS NFS server. For details on these definitions, see the *z/OS Integrated Security Services Network Authentication Service Administration*.
- The Kerberos principals on NFS clients need to be defined to RACF and assigned a RACF identity. In addition, for Linux clients a principal nfs/hostname.domain should be defined to RACF. This is because Linux Clients use this principal for mounts and some state operations. For further details on defining principals, see *z/OS Security Server RACF Security Administrator's Guide*.
- When acquiring Kerberos tickets, NFS clients must use MD5 Checksum with DES encryption.
- No UNIX style user ID checking will be performed. Clients will always be validated to check for authorization based on their GSS credentials.
- The z/OS NFS server must have READ access to the IRR.RUSERMAP resource in the FACILITY class.

Note: For more information on setting up the z/OS NFS server with RPCSEC_GSS security, see "Configuring a secure z/OS NFS server" on page 188.

GSS credential acquisition

GSS credentials enable the communicating applications to establish security contexts with each other. They can contain multiple cryptographic keys that are required for authentication and message encryption to be performed with different algorithms. The z/OS NFS server uses Kerberos V5 as its security mechanism for acquiring the GSS credentials. The z/OS NFS server initially acquires these credentials during server startup. The z/OS NFS server uses the credentials for accepting the security context requests from NFS clients, and the same credentials may be used for initiating security contexts during RPC callbacks. The Kerberos principal for the z/OS NFS server must be defined in the Kerberos key table identified by the KRB5_KTNAME environment variable.

Note: For more information on setting up the z/OS NFS server with RPCSEC_GSS security, see "Configuring a secure z/OS NFS server" on page 188.

The z/OS NFS server will attempt to acquire the GSS credentials for the maximum credential lifetime but the actual lifetime of credentials will depend on the lifetime of the underlying Ticket Granting Ticket of the Kerberos Security Server, and is not controlled or governed by the z/OS NFS server. On expiration of the server's GSS credentials, client requests will receive the RPCSEC_GSS documented errors and the client is expected to refresh the contexts and retry the requests.

Security context acceptance

A security context is a data structure that contains information about the cryptographic state of a program on the client communicating with the server, and is required for RPC message security services. NFS clients create security contexts with the z/OS NFS server as part of the RPCSEC_GSS protocol of data flow. The z/OS NFS server accepts security context requests subject to the following restrictions and recommendations:

1. The z/OS NFS server does not support channel bindings.
2. The z/OS NFS server never initiates any requests as an agent of NFS clients and therefore recommends that clients do not use credential delegation services while creating security contexts.
3. The z/OS NFS server does not support the Out Of Sequence Detection services of GSS API. It expects NFS clients to have the seq_req_flag parameter turned off on their calls to GSS API gss_init_sec_context.
4. The z/OS NFS server recommends that the clients do not use the Message Replay services of the GSS API. It expects NFS clients to have the replay_det_req_flag turned off on their calls to the GSS API gss_init_sec_context. Note that the z/OS NFS server's implementation of the RPCSEC protocol provides for the protection against replay attacks.
5. The z/OS NFS server does not allow clients to authenticate as anonymous principals.
6. The z/OS NFS server recommends that NFS clients use mutual authentication services during context creation. The z/OS NFS server will still honor context creation requests from NFS clients that are unable to, or choose not to, use mutual authentication services in the GSS-API. However, clients that would require RPC callbacks from the z/OS NFS server have to support accepting security contexts with mutual authentication, because the z/OS NFS server always initiates security contexts with mutual authentication services.

Security negotiation

The NFS version 4 protocol facilitates the use of multiple RPC authentication flavors. The z/OS NFS server supports the Kerberos V5 security mechanism and all the pseudo flavors of the Kerberos security mechanism using the cryptographic algorithms referred to in NFS V4 (RFC3530). To facilitate selection of a particular pseudo flavor, the z/OS NFS server supports security negotiation using the NFS V4 protocol's SECINFO operation. IBM strongly recommends that security negotiation be done by the NFS clients using the SECINFO operation with an RPC authentication flavor of RPCSEC_GSS with the krb5i or krb5p pseudo security flavors.

When responding to SECINFO for security negotiation (when multiple security flavors are present for a file system or file), the z/OS NFS server uses an order of preference that has RPCSEC_GSS as the most favored flavor followed by AUTH_SYS. For the authentication flavor of RPCSEC_GSS, the z/OS NFS server has krb5, krb5i, and krb5p as its listed pseudo flavors in descending order of preference. NFS clients are, however, free to choose from any one of the z/OS NFS server-supported security flavors for their NFS V4 requests.

NFS V4 clients that decide to use the AUTH_SYS flavor may still have to do an **mvslogin** like their V2/V3 counterparts, depending on the settings of the **security** site attribute.

Note: Because the NFS version 4 protocol does not use the mount procedure, a mount command from an NFS version 4 client will be translated into a PUTROOTFH operation and a series of lookup operations for each component of the mount pathname. A z/OS NFS server cannot accurately ascertain where the mount command will be completed, so both SAF and EXPORT checking are disabled for the z/OS NFS version 4 lookup operation, unless processing attribute **mvsmnt** is specified, making name space navigation available without **mvslogin** (as SAF would otherwise require, and EXPORT would otherwise prematurely prevent until an export entry was matched). When non-navigation operations are received by the z/OS NFS server, the proper SAF and EXPORT checking will assure that proper access is given. Due to these limitations, it is highly recommended that the **mvsmnt** attribute be used with SAF or SAFEXP enabled for a z/OS NFS version 4 server. Also, if the **mvsmnt** attribute is not used for a SAF or SAFEXP z/OS NFS version 4 server, security verification issues may arise when a z/OS UNIX file system is being accessed because the z/OS NFS server may use the UNIX permission bits instead of allowing RACF to perform its job as desired.

Security Negotiation using the SECINFO operation is performed by the z/OS NFS Client in the following instances:

- During mount point establishment
- During NFS4ERR_WRONGSEC handling

Mount point establishment: When a security flavor is not specified for an NFS V4 mount, the z/OS NFS client queries the supported security flavors on the NFS server using the SECINFO operation. The z/OS NFS client chooses a security flavor returned by the server based on the following order of preference:

sys System Authentication

krb5 Kerberos V5 security providing integrity protection on the RPC header

krb5i Kerberos V5 security providing integrity protection on the RPC header and the RPC data

krb5p Kerberos V5 security providing encryption protection on the RPC data and integrity protection on the RPC header

The chosen security flavor serves as the designated flavor for the NFS client for all future accesses to this mount point.

The following should be noted with regards to Security Negotiation during mount point establishment:

1. Security Negotiation during mount is not done when a security flavor is specified in the secure keyword.
2. During mount when datacaching is specified and a security negotiation was attempted, Datacaching is turned off.

NFS4ERR_WRONGSEC handling

NFS servers fail an NFS request with NFS4ERR_WRONGSEC if the security policy on the mount point at the server's end does not allow the authentication flavor with which the request was issued. After receiving this error, the z/OS NFS client negotiates security with the NFS server by issuing a SECINFO operation to query the server-supported security flavors. The z/OS NFS client chooses a security flavor from the server supported flavors based on the order of preference specified in “Mount point establishment” on page 157 and retries the failing request with this newly-chosen security flavor. This security flavor serves as the designated security flavor for all future accesses to that mount point.

The following should be noted with regards to Security Negotiation during NFS4ERR_WRONGSEC handling:

1. On existing mount points and objects, the client only negotiates security when it is an upgrade to a more secure flavor. The order of flavors in the descending order of security that they provide is as follows:
   - krb5p
   - krb5i
   - krb5
   - sys
2. For existing files, Security Negotiation is not done when datacaching is on for that file.

Protecting the file system on z/OS with the Security site attribute

You can use the **security** site attribute, with the NFS V2, V3, and V4 protocols, to select the level of protection for different types of data access. A different protection level can be specified for MVS data sets, HFS files, and data that is accessed using the public file handle. The attribute used to protect data access is the **security** attribute. The format of the keyword is **security(mvs[,hfs,public])**. The following are the security options: **exports**, **none**, **saf**, and **safexp**. See “Site attributes syntax” on page 134 for syntax rules.) The z/OS NFS server can be configured to handle security in the following ways:

- None
- Exports list checking
- System Authorization Facility (SAF) checking
- Customized installation security exit
- System Authorization Facility (SAF) checking with checklist processing (to bypass SAF for files and directories under selected mount points)
- A combination of these approaches

Note: The UNIX permission checking against the z/OS UNIX hierarchical file system might appear to be inconsistent if the definitions of UID, GID, and SGID are not consistent throughout the domain of the network.

Unrestricted data access–security(none)

If you do not want to restrict data access, you can use the **security(none)** attribute. Neither exports list checking nor SAF checking is done. Client users can access z/OS files without a z/OS user ID and without using the **mvslogin** command. They simply mount the z/OS file systems that they want to access and unmount when they are finished. For z/OS UNIX files, the UNIX permission bits are checked before access is granted to the client user. See Figure 17 and Table 26 on page 162 for information on permission checking.

Note: If UID or GID from the RPC request is zero, it will be mapped to 65534 (nobody) before the UNIX permission checking is performed.

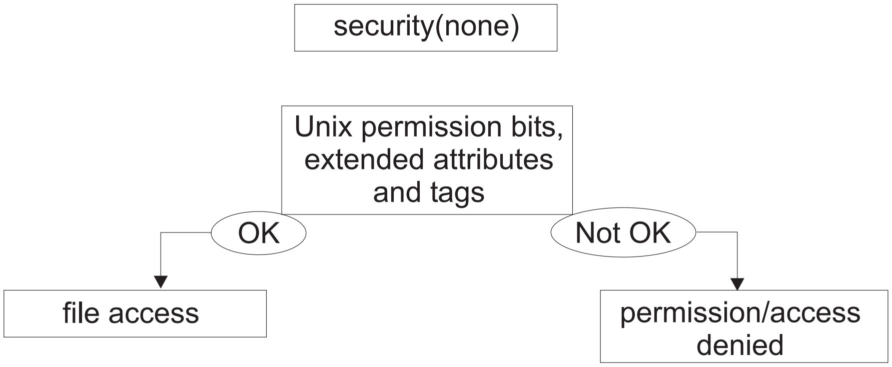


Figure 17. Permission checking for the security(none) attribute

Exports list checking–security(exports)

When you specify **security(exports)** in the attributes data set, the NFS server checks the client IP address against the exports list, which is generated from the exports data set, to determine whether or not a mount is to be granted. The NFS server also checks the requested directory (or high-level qualifier) to be mounted. For z/OS UNIX data, it also checks the UNIX permission bits before granting file access to a client user. See Figure 18 and Table 26 on page 162 for information on permission checking.

Note: If UID or GID from the RPC request is zero, it will be mapped to 65534 (nobody) before the UNIX permission is performed.

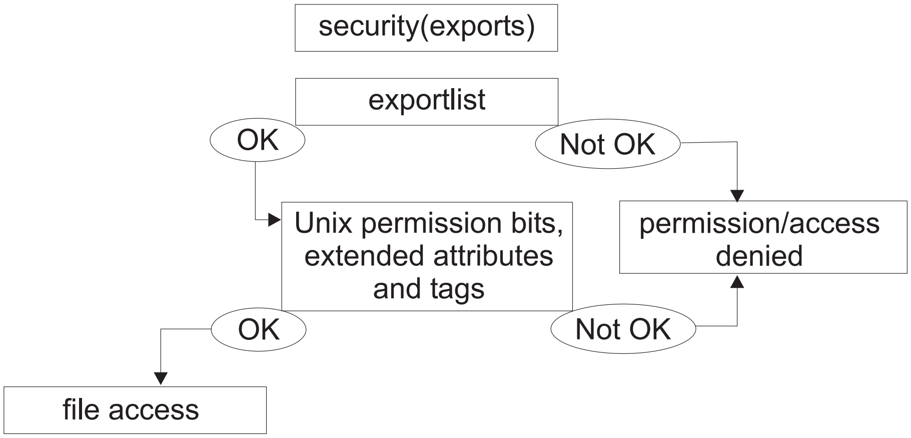


Figure 18. Permission checking for the security(exports) attribute

For more information about the *exports* data set, see “Exports data set” on page 172.

SAF checking–security(saf)

When you specify **security(saf)** in the attributes data set, the NFS server uses RACF or an equivalent product to control access to z/OS file systems. All RACF requests from the server are made through SAF. SAF directs control to RACF, or an equivalent security product, if it is active.

The server uses SAF to validate the z/OS user id and password supplied by the client user. It also uses SAF to validate that the client user is allowed to access z/OS data. A RACF user ID must be defined for each client user that requires access to the server.

For z/OS UNIX data, z/OS UNIX checks the UNIX permission bits, or ACLs, before granting file access to a client user. See Figure 19 and Table 26 on page 162 for information on permission checking. For users accessing z/OS UNIX, their RACF user ID must have an z/OS UNIX segment defined in the RACF profile.

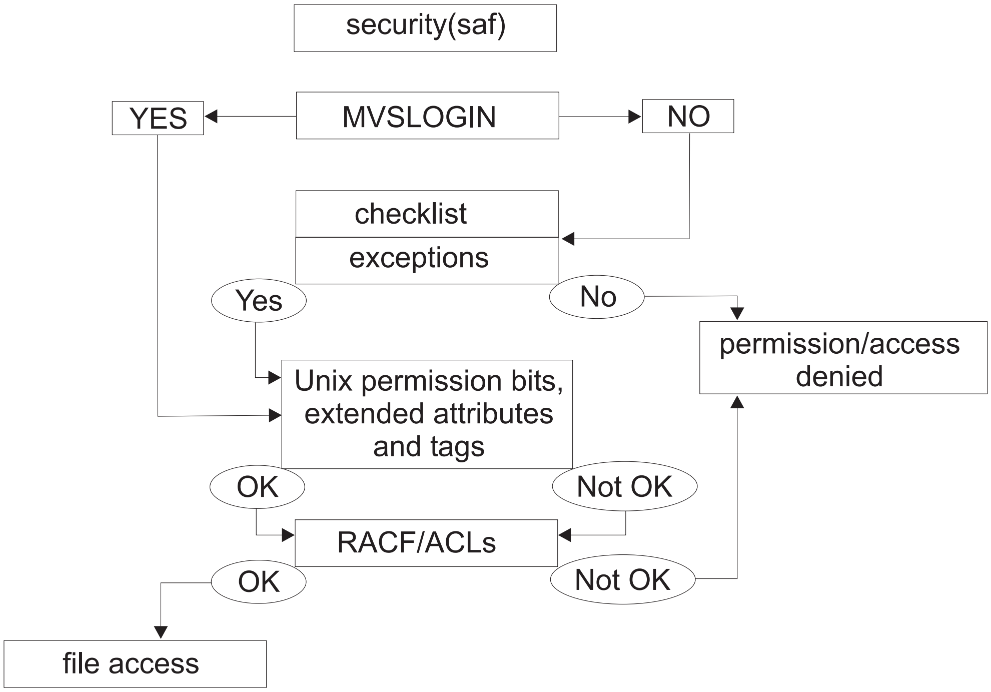


Figure 19. Permission checking for the security(saf) attribute

SAF and exports list checking–security(safexp)

When you specify **security(safexp)** in the attributes data set, the NFS server checks for both RACF authorization and exports list authorization before granting a client user access to z/OS data. For z/OS UNIX data, z/OS UNIX checks the UNIX permission bits, or ACLs, before granting file access to a client user. See Figure 20 on page 161 and Table 26 on page 162 for information on permission checking. This is the most restrictive means of limiting file system access. It requires client users to use the **mvslogin** command.

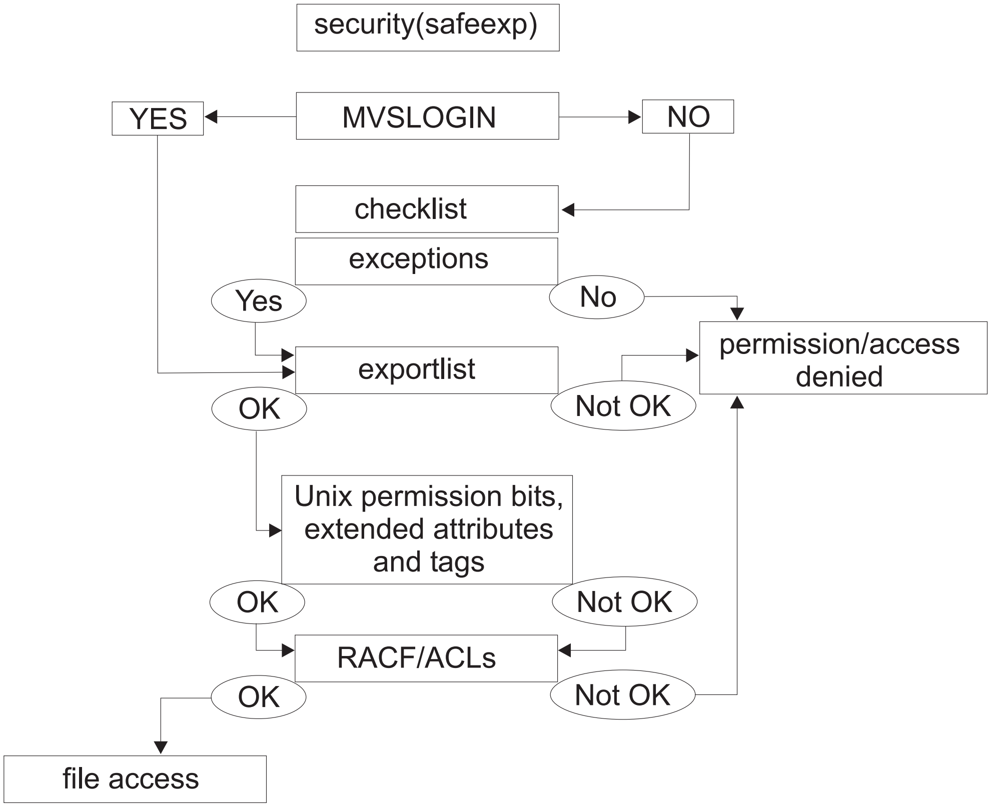


Figure 20. Permission checking for the security(safexp) attribute

For more information about the exports data set, see “Exports data set” on page 172.

SAF checking with checklist processing

When you specify **security(saf)** or **security(safexp)** with the **checklist** attribute, the NFS server performs SAF as described in “SAF and exports list checking–security(safexp)” on page 160. The only exception to this is that it will not check the files and directories that are underneath the mount points that either match the mount point or the children of the mount points that are specified in the exports data set using the **dirsuf** parameter. For more information, see “Exports data set” on page 172.

Table 26. UID, GID, SGID permission checking with security site attribute

Client			z/OS			Comments
uid	gid	sgids	uid	gid	sgid	
uid_a	*gid_a*	*sgid_b,...*	N/A	N/A	N/A	When security(none) or security(export) is used the z/OS NFS server checks the object owner uid, owner_group gid and its permission bits against the client uid (uid_a), gid (gid_a), and supplemental gids (sgid_b,...) in the RPC AUTH_SYS authentication. **Advantages:** Performance may be improved. **Disadvantages:** 1. The RPC AUTH_SYS allows only a maximum of 16 supplemental gids. 2. Spoofing of client uids, gids, and sgids cannot be prevented. 3. The object in the underlying physical file system (zFS or HFS) could have extended ACL entries, but the z/OS NFS server does not check
uid_a	*gid_a*	*sgid_b,..*	*uid_A*	*gid_A*	*sgid_B,..*	When security(saf) or security(safexp) is used the z/OS NFS server defers to the underlying physical file system (zFS or HFS) and RACF to check the object owner uid, owner_group gid, and its permission bits against the mapped Client-z/OS uid (uid_A), gid (gid_A), and supplemental gids (sgid_B,...) from RACF User's Definition **Notes:** 1. mvslogin establishes the mapped Client to z/OS uid_A, gid_A, sgid_B,.... 2. The RPC AUTH_SYS authentication (uid_a) is used to find the mapped Client to z/OS segment (uid_A, gid_A, sgid_B,...). **Advantages:** 1. The RPC AUTH_SYS limitation of maximum 16 supplemental gids is eliminated because of the mapped Client to z/OS supplemental gids (sgid_B,...). 2. Spoofing of uid/gids/sgids is prevented by mvslogin 3. Exploits the underlying physical file system ACL support **Disadvantages:** Performance may be impaired.

File system export

A system administrator issues the **mount** command to an NFS server and makes a remote file system available to the user. The z/OS server keeps a list of file systems and associated access restrictions in an export file. It then compares incoming mount requests to the entries in the file. If a match is found in the export file and the client is authorized for access, then the file system is successfully mounted.

Table 27 shows server processing of a mount request.

Table 27. z/OS server processing of a mount request

Security Option	Export File	z/OS UNIX File	MVS Data Set
none	Not required	No checking exported	No checking exported
saf	Not required	No checking exported	No checking exported

Table 27. z/OS server processing of a mount request (continued)

Security Option	Export File	z/OS UNIX File	MVS Data Set
exports	Required	Checking export file	Checking export file
safexp	Required	Checking export file	Checking export file
Note: MVSLOGIN is not required for NFS mount request.			

Authorization of file operations

After the file system is mounted, the user performs the normal file operations on the NFS-mounted remote file system. z/OS NFS server adds the z/OS SAF checking in addition to the UNIX file permissions check.

Note: MVS z/OS conventional data sets do not support UNIX permission bits in the file attribute structure. By disabling the SAF security, there is no authorization checking for file operation to MVS z/OS conventional data set. The UNIX permission bits checking is still performed for z/OS UNIX file operations when the SAF security is disabled.

Table 28 shows server processing of a file request.

Table 28. z/OS server processing of a file request

Security Option	MVSLOGIN	z/OS UNIX File	MVS Data Set
none	Not required	Check file permission bits	No checking
saf	Not Required***	SAF check***	SAF check***
exports	Not required	Check file permission bits	No checking
safexp	Not Required***	SAF check***	SAF check***
Note: z/OS UNIX segment must be defined for z/OS file operation. (***This does not apply when checklist requirements are satisfied.)			

Customizing installation security exits

You can write installation-wide exits or replaceable modules that customize Network File System security processing, by using product-sensitive programming interfaces provided by the server. Depending on how you code the exit, client users could be required to use the **mvslogin** command even for the **security(none)** and **security(exports)** attributes.

For more information about customizing your installation's security exits see “Login installation-wide exit” on page 241 and “File security installation-wide exit” on page 247.

Using UNIX style credentials for authentication

Authentication is the process of verifying the identity of a client system. This ensures that one client system cannot masquerade as another client system (perhaps with a different set of privileges). Client systems are identified by a set of credentials and authenticated with verification information passed in messages sent to server systems. There are several different conventions for exchanging authentication information in the NFS protocol, including these credentials:

- Null
- UNIX style
- DES-style

- Other, user written

The z/OS NFS server supports the System Authentication flavor of the RPC protocol that employs the UNIX style credentials for all supported NFS protocol versions. For the NFS version 4 protocol, the z/OS NFS server also supports the RPCSEC_GSS authentication flavor, which employs GSS credentials. For its RPCSEC_GSS authentication support, the z/OS NFS server only supports the Kerberos V5 security mechanism.

The z/OS NFS client utilizes z/OS UNIX-socket-enabled RPCs to communicate with remote z/OS NFS servers over a TCP/IP network. The credential includes the user ID (UID), group ID (GID), and a list of GIDs to which the user belongs. z/OS NFS supports all GID groups specified in the GID group list, which extends support beyond the 16 GID group restriction of the UNIX style AUTH_SYS authentication flavor. As of V1R11 the z/OS NFS Client also supports the RPCSEC_GSS authentication flavor.

Converting data

The z/OS NFS client supports data conversion defined by the universal character encoding standard known as the Unicode Standard on z/OS V1R2 (and above) when reading data from a remote NFS server or writing data to a remote z/OS NFS server. The Unicode Standard offers character conversion as well as basic case conversion. Within character conversion, characters are converted from one coded character set identifier (CCSID) to another. CCSID information is obtained from the **cln_ccsid** and **srv_ccsid** parameters.

Only single byte to single byte data conversion is supported. For example, if a client file has a CCSID of 437 and a server file has a CCSID of 297, data conversion will occur between USA ASCII format (CCSID 437) and French EBCDIC format (CCSID 297). Single byte to multiple byte conversion (including double byte character set (DBCS)) is not supported and will result in NFS terminating with an error message. NFS version 4 protocol (NFSv4) differs from NFSv2 and NFSv3 protocol in handling single to multiple byte conversion. Therefore, the technique-search-order specified in the **convserv()** attribute should consider the effects of the NFS protocol being used. See“Creating the conversion environment for Unicode Services” for further details.

The **cln_ccsid**, **srv_ccsid**, **xlat**, **tag/notag**, and **convserv** attributes identify whether data conversion is performed, and how data conversion is done. These parameters are supported by the z/OS NFS client installation parameter and TSO MOUNT command. The parameters on a TSO MOUNT command override the parameters specified as a z/OS NFS client installation parameter.

Creating the conversion environment for Unicode Services

The z/OS client or server uses Unicode Services to support data conversion on files in either EBCDIC or ASCII formats as well as other data formats that are defined with a CCSID. No setup is needed to begin using Unicode Services. As of Version 1 Release 7, z/OS ships with Unicode Services ready to use. See *z/OS Unicode Services User's Guide and Reference* (previously titled *z/OS Support for Unicode: Using Unicode Services*) for more information .

With the NFS version 4 protocol (NFSv4), metadata are transferred between the server and client in the UTF-8 data format (ASCII text is not transferred directly). The CCSID used by Unicode Services for UTF-8 is 1208.

The **convserv** attribute defines how data conversion is performed between CCSIDs by specifying the conversion technique-search-order which Unicode Services will use for specified srv_ccsid(x) and cln_ccsid(x) code pages. Technique consists of up to five technique-characters corresponding to the available techniques (R, E, C, L and M) used to define the technique search order for Unicode Services to process the specified code pages.

The technique-characters, with description, are defined as follows. See *z/OS Unicode Services User's Guide and Reference* (previously titled *z/OS Support for Unicode: Using Unicode Services*) for more information.

R Roundtrip conversion

Roundtrip conversions between two CCSIDs assure that all characters making the "roundtrip" arrive as they were originally.

E Enforced Subset conversion

Enforced Subset conversions map only those characters from one CCSID to another that have a corresponding character in the second CCSID. All other characters are replaced by a substitution character.

C Customized conversion

Customized conversions use conversion tables that have been created to address some special requirements.

L Language Environment-Behavior conversion

Language Environment-Behavior conversions use tables that map characters like the *iconv()* function of the Language Environment Runtime library.

M Modified Language Environment-Behavior conversion

Modified Language Environment-Behavior conversions use tables that map characters like the *iconv()* function of the Language Environment Runtime library does for converters ending with "C" (for example IBM-932C).

For mixed data format conversion, it is advisable to use more than one technique-character as one of the sub-conversions might exist only in round-trip mode and one only in enforced-subset. In the case of NFSv4, a technique-search-order of convserv(RE) or convserv(ER) would be required. In contrast, in the case of NFSv2 or NFSv3, convserv(R) would be sufficient for the data conversion.

The **convserv** attribute uses a default value of "LRE" which is recommended to provide correct translation of EBCDIC-newline to ASCII and back.

Collecting NFS usage data

The z/OS NFS client does not produce any System Management Facilities (SMF) records. However, it does provide the accounting information to z/OS UNIX for SMF recording. z/OS in turn provides the SMF recording services for all physical file systems (PFSs).

The z/OS NFS server does not produce z/OS UNIX SMF records. However, z/OS UNIX provides the SMF recording services for all physical file systems (PFSs).

You can use the SMF records that the z/OS NFS server produces to keep track of how MVS z/OS conventional data sets are accessed, and how long each Network File System user session lasts. The z/OS NFS server writes the following SMF records:

Record type-42 subtype 7
This record, written when a file times out, provides the Network File System file usage statistics.

Record type-42 subtype 8
This record, written when a client user logs out of NFS, provides the Network File System user session statistics.

Record type-42 subtype 26
This record, written when a client creates, removes, or renames the file objects on the NFS mounted file system, provides the NFS Client's information, the type of operation (create, remove, rename) and object descriptive information (depending on file system type: MVS or z/OS Unix). For z/OS Unix objects, the file system name, device number, object name, FID, and parent FID information are saved. If the pathname of an object is desired, it is recommended to enable SMF recording for USS Type 92 and RACF Type 80 records in order to derive this information. For MVS objects, the volume name, full dataset name, and member name (if appropriate) are saved.

For records containing Internet Protocol (IP) Version 6 addresses, the z/OS NFS server writes a specific type of SMF record. This record type is indicated by a version number of **2** in the smf42psv record field. In these records, the IP address field (smf42cip) is expanded to hold the larger IP V6 address values. For details on these record fields, see Appendix J, "SMF C and assembler header macros," on page 507.

You can control the SMF data collection in the following ways:

- You can use the **smf** site attribute, described in "Site attributes syntax" on page 134, to determine which, if any, SMF statistics are to be collected.
- You can use the **smf=on** or **smf=off** operand of the **modify** command. See "Smf operand" on page 236 for a description of this command, which turns SMF data collection on and off.
- You can generate an SMF report. Use the SMF report sample routine, GFSAPSMF, that can be found in the NFSSAMP library.
- You can use the SMF C and Assembler header macros. See Appendix J, "SMF C and assembler header macros," on page 507 for copies of the C header macro, GFSASSMF, and the Assembler header macro GFSAUSMF. Both header macros contain the mapping for SMF records and can be found in the NFSMAC library.
- Check the SMF setting in the system in SYS1.PARMLIB(SMFPRMnn) for the SMF record type and subtype, where nn is determined by IEASYSmm and the operator command (SET SMF=nn). The Network File System uses the SMF type 42 record, subtypes 7 and 8. You specify **SMF=nn** so the system picks the member of SMFPRM with suffix nn.

You can display the SMF settings with the **d smf,0** operator command.

The SMF write macro, SMFWTM, is used to write the SMF records to the SMF data set. When the server starts, the SMF option is disabled. Therefore, the operator needs to explicitly enable the SMF collection.

For more information about SMF see *z/OS MVS System Management Facilities (SMF)*.

Configuring the z/OS NFS client

This section describes the tasks you can perform to configure the z/OS NFS client. These tasks include creating the PARMLIB statement and updating z/OS system data sets for the client. This section also includes information about allocating client log data sets and mounting remote file systems.

Creating the PARMLIB statement for the client

During z/OS UNIX file system initialization, the z/OS NFS client is started and run in the logical file system (LFS) colony address space. The filesystype parmlib statement for the z/OS NFS client must be present in the SYS1.PARMLIB(BPXPRMxx) parmlib member in order to start the z/OS NFS client. For more information about z/OS UNIX file system reference see *z/OS UNIX System Services File System Interface Reference*.

Updating z/OS system data sets for the client

To accommodate the z/OS NFS client you must update z/OS system data sets PARMLIB, PROCLIB, and the DD statement.

PARMLIB updates

Add the data set defined in the GFSCPROC STEPLIB containing the z/OS NFS client library to the system's APF authorization list (IEAAPFxx). A sample cataloged procedure named GFSCPROC is provided as a member of the sample library NFSSAMP, see "Sample z/OS NFS client startup procedures" on page 499.

Add the filesystype parmlib statement shown in Figure 21 to the z/OS UNIX parmlib member (BPXPRMxx):

```
FILESYSTYPE
   TYPE(NFS)
   ENTRYPOINT(GFSCINIT)
   PARM('installation parms')
   ASNAME(proc_name)
```

Figure 21. Sample filesystype parmlib statement

The name in the TYPE operand must be NFS, otherwise the utility program nfsstat fails.

The operand ENTRYPOINT(GFSCINIT) specifies the entry point for the z/OS NFS client initialization.

The operand PARM('installation parms') specifies the installation parameters for the z/OS NFS client. See Table 16 on page 111 in "Mount processing parameters and installation parameters" on page 111 for a list of valid installation parameters.

The operand ASNAME(proc_name) specifies the procedure name in SYS1.PROCLIB that is used by z/OS UNIX to start the address space in which the z/OS NFS client is initialized.

Note: The proc_name is also used for the name of the address space.

For data integrity and data isolation among different PFSs, the z/OS NFS client is required to start in a separate and standalone colony address space. To start the NFS client in a separate and standalone colony address space, a unique proc_name must be used.

For information about BSAM, QSAM, and VSAM ESDS files, see "BSAM, QSAM, and VSAM ESDS access to remote files" on page 87.

PROCLIB updates

Add the procedure name, *proc_name*, specified in the **ASNAME**(*proc_name*) operand to the system `PROCLIB`.

A sample cataloged procedure named GFSCPROC is provided as a member of the sample library NFSSAMP, see "Sample z/OS NFS client startup procedures" on page 499.

Modify the MVSNFSC procedure and place it in your system PROCLIB. Add the DD statements:

```
NFSCMSG1 as the DD for the primary log data set
NFSCMSG2 as the DD for the secondary log data set
SYSxDUMP as the DD for the SYSxDUMP data set ('x' = U or M)
```

Allocating client log data sets

For information about allocating the z/OS NFS client primary and secondary log data sets, see Appendix K, "Capturing diagnostic information using z/OS NFS log data sets and from other components," on page 523.

NFS Client with Multiple TCPIP stacks

In order for the NFS client to bind to multiple TCPIP stacks, a single rpcbind/portmap should be used. The rpcbind/portmap procs should not use affinity to bind to a specific TCPIP stack. Also, the system should use a single resolver for all stacks on the system. If you require transport affinity with the NFS Client, see the section "Using specific transports under CINET" in *z/OS UNIX System Services Planning*, GA22-7800 for information on using PARM=TP(TPNAME) on the EXEC statement that starts BPXVCLNY in the colony address space procedure.

Note: Multiple instances of the NFS Client are not supported.

Mounting remote file systems

In order to make a connection between a mount point on your local z/OS UNIX file system and one or more files on a remote MVS, AIX, UNIX, z/OS, or other file system, any z/OS UNIX mount method may be used, including:

- z/OS UNIX automount facility
- /etc/rc shell scripts support
- z/OS UNIX shell mount command
- SO MOUNT
- BPXPMRxx MOUNT statement

The remote file system can be mounted only after the z/OS UNIX file system, z/OS NFS client, and TCP/IP initializations are complete. The mount can only be performed by a z/OS UNIX superuser (uid=0). For more information about the

TSO MOUNT command, when used with the z/OS NFS client, see “Mount command syntax and examples” on page 85.

When using the automount facility of z/OS UNIX System Services, the remote file system is mounted on its first data access attempt if it is not already mounted.

When the automount facility is used to manage remote NFS mount points, the z/OS NFS user could experience ESTALE/EIO errors if the automounter unmounts the accessed mount point when the time limits specified by the automount duration and delay parameters have been exceeded. The automount default, DURATION=NOLIMIT, disables the DURATION timeout period. The DELAY for unmounting file systems should be longer than the time z/OS NFS clients are likely to keep z/OS NFS mounts to the files and directories active. For more information about the z/OS UNIX automount facility, see *z/OS UNIX System Services Planning* and *z/OS UNIX System Services Command Reference*.

The remote file system must be mounted on the z/OS UNIX file system prior to any reference being made to the remote data. Once mounted, the remote file system can be treated as an extension of the local z/OS UNIX file system.

Setting up reserved (privileged) ports

The z/OS NFS client uses a reserved (privileged) port to prevent the NFS server from rejecting a client request. The z/OS client attempts to use reserved port 1023 and if that port is not available, the z/OS client will subtract one from 1023 until a reserve port is available. If no reserve ports are available, the z/OS client will issue error message GFSC724E.

The amount of reserved ports the z/OS client can use is based on the client attribute biod. The amount of reserved ports can be calculated from the following formula:

```
reserved ports = 8 + ( #biod * 4 )
```

The privileged ports should be reserved in the tcpip.profile file using the PORTRANGE statement. The default biod(6) and 8 additional ports correspond to 32 privileged ports that can be used by the z/OS client. For biod(6), the tcpip.profile file should include the following PORTRANGE statement:

```
PORTRANGE 991 32 UDP MVSNFSC
PORTRANGE 991 32 TCP MVSNFSC
```

This allows ports 991 through 1023 to be used by the z/OS client. Note that MVSNFSC is the default z/OS NFS client start-up procedure. Please specify the correct z/OS NFS client start-up procedure if it is different than the default.

When specifying secure(udp) or proto(udp), the z/OS client uses the privileged UDP ports to communicate with the NFS servers. When specifying proto(tcp) the z/OS client uses the privileged TCP ports to communicate the MOUNT RPC or UNMOUNT RPC with the NFS server. However, the z/OS client uses the ephemeral TCP ports to communicate NFS RPC with the NFS server. As a result, the z/OS client does not work with NFS servers that require **all** source TCP ports to be privileged.

Configuring a secure z/OS NFS client

In order for the z/OS NFS client to support the RPCSEC_GSS authentication flavor using the Kerberos V5 Security Mechanism, the following should be done. We are assuming that a properly configured KDC is already setup in your environment. If a KDC is not already configured, see Appendix M, “Setting up NFS functions with Kerberos Support,” on page 535. This document assumes you will be using a KDC provided by “Security Server and Integrated Security Services” (RACF) provided by IBM but are not specific to a KDC provided by IBM. The following steps need to be run on the system that contains the KDC unless otherwise noted:

1. 1. Add the client principal "mvsnfsc" to the Kerberos database on the KDC with a defined password. This principal should not use randkey as the password. For example, for the z/OS NDBM type or SUN KDC, issue the command "addprinc mvsnfsc" in the kadmin interface, then enter the desired password in the prompt. For the z/OS SAF type KDC, the password can be defined in the "PASSWORD" field of the RACF "adduser" or "altuser" commands. Refer to “Setting up a Kerberos Key Distribution Center” on page 537 for more information.
2. Map the principal "mvsnfsc" in lower case to a RACF user. For example:

   ```
   ADDUSER pznfsc OWNER(ibmuser) OMVS(UID(100))
   ALTUSER pznfsc PASSWORD(password) NOEXPIRED KERB(KERBNAME('mvsnfsc'))
   PASSWORD USER(pznfsc) NOINTERVAL
   ```

   Notes:

   a. If you are NOT using a SAF KDC skip the RACF commands in Step 1 and create a principal ‘mvsnfsc’ according to your venders KDC documentation.
   b. Change “pznfsc” to meet your installation requirements to refer to the user of the z/OS NFS Client.
   c. The RACF user ID “pznfsc” to whom the z/OS NFS client uses requires a unique UID in the OMVS segment.
   d. The NFS Client requires that the Kerberos segment use KERBNAME ‘mvsnfsc’ and it must be in lowercase.
   e. Including PASSWORD option “NOINTERVAL” prevents the password from expiring.
3. From the omvs shell, the system administrator must add the principal "mvsnfsc" into the keytab. If /etc/skrb/krb5.keytab does not exist, create a new one. For example:

   ```
   IBMUSER:/ :> keytab add mvsnfsc -p password -k /etc/skrb/krb5.keytab -v 1
   ```

   Notes:

   a. The "password" in step 2 must match the "password" entered in step 1, when the principal was added to the SAF KDC Kerberos database. This principal "mvsnfsc" is used to perform mounts for all users. To gain access to a secure mount point, all users should perform a "kinit" to acquire their Kerberos credentials.
   b. The key version used to create the keytab must be the same key version as in the racf database. The “-v” option of the keytab command is used to specify the key version when adding a principal to a keytab. Issue the following RACF command to see the current key version: LU pznfsc NORACF KERB Or from a non SAF or NDBM KDC from kadmin interface issue: getprinc mvsnfsc
   c. The password used with the “keytab” command is case sensitive. If mixed case password support is not in effect you must enter the password in uppercase.
   d. SFTP, or Secure copy the new keytab to the z/OS NFS Client and place in “/etc/skrb/krb5.keytab".
4. The z/OS NFS client requires the Kerberos configuration file “krb5.conf” be configured to match your sites Kerberos environment.

Sample /etc/skrb/krb5.conf file to be put on the z/OS NFS client system:

```
[libdefaults]
default_realm = KRB390.IBM.COM
kdc_default_options = 0x40000010
use_dns_lookup = 0
default_tkt_enctypes = des-cbc-crc
default_tgs_enctypes = des-cbc-crc

[realms]
KRB390.IBM.COM = {
kdc = dcesec4.krb390.ibm.com:88
kpasswd_server = dcesec4.krb390.ibm.com:464
admin_server = dcesec4.krb390.ibm.com:749
}
KRB2000.IBM.COM = {
kdc = sstone1.krb2000.ibm.com:88
admin_server = sstone1.krb2000.ibm.com:749
}

[domain_realm] .
krb390.ibm.com = KRB390.IBM.COM .
krb2000.ibm.com = KRB2000.IBM.COM
```

Note: If the z/OS NFS server will be supporting multiple platforms of NFS clients, IBM recommends using des-cbc-crc encryption types only for compatibility, as shown in this example.

Configuring the z/OS NFS server

This section describes the tasks you can perform to configure the z/OS NFS server. These tasks include allocating and modifying data sets, allocating mount handle, lock, side file, and z/OS NFS server log data sets, modifying tcpip.ETC.RPC, and updating z/OS system data sets for the server. This section also includes information about the data conversion between EBCDIC and ASCII.

Attributes data set

To allocate and modify the attributes data set, perform the following tasks:

1. Allocate a fixed-block partitioned data set or a fixed-block sequential data set with a record length of 80.
2. Copy the sample member GFSAPATT from the *prefix*.NFSSAMP data set into the allocated attributes data set.
3. Modify the attributes to suit your installation. Appendix E, “NFS system server sample attribute table,” on page 465 shows the sample attributes data set. You can specify three sets of attributes within the attributes data set.
   - Data set creation attributes
   - Processing attributes
   - Site attributes.

Notes:

1. Client users can override the processing and data set creation attributes (for their own sessions), but not the site attributes.
2. The attributes data set specified on the NFSATTR DD statement of the mvsnfs startup procedure is read during server startup processing. Further changes to this data set do not take effect until the server is restarted. Also, whenever any attributes are changed, all the previous existing mount points have to be unmounted and mounted again (using the **umount** and **mount** client commands) if you want the mount points to pick up the new attributes.

Specify this attributes data set for the NFSATTR DD statement in the MVSNFS cataloged procedure.

Coding attribute statements

Here are guidelines for coding attribute statements:

- You can continue a line in the attributes data set by placing a "\" or "+" at the end of the line.
- A "#" anywhere in the data set indicates a comment that extends to the end of the line (unless the **altsym** keyword is used in the **start** command or the server startup procedure; if **altsym** is used, a ";" indicates a comment but with one exception: If the fn_delimiter(;) attribute uses a semicolon, the fn_delimiter semicolon is treated as a delimiter between the file name and the attributes that follow, not as a comment symbol).
- If you specify more than one attribute on a line, separate the attributes with a comma and a space.

Exports data set

To allocate and modify the exports data set, perform the following tasks.

1. Allocate a fixed block partitioned data set or a fixed block sequential data set with record length of 80.
2. Copy the sample member GFSAPEXP from the *prefix*.NFSSAMP data set into the allocated data set.
3. Modify the sample exports data set to suit your installation. Appendix F, "Sample exports data set," on page 487 shows the sample exports data set.
4. Specify this exports data set for the EXPORTS DD statement in the MVSNFS cataloged procedure.

The exports data set contains entries for directories that can be exported to clients. It is used by the server to determine which data sets' high-level qualifiers or z/OS UNIX directories can be mounted by a client in a read or write mode.

Note: You cannot specify exporting a "parent directory" or a subdirectory of an exported directory. For example, if you specify DIR1 in the exports data set, DIR1 and all its subdirectories are exported. You cannot specify any subdirectories under DIR1 in the exports data set.

When the modified exports data set takes effect

The exports data set specified in the exports DD statement of the MVSNFS startup procedure is read during server startup processing. Changes to this file do not take effect for new mount points until the **exportfs** operand of the **modify** command is completed or the server is restarted. The changes will also be immediately enforced at z/OS NFS server startup or EXPORTFS operator command time regardless of the state of any pre-existing mount points. Note that any pre-existing mount points continue to exist even if they have been removed from the exports dataset. However, any security changes apply immediately.

When the exportfs operand of the modify command is issued, any errors found in the file are sent to the system log, the entire exports list is not refreshed, and processing continues.

When the server is started, if any errors are found in the file, they are sent to the system log, and the server stops once the entire exports data set has been read.

Directory statement

Use the directory statements in the exports data set to limit access of directories to specified client workstations.

- Entries can be up to 4096 characters long. Directory names for MVS data sets must follow z/OS naming conventions.
- Lines can be continued by placing a backslash (\) or a plus sign (+) at the end of the line.
- A "#" anywhere in the data set indicates a comment that extends to the end of the line (unless the **altsym** keyword is used in the **start** command or the server startup procedure; if the altsym keyword is used, the ";" indicates a comment).
- Spaces cannot be used in the keywords.
- A vertical bar (|) acts as a separator character between multiple list entries such as client ids or network security values. Before z/OS V1R8, the separators were colon (:) characters . Since client ids can include colons as part of their IPv6 addresses, the separator characters are changed to prevent ambiguity. Any colon separator characters must be changed to vertical bars in exports files that are used with z/OS V1R8.
- The parameters to the right of directory are optional. Except for **ro** and **rw**, the parameters can be combined. If they are combined, only the first parameter is preceded by "-". For example:

  ```
  user1.test -access=rs60001|sun1|sun2,ro
  ```

Note: Prefix site attributes apply to the exports file. Either the hfsprefix or mvsprefix can be specified as part of the directory name. If no prefix is specified, then the implicit prefix algorithm applies. If both implicit options are specified, the export entry can apply to both z/OS UNIX file systems and MVS data sets. The existence check is applied to the mount request, not to the export entry itself.

An entry for a directory is specified in Figure 22 on page 174.

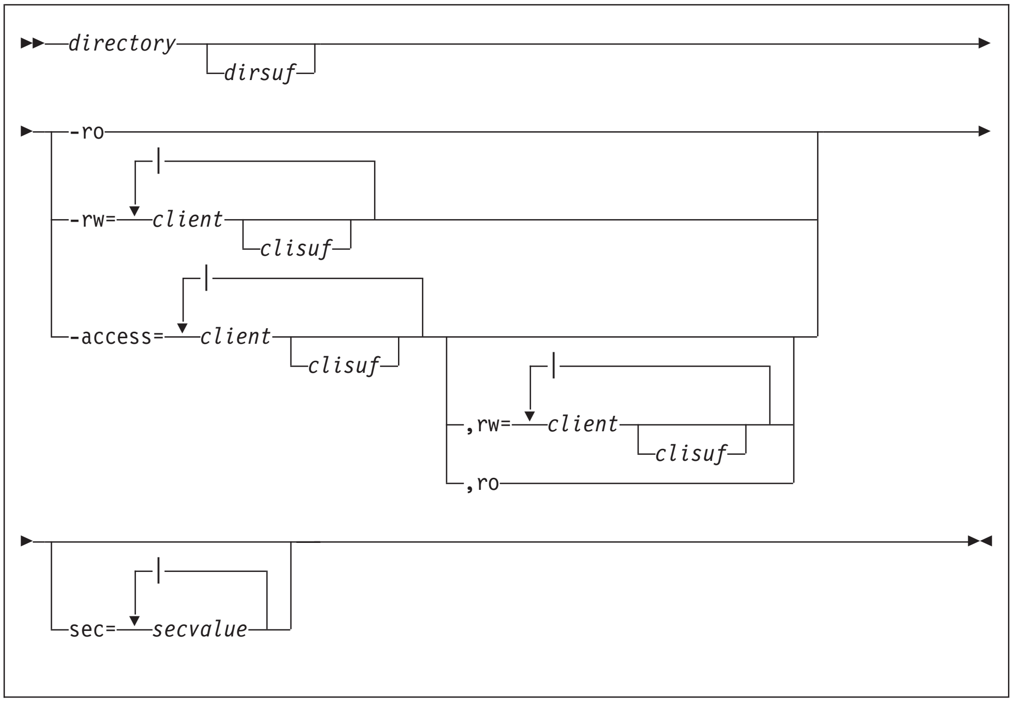


Figure 22. Entry for a directory

where

directory
For MVS data sets, the MVS high-level qualifier, partitioned data set name, or alias to a user catalog; the name must conform to MVS data set naming conventions. The name may be preceded by the MVS prefix.

For z/OS UNIX file systems, the entire UNIX directory path name, starting from the root. The name may be preceded by the HFS prefix.

If no prefix is specified, the implicit prefix algorithm is used for determining which file system type this entry applies to. If both implicit options are active, and are valid for this entry, then the entry will be assumed to apply to both file system types. No existence check is performed to refine the file system type selection to a single option.

For both MVS data sets and z/OS UNIX directories, wildcard symbols can be used to create regular expressions, and MVS system symbols can also be used to create regular expressions for MVS data sets.

dirsuf
Suffix for a directory to be exempt from SAF checking, even though SAF or SAFEXP is specified as the security option. This parameter emulates an entry in the NFS checklist data set prior to z/OS V1R8. Prior to z/OS V1R8, the Checklist used a one-to-one correspondence policy, that is, one checklist entry should generate one real path that was to be exempted from SAF checking. In z/OS V1R8 and above, the former checklist one-to-one correspondence policy remains unchanged; MVS system symbols and wildcard symbols must not be used in a directory if dirsuf is used together with a directory. The suffix is ignored if the **checklist** site attribute is not specified, or the **security** site attribute is not set to SAF or SAFEXP. The directory suffix can have the following values:

blank

If the **security** site attribute is set to SAF or SAFEXP, SAF checking is performed for this directory and its sub-directories.

<nosaf>

If the **security** site attribute is set to SAF or SAFEXP, SAF checking is not performed for this directory and its sub-directories. This value emulates the checklist function prior to z/OS V1R8.

<*sub-directory list*,**nosaf>**

If the **security** site attribute is set to SAF or SAFEXP, SAF checking is performed for this directory. However, for the specified sub_directory_list, and its sub-directories, SAF checking is not performed.

sub-directory list

A list of sub-directories separated by commas, and optionally client host lists.

sub-directory[**,hosts=***client_list*]

sub-directory

Name of a sub-directory to which the **nosaf** function is to be applied. The directory name is appended to the front of each sub-directory name to generate to full name of the directory item for which no SAF checking is to be performed.

A sub-directory entry can also refer to a specific member of an MVS PDS or PDSE for which no SAF checking is to be performed: *sub_directory(member)*.

client_list

List of client host names to which the **nosaf** function is to be applied. If no *client_list* is specified, then the function applies to all clients.

Note: The hosts=*client_list* specification in the directory suffix expands the checklist functionality beyond that available in prior releases. This parameter allows you to limit the checklist applicability to a subset of hosts, only those specified in the *client_list.*

-ro

Export the directory as read only. If not specified, the directory is exported as read/write.

-rw=*client*[*clisuf*][|*client*...]

The directory is exported as read/write to specified *clients*, and read-only to everyone else. Use a vertical bar (|) to separate client names. *Client* names may be specified as shown in "Client id specification" on page 179.

-access=*client*[*clisuf*][|*client*....] [[**,rw=***client*[|*client*...]] | [**,ro**]]

Gives access only to *clients* listed. *Client* names may be specified as shown in "Client id specification" on page 179.

If neither **rw** nor **ro** is specified for the **-access** parameter, then the clients listed have read/write access and the rest of the clients have no access.

If the **rw** parameter is specified for the **-access** parameter, the associated clients have read/write access to the directory, and the clients specified in the **access** list but not in the **rw** list have read-only access.

If the **ro** parameter is specified for the **-access** parameter, the clients in the **access** list have read-only access to the directory, and the rest of the clients have no access.

Use a vertical bar (|) to separate client names.

client

Name of the client to which the specification applies. *Client* names may be specified as shown in "Client id specification" on page 179.

clisuf

The client suffix can be specified to give root access to the root user of the specified client. The client suffix can have the following values:

blank

If no MVSLOGIN has been done, the id of root users from the client system are converted to id=**nobody** (-2) before access permissions are checked. This means that there is a good chance that root users will be denied access to the directory while other users from that client have access.

<root>

If no MVSLOGIN has been done, root users from the client system are given root access to this directory and its subdirectories (that is, the user is treated as a trusted user).

sec=*secvalue*

Specifies the acceptable level of network transmission security access that a client's RPC request must provide. If a client attempts to access an object in the directory using a network security level that is not specified on the **sec** parameter, the access is denied.

krb5

Provides Kerberos V5 based integrity on the RPC credentials (but not data), when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 integrity algorithm and the RPCSEC_GSS service of rpc_gss_svc_none.

krb5i

Provides Kerberos V5 based integrity on both the RPC credentials and data, when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 integrity algorithm and the RPCSEC_GSS service of rpc_gss_svc_integrity.

krb5p

Provides Kerberos V5 based integrity and privacy on both the RPC credentials and data, when the RPC authentication flavor is RPCSEC_GSS. It uses the DES_MAC_MD5 algorithm for integrity and 56 bit DES for privacy. The RPCSEC_GSS service used here is rpc_gss_svc_privacy.

sys

Specifies that the AUTH_SYS authentication flavor can also be used to access this file system. Note that the AUTH_SYS authentication flavor does not provide any integrity or privacy protection.

Use a vertical bar (|) to separate security levels.

If the **sec** parameter is provided, it further restricts the network transmission protection of specific export entries, in the domain of the file system wide network transmission protection, which is controlled by the **mvssec**, **hfssec**, or

pubsec site attributes. When this parameter is not specified, the allowed security levels are governed by the **mvssec**, **hfssec**, and **pubsec** site attributes.

Note: If no options are specified, the default value allows any client to mount the given directory with read/write access.

Examples of entries in an exports data set: Following are examples of entries in an exports data set.

Examples of specifying directories: The following are examples of specifying directories for z/OS UNIX files:

```
/u                # If the IMPPREFIX( ) site attribute is set to HFS, this entry
                  # will be interpreted to refer to z/OS UNIX directory /u.
                  # If the IMPPREFIX( ) site attribute is set to MVS, this entry
                  # will be interpreted to refer to MVS high-level qualifier 'u'.
                  # If the IMPPREFIX( ) site attribute is set to HFS,MVS or MVS,HFS,
                  # this entry will be interpreted to refer to both.
/hfs/u            # If the SECURITY Site Attribute is set to SAF or SAFEXP,
                  # SAF checking is performed for /hfs/u and its subdirectories.
/hfs/u<nosaf>     # If the SECURITY Site Attribute is set to SAF or SAFEXP, NO
                  # SAF checking is performed for /hfs/u and its subdirectories.
/hfs/u<vrr,nosaf> # If the SECURITY Site Attribute is set to SAF or SAFEXP, SAF
                  # checking will be performed for /hfs/u, but NO SAF checking
                  # will be done for /hfs/u/vrr and its subdirectories.
/hfs/u<vrr,/vrs,nosaf> # If the SECURITY Site Attribute is set to SAF or SAFEXP,
                  # SAF checking will be performed for /hfs/u, but NO SAF checking
                  # will be done for /hfs/u/vrr, /hfs/u/vrs and their
                  # subdirectories.
/hfs/u<vrr,hosts=host1,host2,vndrcvs,nosaf> # if the SECURITY Site Attribute ist
                  # set to SAF or SAFEXP, SAF checking will be performed for
                  # /hfs/u,and for client hosts other than host1 and host2 for
                  # /hfs/u/vrr. NO SAF checking will be done for hosts host1 and
                  # host2 for /hfs/u/vrr or for any host for /hfs/u/vndrcvs.
                  # The same applies to subdirectories of vndrcvr and vndrcvs.
```

The following are examples of specifying directories for MVS data sets:

```
/mvs/a.b          # If the SECURITY Site Attribute is set to SAF or SAFEXP, SAF
                  # checking is performed for a.b and its lower level qualifiers.
/mvs/a.b<nosaf>   # If the SECURITY Site Attribute is set to SAF or SAFEXP
                  # NO SAF checking is performed.
a.b<nosaf>        # If the SECURITY Site Attribute is set to SAF or SAFEXP,
                  # NO SAF checking is performed for a.b and its lower level
                  # qualifiers.
a.b<c.d,nosaf>    # If the SECURITY Site Attribute is set to SAF or SAFEXP, SAF
                  # checking will be performed for a.b and a.b.c, but NO SAF
                  # checking will be done for a.b.c.d and its lower level
                  # qualifiers.
a.b<c.d(memb1),nosaf> # If the SECURITY Site Attribute is set to SAF or SAFEXP,
                  # SAF checking will be performed for a.b, a.b.c, and a.b.c.d
                  # and all its members, except NO SAF checking will be done for
                  # a.b.c.d(memb1).
a.b<c.e,c.d(memb1),nosaf> # If the SECURITY Site Attribute is set to SAF or SAFEXP,
                  # SAF checking will be performed for a.b, a.b.c, and a.b.c.d and
                  # all its members, except NO SAF checking will be done for
                  # a.b.c.e and a.b.c.d(memb1).
A.b<c.e,c.d(member1),hosts=host1,nosaf> # If the SECURITY Site Attribute is set to
                  # SAF or SAFEXP, SAF checking will be performed for a.b, a.b.c
                  # and a.b.c.d and all its members, except NO SAF checking will
                  # be done for a.b.c.e or for host1 for a.b.c.d(member1).
```

Examples of specifying access parameters: Following are examples of specifying access values in an exports data set.

```
/mvs/mvsnfs     -ro,sec=krb5                 # give read-only access
                                             # to all clients with RPCSEC_GSS
                                             # security specified
                                             #
theresa.text                                 # give read/write
                                             # access to all clients
                                             #
robert.mixds   -rw=fsrs001|fslab004|fslab007 #
                                             # give read/write access
                                             # to the clients named
                                             # fsrs001, fslab004 and
                                             # fslab007, and give
                                             # read-only access to
                                             # all other clients
                                             #
/hfs/newproductdirectory  -rw=johnson        # give read/write access to
                                             # this z/OS UNIX directory to the
                                             # client named johnson;
                                             # give read-only access to
                                             # all others
                                             #
barbara.pds    -access=fsrs001|fslab007      #
                                             # give read/write access
                                             # only to clients named
                                             # fsrs001and fslab007
                                             #
daniel.pds2    -access=fslab004,ro           # give read-only access
                                             # only to the client
                                             # named fslab004
                                             #
virginia.vsam    -access=fslab004|fslab007,rw=fslab004 #
                                             # give read-only
                                             # access only to the
                                             # client named fslab007,
                                             # and give read/write
                                             # access to fslab004.
                                             #
/hfs/u          -sec=krb5|krb5i|krb5p        # client must use krb5, krb5i or
                                             # krb5p authentication levels to
                                             # access server, provided that
                                             # hfssec also allows these
                                             # authentication levels.
```

Notes:

1. If your installation cannot use the “#” as a comment delimiter, see “Starting the z/OS NFS server” on page 215.
2. The keywords **ro** and **rw** are mutually exclusive.
3. The ability to write (that is, **rw** specified or **access** specified without other parameters) implies read access also.
4. If **access** and **rw** are specified together, the client names in the **rw** list are logically or'ed with the **access** list to determine the total list of clients with read access.
5. Multiple lines can be used in the exports data set for a given directory to merge the **access** list and the **rw** list. However, similar clauses (for example, an access followed by an access) completely replace any previous specification. If **ro** is specified for a data set on one line and a further line specifies **rw** for that data set, the **rw** undoes the **ro** specified earlier. Similarly, a line with null options completely undoes all previous specifications for that directory, giving read/write access to all clients.
6. It is not appropriate to have the same data set or z/OS UNIX directory defined more than once in the exports data set. If for any reason this is the case, only the last definition in the exports data set is valid.

7. If a directory entry is specified without a prefix, the IMPPREFIX( ) site attribute specifies both options, and there is nothing in the directory name syntax that explicitly limits it to one or the other of the two file system types, then the directory entry is exported for both file system types.

Client id specification

There are several options for specifying the client hostname in the exports data. Some options apply only when NODHCP is specified in the site attributes file and others apply regardless of the DHCP mode. The client specification options are as follows.

Single hostname
: This is the most common format and the one supported in releases before z/OS V1R8. In this format the client is specified by a hostname recognized by the DNS resolver. This name must be unique and unchanging for the duration of the NFS connections. A client suffix may be specified with this format.

Netgroup name
: Name of a netgroup defined in the local /etc/netgroup file. The group entry in the file lists the hosts who are members of the group. Only the host part of each netgroup member is considered for checking for membership. Empty host parts, or those containing a single dash (-) are ignored. Netgroup names must be preceded by an at-sign (@), for example @group. A client suffix may not be specified with this format.

Single IP address
: A client may be specified by an IPv4 or IPv6 address. Invalid IPv4 or IPv6 address specifications are ignored. If the NFS server starts in IPv4 mode and an IPv6 address is specified, it is ignored. If the NFS Server starts in IPv6 mode and an IPv4 address is specified, the address is translated to an IPv4-mapped address (which is standard IPv4 address handling in IPv6 networks). This option is only valid in NODHCP mode. In DHCP mode such client specifications are ignored. A client suffix may be specified with this format.

IP networks
: Directories can also be exported to all hosts on an IPv4 or IPv6 network or subnetwork simultaneously. For IPv4 networks, specify an IP address and netmask pair as address/netmask where the netmask can be specified in dotted-decimal format, or as a contiguous mask length. For example, either '/255.255.252.0' or '/22' appended to the network base address will result in identical subnetworks with 10 bits of hostname. IPv4 addresses and mask lengths are checked for format and range, and ignored if invalid. A range from 1 to 31 is assumed.

  For IPv6 networks, this is done by specifying IPv6 address/prefix-length. For example, the node address 12AB:0:0:CD30:123:4567:89AB:CDEF and its subnet number 12AB:0:0:CD30::/60 can be abbreviated as 12AB:0:0:CD30:123:4567:89AB:CDEF/60. IPv6 addresses and prefix-length are checked for format and range, and ignored if invalid. A range from 1 to 127 is assumed.

  If the NFS server starts in IPv4 mode and an IPv6 address is specified, it is ignored. If the NFS Server starts in IPv6 mode and an IPv4 address is specified, the address is translated to an IPv4-mapped address (standard IPv4 address handling in IPv6 networks).

  This is option is only valid in NODHCP mode. In DHCP mode, IP network entries are ignored. A client suffix may not be specified with this format.

Netgroup definitions: Since z/OS does not support the NIS environment, z/OS NFS can only support netgroups defined in the local /etc/netgroup file. z/OS NFS assumes the same file record length restriction as for the NIS environment (that is, a maximum size of 1024 bytes). A netgroup cannot have a client id list longer than 1024 characters. z/OS NFS assumes the same wildcard character ('*' and '?') restrictions as the NIS environment (that is, no wildcard character specification as part of a netgroup name, nor as part of client host names specified in a netgroup file).

z/OS NFS also assumes that the local netgroup file contains netgroup information without the NIS "+" token. In other words, the local netgroup file does not contain any LOCAL netgroup information and all network netgroup information put into the file from the NIS data base without the NIS "+" token. Any "+" tokens encountered are ignored.

Here is an example of some sample local netgroup files:

```
# /etc/netgroup local file.
#
LocalAdmins (,root,)
NetAdmins (sonne,user1,) (sonne,user2,) LocalAdmins
Gateways (rechner1,,) (rechner7,,)
```

Here is another example:

```
# /etc/netgroup local file.
# shown one netgroup, it must contain no more than 1024 chars.

cnet2331 (bart.foobar.net,,cnet2331.foobar.net) \
         (lisa.foobar.net,,cnet2331.foobar.net) \
         (dead.foobar.net,,cnet2331.foobar.net)
```

Wildcard characters: In DHCP mode, client host names may contain the wildcard characters ' * ' and ' ? ' . Wildcard characters ' * ' and ' ? ' cannot be specified in netgroup names, nor as part of client host names specified in a netgroup file. These characters can be used to make exports files more compact; for example, *.cs.foo.edu matches all hosts in the domain cs.foo.edu. However, these wildcard characters do not match the dots in a domain name, so that example does not include hosts such as a.b.cs.foo.edu.

When wildcard characters are used, the domain name is mandatory because the NFS server cannot check wildcard host names with DNS or rely on any default domain name definitions.

When wildcard characters are used, client suffix specification is not permitted.

IP address representation: Text representations of IP addresses must conform to the industry standard defined in RFC-2373, IP Version 6 Addressing Architecture. IPv4 addresses and masks must be written in the standard IPv4 dotted-decimal form:

```
ddd.ddd.ddd.ddd
```

where ddd is a one-to-three digit decimal number between 0 and 255.

IPv6 addresses must be written in the standard IPv6 form:

```
x:x:x:x:x:x:x:x
```

where the xcharacters are the hexadecimal value of the eight 16-bit pieces of the address. For example:

```
FEDC:BA98:7654:3210:FEDC:BA98:7654:3210
1080:0:0:0:8:800:200C:417A
```

Due to some methods of allocating IPv6 addresses, it is common for addresses to contain long strings of zero bits. To simplify writing addresses containing zero bits, you can use a special syntax to compress the zeros. You can use two colons (::) to indicate multiple groups of 16-bits of zeros. The double colons (::) can only appear once in an address. They can also be used to compress the leading and/or trailing zeros in an address. For example, the following shorthand addresses can be used:

Table 29. Shorthand for addresses with multiple zero bits

Address	Shorthand	Description
1080:0:0:0:8:800:200C:417A	1080::8:800:200C:417A	Unicast address
FF01:0:0:0:0:0:0:101	FF01::101	Multicast address
0:0:0:0:0:0:0:1	::1	Loopback address
0:0:0:0:0:0:0:0	::	Unspecified address

An alternative form that is sometimes more convenient in a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the `x` characters are the hexadecimal values of the six high-order 16-bit pieces of the address, and the `d` characters are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). Some examples follow:

Table 30. Shorthand for addresses in mixed IPv4 and IPv6 environments

Address	Compressed Form	Description
0:0:0:0:0:0:13.1.68.3	::13.1.68.3	IPv4-compatible addresses
0:0:0:0:0:FFFF:129.144.52.38	::FFFF:129.144.52.38	IPv4-mapped addresses

Address Prefix Representation: The text representation of IPv6 address prefixes is similar to that of IPv4 addresses prefixes. They are written in CIDR notation. An IPv6 address prefix is represented by the notation: `Ipv6-address/prefix-length`.

Abbreviation: a node address and its subnet address can be abbreviated, as in the following example:

Node address
12AB:0:0:CD30:123:4567:89AB:CDEF

Its subnet number
12AB:0:0:CD30::/60

Abbreviation
12AB:0:0:CD30:123:4567:89AB:CDEF/60

Checklist data set

In z/OS V1R8, the function of the z/OS NFS checklist data set was merged into the exports data set. If you used a checklist data set in previous releases, use the **dirsuf** parameter to specify its contents in the exports data set. See “Directory statement” on page 173 for details. With this merger, the checklist information can be refreshed dynamically using the **exportfs** operand of the **modify mvsnfs** command, instead of requiring a restart of the NFS server.

Mount handle data sets

The mount handle data sets are used to record active mounts during server operation and allow clients to stay mounted when the server is shut down and restarted. The Network File System alternates between two data sets to record this information; only one data set is used at a time, and it is switched at either shutdown or at resource cleanup timeout.

To create the mount handle data sets, perform the following tasks:

1. Allocate two empty VSAM KSDS data sets with the following attributes:
   - Starting with offset 0, the first 16 bytes in the record are the prime key field.
   - The maximum record length of the mount handle data set is 2000 bytes, and the average record length is 1700 bytes.

   ```
   DEFINE CLUSTER (NAME(mount_handle_data_set_name) -
               VOL(vsam_volume_name) -
               CYL(1 1) -
               INDEXED -
               REUSE -
               KEYS(16 0) -
               SHAREOPTIONS(1 3) -
               RECSZ(1700 2000))
   ```

   See Appendix L, "GFSAMHDJ sample code for creating NFS mount handle data sets and lock data sets," on page 531 for sample JCL showing how to create the mount handle data sets.
2. Specify these two data sets to the FHDBASE DD statement and FHDBASE2 DD statement in the MVSNFS procedure (see Figure 23).
3. The server switches data sets after resource cleanup has run.
4. Resource cleanup is done at Network File System shutdown and resource cleanup timeout.

Figure 23 shows how to specify the mount handle data set on the FHDBASE DD statement and FHDBASE2 DD statement on the MVSNFS procedure.

```
//*     FHDBASE AND FHDBASE2 ARE
//*     THE MOUNT HANDLE DATA SETS.
//*     THEY NEED TO BE PREALLOCATED
//*     AS EMPTY VSAM KSDS DATA SETS.
//*     THEY WILL BE USED ALTERNATELY.
//*     SAMPLE JCL CAN BE FOUND IN HLQ.NFSSAMP(GFSAMHDJ).
//*
//FHDBASE  DD   DISP=SHR,DSN=MVSNFS.FHDBASE
//FHDBASE2 DD   DISP=SHR,DSN=MVSNFS.FHDBASE2
```

Figure 23. Specifying the mount handle data set in the MVSNFS procedure

Lock data sets

Lock data sets are VSAM key-sequenced data sets that record the client host IP address and, for NFS V4, the client identification as well as the client host name. Following a server outage, the z/OS NFS server reads the new lock data sets during initialization to determine which clients can reclaim locks. The z/OS Network File System alternates between two data sets to record this information; only one data set is used at a time, and it is switched at shutdown, at resource cleanup timeout, and at the end of the grace period at server startup.

Note: The lock data sets must always be allocated, even if **nonlm** is specified in the site attributes.

To create the lock data sets, perform the following tasks:

1. Allocate two empty VSAM KSDS data sets, on separate DASD volumes to reduce the possibility that a single failure would result in the loss of both data sets. Allocate them with the following attributes:
   - Starting with offset 0, the first eight bytes in the record are the prime key field.
   - The maximum record length of the lock data set is 2000 bytes, and the average record length is 1700 bytes.

     ```
     DEFINE CLUSTER (NAME(lock_data_set_name) -
                 VOL(vsam_volume_name) -
                 CYL(1 1) -
                 INDEXED -
                 REUSE -
                 KEYS(8 0) -
                 SHAREOPTIONS(1 3) -
                 RECSZ(1700 2000))
     ```

     See Appendix L, “GFSAMHDJ sample code for creating NFS mount handle data sets and lock data sets,” on page 531 for sample JCL showing how to create the lock data sets.
2. Specify these two data sets to the LDBASE DD statement and LDBASE2 DD statement in the MVSNFS procedure (see Figure 24).
3. The server switches data sets after resource cleanup has run.
4. Resource cleanup is done at Network File System shutdown, resource cleanup timeout, and at the end of the grace period at server startup.

Figure 24 shows how to specify the lock data set on the LDBASE DD statement and LDBASE2 DD statement on the MVSNFS procedure.

```
//*     LDBASE AND LDBASE2 ARE
//*     THE LOCK DATA SETS.
//*     THEY NEED TO BE PREALLOCATED
//*     AS EMPTY VSAM KSDS DATA SETS.
//*     THEY WILL BE USED ALTERNATELY.
//*     SAMPLE JCL CAN BE FOUND IN HLQ.NFSSAMP(GFSAMHDJ).
//*
//LDBASE  DD   DISP=SHR,DSN=MVSNFS.LDBASE
//LDBASE2 DD   DISP=SHR,DSN=MVSNFS.LDBASE2
```

Figure 24. Specifying the lock data set in the MVSNFS procedure

Converting data between ASCII and EBCDIC - NFS V2 and V3 only

For text processing mode, with the NFS version 2 and 3 protocols, data is converted between EBCDIC and ASCII. No double byte character set (DBCS) or multiple byte character set (MBCS) forms of data are converted.

Customizing the translation table

You can customize the translation table for NFS using the processing attribute **xlat(member_name)**. The parameter **(member_name)** is the member name of a PDS or PDSE containing the customized translation table. This attribute can be specified either in the mount operation or in the attribute file. It can be specified on a file operation but is ignored, only the mount or the xlat value takes effect.

If the processing attribute, **xlat**, is not specified in the attribute file, the NFS internal translation table is used as the installation default translation table. When the **xlat(member_name)** processing attribute is specified in the attribute file, this customized translation table becomes the installation default translation table. The NFS internal translation table is derived from EBCDIC code page 0037 and ISO 8859-(ASCII). RPC arguments, such as filenames, are always translated by the installation default translation table. Data shipped with RPCs are translated by the mount specified translation table, if any. Otherwise, they are also translated by the installation default translation table.

A mount request with processing attribute, **xlat**, specified overrides the installation default translation table.

When accessing z/OS UNIX files, you must specify the OEMVS311 translation table or your customized translation table either in the mount request or in the default attributes. The OEMVS311 table translates ASCII (ISO 8859-1) to and from EBCDIC (1047 - z/OS UNIX). TCP/IP for MVS version 3.1 provides the OEMVS311 table. This table translates the UNIX line terminator (**lf**) to the z/OS UNIX line terminator (**nl**).

See *z/OS Communications Server: IP Configuration Reference* for more information about creating and customizing your own translation tables.

Enabling the xlat processing attribute

A DD statement, NFSXLAT, is required in the Network File System startup procedure to enable the **xlat(member_name)** processing attribute:

```
//NFSXLAT        DD       DSN=dataset_name,DSP=SHR
```

where

dataset_name
Specifies the name of a PDS or PDSE whose member contains the customized translation table.

A PDS or PDSE, dataset_name, is created by the CONXLAT utility whose member contains the customized translation table.

Notes:

1. See *z/OS Communications Server: IP Configuration Reference*, “Using Translation Tables,” for more information about creating and customizing your own translation tables.
2. You can edit or modify the translation table from your own or from a member in the tcpip.SEZATCPX data set and then use CONVXLAT utility to convert the source table into binary format. The CONVXLAT utility can take a PDS or PDSE as input, and its output data set can be physical sequential, PDS or PDSE.
3. The Network File System only supports PDS and PDSE. A sequential data set must be copied to either a PDS or PDSE member.
4. The Network File System does not support the translation for multiple-byte character sets.
5. Sample steps for creating the xlat member:
   a. Run the TCPIP CONVXLAT utility to create a physical sequential (PS) data set with DSORG=PS, RECFM=F, LRECL=256, BLKSIZE=256;

      ```
      "convxlat" 'tcpip.sezatcpx(standard)' 'hlq.xlat.output'
      ```

   b. Allocate a PDS data set with DSORG=PO, RECFM=F, LRECL=256, BLKSIZE=256; copy the CONVXLAT output data set as a member in the PDS data set
   c. Allocate the xlat member in the z/OS NFS startup procedure.

Updating z/OS system data sets for the server

Update the following z/OS system data sets to accommodate the z/OS NFS server:

- PARMLIB updates

  Add the data set defined in the GFSAPROC STEPLIB containing the z/OS NFS server library to the system's APF authorization list (IEAAPFxx). A sample cataloged procedure named GFSAPROC is provided as a member of the sample library NFSSAMP, see “Sample z/OS NFS server startup procedures” on page 495.
- PROCLIB updates

  A sample cataloged procedure named GFSAPROC is provided as a member of the sample library NFSSAMP, see “Sample z/OS NFS server startup procedures” on page 495. Modify the MVSNFS procedure and place it in your system PROCLIB. Add the DD statements:

  ```
  EXPORTS as the DD for the exports data set
  NFSATTR as the DD for the attributes data set
  FHDBASE and FHDBASE2 as the DD for the mount handle data set
  NFSXLAT as the DD to enable the xlat processing attribute
  NFSLOG1 as the DD for the primary log data set
  NFSLOG2 as the DD for the secondary log data set
  SYSxDUMP as the DD for the SYSxDUMP data set ('x' = U or M)
  LDBASE and LDBASE2 as the DD for the lock data sets
  ```

Allocating the z/OS NFS server log data sets

For information about allocating the z/OS NFS server primary and secondary log data sets, see Appendix K, “Capturing diagnostic information using z/OS NFS log data sets and from other components,” on page 523.

Side file data set

A side file data set is allocated as a fully-qualified MVS data set name.

- Record format of FB 80 is required.
- Data set organization of Physical Sequential is required.
- The name must follow the naming convention:

  "USERID.MLQ.NFS.MAPPING"

  where:

  USERID
  can be any valid userid.

  MLQ
  is optional; if present, may be any valid MVS qualifiers

  NFS.MAPPING
  are required low-level qualifiers

The last two qualifiers of the data set name must be NFS.MAPPING .

A sample side file, GFSAPMAP, is provided in the SYS1.NFSSAMP library. For information about side files, see “File extension mapping” on page 40, “Processing attributes syntax” on page 121, “Site attributes syntax” on page 134, and Appendix E, “NFS system server sample attribute table,” on page 465.

Modifying tcpip.ETC.RPC and etc/rpc

Add the entries in Table 31 on page 186 to the tcpip.ETC.RPC and etc/rpc files for the services provided by the z/OS NFS server:

Table 31. Modifying tcpip.ETC.RPC and etc/rpc

Service	Number	Alias	Description
nfsd	100003	nfs nfsprog	# Network File System daemon
mountd	100005	mount showmount	# Mount daemon
nlockmgr	100021	nlm nfs_lockd	# Network Lock Manager
status	100024	nsm nfs_statd	# Network Status Monitor
mvsmount	100044	nfs_mvsmnt	# MVSmount daemon (for mvslogin, mvslogout)
showattr	100059	nfs_showattr	# showattr daemon
pcnfsd	150001	nfs_pcnfs	# pcnfs daemon

Setting up a user-specified port range

The /etc/services file must define the port number entries for services **mountd**, **mvsmount**, **pcnfsd**, **showattr** , network status monitor (**status**), and network lock manager (**nlockmgr**) of the z/OS NFS server. Figure 25 outlines the port numbers for these services, with contiguous port numbers 2043-2049 as examples.

```
# NFS server
#
#  Port 2049 must be used for nfsd.
#
#  Consecutive port numbers must be assigned for the NFS status,
#   nlockmgr, mountd, mvsmount, showattr, and pcnfsd services.
#   The example belows uses ports 2043-2048.
#
#  When the NFS callback function is being used the services
#   nfsscb_b and nfsscb_e should reserve 100 consecutive ports.
#   The example below uses port 10300 for the beginning port
#   and port 10399 as the ending port.
#  For additional information see the Network File System Guide
#   and Reference manual.
#
Service       port/protocol  Alias        Description
status          2043/tcp   nfs_statd     # NFS State daemon (NSM)
status          2043/udp   nfs_statd     # NFS State daemon (NSM)
nlockmgr        2044/tcp   nfs_lockd     # NFS Lock daemon (NLM)
nlockmgr        2044/udp   nfs_lockd     # NFS Lock daemon (NLM)
mountd          2045/tcp   mount         # NFS mount daemon
mountd          2045/udp   mount         # NFS mount daemon
mvsmount        2046/tcp   nfs_mvsmnt    # NFS mvsmount daemon
mvsmount        2046/udp   nfs_mvsmnt    # NFS mvsmount daemon
showattr        2047/tcp   nfs_showattr  # NFS showattr daemon
showattr        2047/udp   nfs_showattr  # NFS showattr daemon
pcnfsd          2048/udp   nfs_pcnfs     # NFS pcnfsd daemon
nfsd            2049/tcp   nfs           # NFS server daemon
                                         #  - do not change
nfsd            2049/udp   nfs           # NFS server daemon
                                         #  - do not change
#
#  NFS Callback function port range
#
nfsscb_b       10300/tcp                 # NFSS callback port begin
nfsscb_e       10399/tcp                 # NFSS callback port end
nfsscb_b       10300/udp                 # NFSS callback port begin
nfsscb_e       10399/udp                 # NFSS callback port end
```

Figure 25. Modify /etc/services for mountd, mvsmount, pcnfsd, showattr, status, and nlockmgr

The user specified range of ports provides a flexible port range to accommodate programs such as a firewall that supports a range of ports for security purposes between the NFS Client and the NFS Server. Firewall security honors a limited port range so the NFS server has to allow the user to specify a few server ports in /etc/services to narrow the port ranges used for programs **mountd**, **mvsmount**, **pcnfsd** and **showattr**.

The tcpip.profile file must define the port range entries for services **nfsd**, **mountd**, **mvsmount**, **pcnfsd**, **showattr**, **status**, and **nlockmgr** of the z/OS NFS server. Figure 26 outlines the port ranges for these seven services with contiguous port numbers 2043-2049 as examples, starting with port 2043 for the network status monitor.

```
PORTRANGE 2043 7 UDP mvsnfs ; Reserved for startup JCL,
                            ;     mvsnfs
PORTRANGE 2043 7 TCP mvsnfs ; Reserved for startup JCL,
                            ;     mvsnfs
PORTRANGE 10300 100 UDP mvsnfs   ; Reserved for startup JCL,
                            ;     mvsnfs
PORTRANGE 10300 100 TCP mvsnfs   ; Reserved for startup JCL,
                            ;     mvsnfs
```

Figure 26. Modify tcpip.profile for z/OS NFS server services

Note: Check the z/OS USS BPXPRMxx parmlib member for INADDRANYPORT and INADDRANYCOUNT. Their range cannot include the z/OS NFS server port ranges for the server to initialize. See *z/OS Communications Server: IP Configuration Guide* for more details.

If the z/OS NFS server is started in the TCP/IP Autolog section, then the NOAUTOLOG parameter should be specified on the PORTRANGE statement, unless there will always be listeners/sockets on all ports defined in the statement. For additional information, refer to the section on AUTOLOG in *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Add the following two lines in /etc/rpc on the client side to see mvsmount and showattr information:

```
mvsmount         100044   mvsmount
showattr         100059   showattr
```

The query result from the client side will show the following messages:

```
# rpcinfo -p hostname
    program   vers     proto      port    service
    100000    2        tcp        111     portmapper
    100000    2        udp        111     portmapper
    100024    1        udp        2043    status
    100024    1        tcp        2043    status
    100021    1        udp        2044    nlockmgr
    100021    1        tcp        2044    nlockmgr
    100021    3        tcp        2044    nlockmgr
    100021    3        udp        2044    nlockmgr
    100021    4        tcp        2044    nlockmgr
    100021    4        udp        2044    nlockmgr
    100005    1        udp        2045    mountd
    100005    1        tcp        2045    mountd
    100005    3        udp        2045    mountd
    100005    3        tcp        2045    mountd
    100044    1        udp        2046    mvsmount
    100044    1        tcp        2046    mvsmount
    100059    2        udp        2047    showattr
    100059    2        tcp        2047    showattr
```

```
150001    1       udp       2048    pcnfsd
150001    2       udp       2048    pcnfsd
100003    2       udp       2049    nfs
100003    2       tcp       2049    nfs
100003    3       udp       2049    nfs
100003    3       tcp       2049    nfs
100003    4       tcp       2049    nfs
```

Configuring a secure z/OS NFS server

In order for the z/OS NFS version 4 server to be able to provide RPCSEC_GSS security authentication flavors such as krb5, krb5i and krb5p, the z/OS NFS server must be configured to communicate with the Kerberos facilities. To do so, complete the following steps.

We are assuming that a properly configured KDC is already setup in your environment. If a KDC is not already configured please see appendix M. This section is split up into two parts to include specific examples using a KDC provided by “Security Server and Integrated Security Services” (RACF) provided by IBM and generic examples for non RACF KDC’s. These steps assume that Resource Access Control Facility (RACF) is available in the system. If you have a different but equivalent external security manager, please refer to its product documentation for instructions. A domain name server (DNS) resolver should also be available to the z/OS system in order to enable the security feature. Otherwise message GFSA735I is shown during startup of the secure z/OS NFS server. Since there are many options to set up a DNS resolver, such as /etc/resolv.conf or GLOBAL TCPIPDATA, specific examples are not given here. For more information on setting up a DNS resolver, see *z/OS Communications Server: IP Configuration Guide*.

1. The Kerberos key distribution center (KDC) must be running, and must contain the z/OS NFS server’s principal before the secure z/OS NFS server starts. If the KDC is not set up correctly, whether the z/OS NFS server can start depends on the hfssec, mvssec, and pubsec attribute settings. If any of these three attributes also contains the sys security flavor in addition to any of the Kerberos flavors, the z/OS NFS server is started with message GFSA737I and functions with the sys security flavor only. On the other hand, if none of the hfssec, mvssec, or pubsec attributes contains the sys security flavor and the KDC is not available, the message GFSA736E is shown and z/OS NFS server does not start. The KDC can be running on z/OS, either on the same host as the z/OS NFS server itself or remotely from the z/OS NFS server. It can also be a KDC running on other platforms, for example, a SUN Solaris system or any other platform.

   For a brief description on how to setup a z/OS KDC, see “Setting up a Kerberos Key Distribution Center” on page 537, or refer to *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926 for more advanced details. For setting up other platforms' KDCs, refer to the specific platform’s documentation.
2. Define local realm and default policy. For example, issue the following TSO command:

   ```
   RDEFINE REALM KERBDFLT KERB(KERBNAME(KRB390.IBM.COM) PASSWORD(password)
   ```

   Note: “KRB390.IBM.COM” should match the Kerberos REALM of the KDC.
3. Define IRR.RUSERMAP and grant READ authority to all system users, issuing TSO commands:

   ```
   RDEFINE FACILITY IRR.RUSERMAP UACC(READ) SETROPTS RACLIST (FACILITY)
   REFRESH PERMIT IRR.RUSERMAP CLASS(FACILITY) ID(mvsnfs) ACCESS(READ)
   SETROPTS CLASSACT (FACILITY)
   ```

Note: If “mvsnfs” is the RACFID of the z/OS NFS server. It is recommended that you add this path in the z/OS UNIX /.profile:

```
"PATH=/usr/lpp/skrb/bin:$PATH"
```

and export the "PATH."

4. Create RACF user ids with Kerberos segments for the z/OS NFS server.

   For example, to add a RACF ID for the z/OS NFS Server (if one does not already exist), issue the TSO command:

   ```
    AU (mvsnfs) OWNER(owner) OMVS(UID(101))
   ```

   To add the needed Kerberos segment to this user definition, issue the TSO command:

   ```
   ALTUSER mvsnfs PASSWORD(password) NOEXPIRED
           KERB(KERBNAME(nfs/hostname.domain))
   PASSWORD USER(mvsnfs) NOINTERVAL
   ```

   If the Kerberos segment is not defined correctly to RACF, the following error message appears on the server when a NFS client tries to mount to z/OS NFS server with Kerberos.

   ```
   GFSA728E SAF APPLICATION USER MAPPING FAILED WITH SAF RETURN CODE 8,
     RACF RETURN CODE 8, AND RACF REASON CODE 16
   ```

   Notes:

   a. If you are NOT using a SAF KDC skip the RACF commands in this step and create a principal ‘nfs/hostname.domain’ according to your venders KDC documentation.
   b. The ALTUSER command converts the password to upper case if the MIXEDCASE SETROPTS option is not set. If MIXEDCASE is not set, you must ensure that the upper case value is used when you request an initial ticket. The principal name is not converted to upper case and the realm name is not included. You must change the password for the user in order to create the Kerberos secret key.
   c. The Kerbname must be the fully qualified hostname and domain. For example hostname.domain could be “host1.ibm.com”
   d. Including PASSWORD option “NOINTERVAL” prevents the password from expiring.

5. The z/OS NFS server requires the Kerberos configuration file “krb5.conf” be configured to match your sites Kerberos environment.

   Sample /etc/skrb/krb5.conf file to be put on the z/OS NFS server host:

   ```
   [libdefaults]
   default_realm = KRB390.IBM.COM
   kdc_default_options = 0x40000010
   use_dns_lookup = 0
   default_tkt_enctypes = des-cbc-crc
   default_tgs_enctypes = des-cbc-crc

   [realms]
   KRB390.IBM.COM = {
   kdc = dcesec4.krb390.ibm.com:88
   kpasswd_server = dcesec4.krb390.ibm.com:464
   admin_server = dcesec4.krb390.ibm.com:749
   }
   KRB2000.IBM.COM = {
   kdc = sstone1.krb2000.ibm.com:88
   admin_server = sstone1.krb2000.ibm.com:749
   }
   ```

```
[domain_realm] .
krb390.ibm.com = KRB390.IBM.COM .
krb2000.ibm.com = KRB2000.IBM.COM
```

Note: If the z/OS NFS server will be supporting multiple platforms of NFS client, IBM recommends using des-cbc-crc encryption types only for compatibility, as shown in this example.

6. 7. Generate the keytab from the KDC and put it in /etc/skrb of the z/OS NFS server unless otherwise defined. For more detailed examples of generating keytabs see “Setting up a Kerberos Key Distribution Center” on page 537 or in *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926.

   From the omvs shell, the system administrator must add the principal "nfs/hostname.domain" into the keytab. If /etc/skrb/krb5.keytab does not exist, create a new one. For example for a SAF or NDBM KDC

   ```
   IBMUSER:/ :> keytab add nfs/hostname.domain -p password -k /etc/skrb/krb5.keytab -v 1
   ```

   For example for a Unix KDC in kadmin:

   ```
   Kadmin: ktadd –e des-cbc-crc:normal –k /etc/krb5/krb5.keytab nfs/hostname.domain
   ```

   Notes:

   a. The "password" in this step must match the "password" entered in step 4, when the principal was added to the SAF KDC Kerberos database. This principal is used to authenticate the z/OS NFS Sertver to the KDC.
   b. The key version used to create the keytab must be the same key version as in the racf database. The “-v” option of the keytab command is used to specify the key version when adding a principal to a keytab. Issue the following RACF command to see the current key version:

      ```
      LU mvsnfs NORACF KERB
      ```

   c. The password used with the “keytab” command is case sensitive. If mixed case password support is not in effect you must enter the password in uppercase

7. For systems with multiple TCPIP stacks you must create the keytab with principals for each stack. If a stack is part of a different REALM then keys will need to be added to the keytab from each KDC. Cross REALM trusts must also be created.

   ```
   IBMUSER:/:>klist -k
   Key table: /etc/skrb/krb5.keytab
   Principal: nfs/host1.domain.com@ KRB390.IBM.COM
   Key version: 4
   Principal: nfs/host2.domain.com@ KRB390.IBM.COM
   Key version: 4
   Principal: nfs/host3.domain.com@ KRB2000.IBM.COM
   Key version: 2
   ```

8. If there is any multi-realm setup in the environment, the z/OS NFS server needs to have the foreign principals mapped to a RACF ID.

   For example: To map a foreign principal “fprinc” in “KRB2000.IBM.COM” to RACF ID “fprealm2”, issue the TSO commands:

   ```
   ADDUSER (fprealm2) OWNER(owner) OMVS(UID(102))
   ALTUSER fprealm2 PASSWORD(password) NOEXPIRED
   PASSWORD USER(fprealm2) NOINTERVAL
   RDEFINE KERBLINK /.../KRB2000.IBM.COM/fprealm2 APPLDATA(''fprealm2')
   ```

   To map the entire foreign realm (every principal in the trusted foreign realm) to a RACF user, issue the TSO command:

```
RDEFINE KERBLINK /.../KRB2000.IBM.COM/ APPLDATA('fprealm2')
```

Note: the /.../ and trailing slash are required. “KRB2000.IBM.COM” is the foreign realm.

9. Start the z/OS NFS server. If set up is correct, the following message should be shown:

```
GFSA730I NETWORK FILE SYSTEM SERVER KERBEROS INITIALIZATION SUCCESSFUL
```

Notes:

1. These are the minimal requirements to set up a secure z/OS NFS server in order for it to communicate with Kerberos facilities. For more advanced configurations, please see *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926.
2. If z/OS NFS server is configured to use a KDC that resides on a remote host, the local KDC procedure (for example, skrbkdc) on the same host as the z/OS NFS server should not be started.

Using dynamic client IP addressing

By default, the z/OS NFS server expects to communicate with clients based on a static client IP address. The server can also use the dynamic host configuration protocol (DHCP) to accept dynamic client IP address changes. To use dynamic client IP addressing, specify the **dhcp** server site attribute. The default attribute, **nodhcp**, tells the server to use the static IP algorithm.

To use dynamic IP addressing, the client must:

- Have a constant host name that the NFS server can identify it by.
- Dynamically update the authentication DNS (dynamic name server) with new IP addresses whenever they change.
- Maintain the TTL (time to live) value that the authentication DNS server specifies to any caching DNS server, based on the frequency with which system IP addresses might change. The larger the TTL value, the greater the possibility that the caching DNS server will have obsolete information. If dynamic addressing is used, the TTL value should be small, ideally zero, but a small value defeats the benefit of caching, so a compromise must be set with the understanding that cached values can become obsolete during the TTL interval and report incorrect information to querying systems like the NFS server.

Regardless of the **dhcp/nodhcp** attribute value, the z/OS NFS server itself continues to have a static IP address.

Terminal ID based restricted MVSLOGIN

When the z/OS NFS Server is used in SECURITY (**saf** or **safexp**) mode, it is necessary for users on NFS clients to issue an NFS Client Enabling Utility MVSLOGIN command from the NFS client system before they can access any files on the NFS Server. Normally, assuming the user has a valid z/OS userid and password, this is not a problem and will successfully provide the user with access to the z/OS system through NFS. However, with the appropriate RACF configuration specifications, the z/OS NFS server also provides the ability to restrict MVSLOGINs based on an NFS client's IP address.

In order to support this capability, the z/OS NFS server transforms an NFS client's IP address into an 8-byte character string, which is then used as the Terminal ID (termid) for that NFS Client. Each decimal number of the IP address is transformed into two hex digits. For example:

IP address	is transformed into
12.15.16.32	0C0F1020
9.157.161.12	099DA10C

To use this capability, the z/OS system administrator must:

1. Activate the RACF class TERMINAL. This is done with the RACF command:
   ```
   SETROPTS CLASSACT(TERMINAL) RACLIST(TERMINAL)
   ```
2. Define the proper resource in the TERMINAL class. This is done with the RACF command:
   ```
   RDEFINE TERMINAL termid UACC(NONE)
   ```

   where *termid* is the terminal Id as generated by the z/OS NFS server using the algorithm cited above.

   Assume a *termid* value of 099DA10C is specified, then NFS client with IP address 9.157.161.12 cannot successfully execute the MVSLOGIN NFS Client Enabling Utility for users, which have NONE access for the *termid* class.
3. Refresh the RACF class TERMINAL. This is done with the RACF command:
   ```
   SETROPTS RACLIST(TERMINAL) REFRESH
   ```
4. Grant permission to some users (for example, USER4 and USER5) from the NFS client with IP address 9.157.161.12 to successfully execute the MVSLOGIN NFS Client Enabling Utility. This is done with the RACF command:
   ```
   PERMIT 099DA10C CLASS(TERMINAL) ID(USER4 USER5) ACCESS(ALTER)
   SETROPTS RACLIST(TERMINAL) REFRESH
   ```

For more details on the RACF configuration specifications, see *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683.

This feature is supported by z/OS NFS only for IPv4 IP addresses in **saf** or **safexp** SECURITY mode, and only in NODHCP mode. The z/OS NFS Server does not support this capability for IPv6 IP addresses (because an IPv6 IP address is too large for this mapping algorithm), or in DHCP mode (because IP addresses change dynamically in DHCP mode).

Note: This feature is also supported with RPCSEC_GSS authentication. However, since mvslogin is no longer required with RPCSEC_GSS, the RACF authentication is done automatically based on the Kerberos segment of the RACF ID.

SERVAUTH based restricted MVSLOGIN

The z/OS NFS server relies on the z/OS Communications Server (CS) and RACF to protect several resources and to restrict access from a network, subnetwork, or particular IP address in the network. Using NETACCESS statements in a TCPIP profile, z/OS CS can map networks, subnetworks, and IP addresses to RACF resource names in the SERVAUTH class (see *z/OS Communications Server: IP Configuration Guide*, SC31-8775). Users that are not permitted access to a particular RACF resource are not allowed to execute MVSLOGIN from the corresponding network, subnetwork, or IP address.

User access to MVS data sets through the z/OS NFS Server can be protected/permitted restricted to/from some network, subnetwork, or IP address (see *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683).

To use this capability, the z/OS system administrator must:

1. Add the NETACCESS section in your TCPIP profile if it does not exist, and modify the NETACCESS section in your TCPIP profile to prevent/permit users from/to accessing a given network, subnetwork, or host.

   NETACCESS examples:

```
  NETACCESS    INBOUND    OUTBOUND ; check both ways
       9.11.117.218 255.255.255.255 SUN1       ; specific UNIX host
       9.11.117.108/32  MVSNFS     ; the z/OS NFS server Requires
                                   ; matching "PERMIT" to grant access
       192.168.0.0/16   CORPNET ; Net address
       192.168.113.19/32  HOST1    ; Specific host address
       192.168.113.0     255.255.255.0   SUBNET1 ; Subnet address
       192.168.192.0/24  CAMPUS ; Subnet address
       Fe80::6:2900:1dc:21bc/128 HOST2  ; IPv6 specific host address
       2001:0DB8:/16   GLBL  ; IPv6 global network
     DEFAULTHOME      HOME       ; Required local zone
     DEFAULT 0        DEFZONE    ;Optional Default security zone
     ENDNETACCESS
```

2. Define and activate a RACF profile for each resource specified in the SERVAUTH class via th NETACCESS statement. Issue the following RACF commands (see *z/OS Security Server RACF Security Administrator's Guide*):

```
RDEFINE SERVAUTH (EZB.NETACCESS.SYSTEM1.TCPIPSJ.MVSNFS)
RDEFINE SERVAUTH (EZB.NETACCESS.SYSTEM1.TCPIPSJ.SUN1)
RDEFINE SERVAUTH (EZB.NETACCESS.SYSTEM1.TCPIPSJ.HOME)
RDEFINE SERVAUTH (EZB.NETACCESS.SYSTEM1.TCPIPSJ.DEFZONE)
SETROPTS CLASSACT(SERVAUTH) REFRESH RACLIST(SERVAUTH)
```

   where SYSTEM1 is the *sysname*, TCPIPSJ is the *tcpname*, and MVSNFS is the *saf_resname* as described below.

   The corresponding RACF profile name has the following format (see *z/OS Communications Server: IP Configuration Reference*, SC31-8776):

```
 EZB.NETACCESS.sysname.tcpname.saf_resname
```

   where

   EZB.NETACCESS
   is constant.

   sysname
   is the value of the MVS &SYSNAME. system symbol.

   tcpname
   is the name of the procedure used to start the TCP stack.

   saf_resname
   is 8-character value from the NETACCESS section.

   An asterisk is allowed as *sysname* and *tcpname*. For example:

```
 EZB.NETACCESS.*.*.CORPNET
 EZB.NETACCESS.*.*.SUBNET1
```

3. To allow an NFS client to create connections with the z/OS NFS Server, socket activity for the NFS Server and the Port mapper, (or RPCBIND) must be permitted with the following RACF commands:

```
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.MVSNFS  ACCESS(ALTER) CLASS(SERVAUTH) ID(IBMUSER)
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.SUN1    ACCESS(ALTER) CLASS(SERVAUTH) ID(IBMUSER)
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.HOME    ACCESS(ALTER) CLASS(SERVAUTH) ID(IBMUSER)
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.DEFZONE ACCESS(NONE)  CLASS(SERVAUTH) ID(IBMUSER)
```

   In this example, assuming IBMUSER is the owning ID of both NFS Server and RPCBIND, it is needed to grant ALTER access to IBMUSER.

4. To allow an NFS client to access the z/OS NFS Server with specific RACF ID (for example, USER3), issue the following RACF PERMIT commands:

```
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.MVSNFS  ACCESS(ALTER) CLASS(SERVAUTH) ID(USER3)
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.SUN1    ACCESS(ALTER) CLASS(SERVAUTH) ID(USER3)
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.HONME   ACCESS(ALTER) CLASS(SERVAUTH) ID(USER3)
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.DEFZONE ACCESS(ALTER) CLASS(SERVAUTH) ID(USER3)
SETROPTS CLASSACT(SERVAUTH) REFRESH RACLIST(SERVAUTH)
```

NFS client SUN1 (9.11.117.218) can now execute MVSLOGIN with RACF ID USER3, but not other RACF users.

5. To prevent an NFS client to access the z/OS NFS Server with specific RACF ID (for example, USER3), issue the following RACF PERMIT commands:

```
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.DEFZONE ACCESS(NONE)  CLASS(SERVAUTH) ID(USER3)
PERMIT EZB.NETACCESS.SYSTEM1.TCPIPSJ.DEFZONE ACCESS(ALTER) CLASS(SERVAUTH) ID(USER5)
SETROPTS CLASSACT(SERVAUTH) REFRESH RACLIST(SERVAUTH)
```

NFS client SUN1 (9.11.117.218) can now execute MVSLOGIN with RACF ID USER5, but not USER3 and other RACF users.

By using conditional PERMIT commands, the system administrator can restrict access to a data set profile (for instance 'USER2.*') for USER5. The RACF will permit the access only if USER5 executes MVSLOGIN from SUBNET1 (IP address 192.168.113.19).

```
PERMIT 'USER2.*' ID(USER5) ACCESS(ALTER)
  WHEN(SERVAUTH(EZB.NETACCESS.*.*.SUBNET1))
```

For more informaiton, see *z/OS Security Server RACF Command Language Reference*, SA22-7687.

Notes:

1. The z/OS NFS server supports this capability only in **saf** or **safexp** SECURITY mode.
2. SERVAUTH supports both IPv4 and IPv6 modes.
3. To change between TERMID and SERVAUTH will require user configuration changes to switch between TERMINAL class security specification and SERVAUTH class specification, respectively.
4. This feature is also supported with RPCSEC_GSS authentication. However, since mvslogin is no longer required with RPCSEC_GSS, the RACF authentication is done automatically based on the Kerberos segment of the RACF ID.

Data Labeling

The z/OS NFS server supports the RACF Data Labeling option MLNAMES (also known as name-hiding). For more details on this option, see *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683.

This option is activated by RACF command:

```
SETROPTS MLNAMES
```

and is deactivated by RACF command:

```
SETROPTS NOMLNAMES
```

When this option is active MVS data set names will be hidden from NFS users who do not have at least READ access to the data sets. Therefore, it may change the contents of an MVS data set index list produced by the **ls -l** command.

Notes:

1. The z/OS NFS server supports this option only in **saf** or **safexp** SECURITY mode.
2. This function only applies to MVS data set access, not to z/OS Unix file access.
3. The name-hiding function can degrade system performance because it requires authorization checks for every object for which a non-SPECIAL user attempts to list the name.

Using multiple TCP/IP stacks

Configuring multiple NFS servers with multiple TCP/IP stacks

You can perform the following tasks to set up the NFS servers for multiple TCP/IP stacks:

- Invocation
- Example procedure to start a NFS server in a multiple server environment
- User interactions
- Errors
- Messages and codes

An NFS server can exploit the ability of the z/OS Communication Server to configure up to eight TCP/IP stacks simultaneously. Each TCP/IP stack can support only one NFS server. All NFS servers have their own IP-address and work independently of each other with each connecting to a specific transport provider. So, each NFS server will use its own unique set of data sets for mount handle database, error log, and startup procedures.

The client works with any NFS server as an independent host. At startup, the client selects an NFS server using the servers IP-address or HOST-NAME on the mount parameter. On shutdown of one of the NFS servers, all the clients connected to that server will be forced to make new connections with another NFS server and to repeat the startup procedures such as mvslogins and mount connections.

Multiple NFS server support provides an environment on z/OS where applications can have system flexibility by running a NFS server on each LPAR of one z/OS system. This lets you, for example, have a production and test NFS server run on one z/OS system. The use of multiple NFS servers also provides the ability to define separate security-schemes, to separate workload on different NFS servers, and use separate attribute definitions.

The z/OS NFS Server is a generic server (refer to the section entitled "Generic server versus server with affinity for a specific transport provider" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775). The z/OS NFS Server relies on another generic server that is the z/OS PORTMAP or z/OS RPCBIND. When configuring a z/OS NFS Server as a generic server, the z/OS PORTMAP or z/OS RPCBIND should be configured as a generic server. When configuring z/OS NFS Servers as multiple servers with transport affinity, multiple z/OS PORTMAP or z/OS RPCBIND should be configured with transport affinity. Mixing generic servers along with servers with transport affinity is not recommended and could lead to undesirable results.

Invocation: To run multiple NFS servers on one z/OS system, it is necessary to have a corresponding number of active TCP/IP stacks each with their own portmapper or rpcbind. A properly configured BPXPRMxx with CINET is required. The NFS server and TCP/IP startup procedures for each TCP/IP stack should have different names.

Example CINET configuration in BPXPRMxx to start an NFS server in a multiple server environment:

```
 FILESYSTYPE TYPE(CINET) ENTRYPOINT(BPXTCINT)
   NETWORK TYPE(CINET)
         DOMAINNAME(AF_INET)
         DOMAINNUMBER(2)
         MAXSOCKETS(64000)
         INADDRANYPORT(4901)
         INADDRANYCOUNT(100)
   NETWORK TYPE(CINET)
         DOMAINNAME(AF_INET6)    /* activate IPv6  */
         DOMAINNUMBER(19)
SUBFILESYSTYPE TYPE(CINET) NAME(TCPIPRX) ENTRYPOINT(EZBPFINI) DEFAULT
SUBFILESYSTYPE TYPE(CINET) NAME(TCPIPRY) ENTRYPOINT(EZBPFINI)
```

Each NFS server startup procedure needs to have the following change:

1. Add the envar parameter with the _BPXK_SETIBMOPT_TRANSPORT environment variable to point to the TCP/IP startup procedure.
2. SYSTCPD DD statement to point to its TCP/IP stack profile. See the example in "Example procedure to start an NFS server in a multiple server environment."

Example procedure to start an NFS server in a multiple server environment: The following contains a sample procedure to start an NFS server in a multiple server environment.

```
//MVSNFS PROC MODULE=GFSAMAIN,PARMS='INFO',
//          NFSPRFX=MVSNFS,TCPIP=TCPIP.OS390R10
//GFSAMAIN EXEC PGM=&MODULE,
//          REGION=0M,
//          TIME=1440,
//*    Use environment variable _BPXK_SETIBMOPT_TRANSPORT to
//*     set affinity for a specific TCP/IP stack. TCPIPR10 is name of
//*     start procedure for a selected TCP/IP stack.
//*
//      PARM=('ENVAR("_BPXK_SETIBMOPT_TRANSPORT=TCPIPR10")',
//           '/&PARMS?)
//*
//*     Define dataset name of selected TCP/IP stack.
//SYSTCPD DD DISP=SHR,DSN=&TCPIP..TCPIP.DATA
//STEPLIB DD DISP=SHR,DSN=USER1.LOADLIB1
//SYSPRINT DD SYSOUT=*
//*
//*SYSMDUMP DD DISP=SHR,DSN=&NFSPRFX..SYSMDUMP
//OUTPUT DD SYSOUT=*
//SYSERR DD SYSOUT=*
//*
//*     Define dataset names which will be used by current NFS server
//NFSLOG1 DD DISP=SHR,DSN=&NFSPRFX..DFSLAB11.LOG11
//NFSLOG2 DD DISP=SHR,DSN=&NFSPRFX..DFSLAB11.LOG21
//NFSATTR DD DISP=SHR,DSN=&NFSPRFX..CNTL(NFSATTRV)
//EXPORTS DD DISP=SHR,DSN=&NFSPRFX..CNTL(EXPORT1)
//FHDBASE DD DISP=SHR,DSN=&NFSPRFX..DFSLAB11.FHDBASE1
//FHDBASE2 DD DISP=SHR,DSN=&NFSPRFX..DFSLAB11.FHDBASE3
```

User interactions: For more information about configuring multiple TCP/IP stacks, see *z/OS UNIX System Services Planning*, GA22-7800 and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Console operators will need to distinguish between different NFS server console messages received from multiple NFS servers on one z/OS system.

System administrators can use the new support to define separate security-schemes, separate workload on different NFS servers, and use separate attribute definitions.

Errors: Existing NFS Server console messages display the start procedure name for a specific NFS server.

Messages and codes: Each NFS Server console message displays the NFS server start procedure as in the following example:

```
GFSA320I(procname) NETWORK FILE SYSTEM SERVER INITIALIZATION FAILED:  text
```

Where (*procname*) is the name of the NFS server start procedure.

Configuring a single NFS server with multiple TCP/IP stacks

You can perform the following tasks to set up an NFS server for multiple TCP/IP stacks:

- Invocation
- Example procedure to start a NFS server in a multiple server environment
- User interactions
- Errors
- Messages and codes

An NFS server can exploit the ability of the z/OS Communication Server to configure up to eight TCP/IP stacks simultaneously. An NFS server can interact with multiple TCP/IP stacks. A single NFS server will use the IP addresses associated with each stack.

The client works with any NFS server as an independent host. At startup, the client selects an NFS server using the server's IP address or HOST-NAME on the mount parameter. On shutdown of the NFS server, all the clients connected to that server will be forced to make new connections with another NFS server and to repeat the startup procedures, such as mvslogin and mount connections.

Invocation: To run a single NFS server with multiple TCP/IP stacks on one z/OS system, properly configure BPXPRMxx with CINET and a single rpcbind or portmapper for all stacks.

Note: The envar parameter should not be used when configuring a single NFS server with multiple stacks.

Example CINET configuration in BPXPRMxx to start a single NFS server in a multi stack environment:

```
FILESYSTYPE TYPE(CINET) ENTRYPOINT(BPXTCINT)
   NETWORK TYPE(CINET)
         DOMAINNAME(AF_INET)
         DOMAINNUMBER(2)
         MAXSOCKETS(64000)
         INADDRANYPORT(4901)
         INADDRANYCOUNT(100)
   NETWORK TYPE(CINET)
         DOMAINNAME(AF_INET6) /* activate IPv6 */
         DOMAINNUMBER(19)
SUBFILESYSTYPE TYPE(CINET) NAME(TCPIPRX) ENTRYPOINT(EZBPFINI) DEFAULT
SUBFILESYSTYPE TYPE(CINET) NAME(TCPIPRY) ENTRYPOINT(EZBPFINI)
```

User interactions: For more information about configuring multiple TCP/IP stacks, see *z/OS UNIX System Services Planning*, GA22-7800 and *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Errors: Existing NFS Server console messages display the start procedure name for a specific NFS server.

Messages and codes: Each NFS Server console message displays the NFS server start procedure as in the following example.

```
GFSA320I(procname) NETWORK FILE SYSTEM SERVER INITIALIZATION FAILED:  text
```

Where (*procname*) is the name of the NFS server start procedure.

Installing the client enabling commands

This section describes the tasks you must perform to install and port the client enabling commands. These tasks include retrieving commands for AIX and Sun Solaris. This section also includes information about porting the **mvslogin**, **mvslogout**, and **showattr** commands and dealing with different compilers and operating systems.

To enable client users to access the z/OS system and to display system attributes, you must install the **mvslogin**, **mvslogout**, and **showattr** commands on the client workstations. For some client machines, you might need to modify the code to port these commands so they run on your client machine. See "Porting the mvslogin, mvslogout, and showattr commands" on page 203. Before you install the commands, make sure that TCP/IP and File Transfer Protocol (FTP) are running both on z/OS and on the client.

Note: The z/OS NFS client utilities, including **mvslogin**, **mvslogout**, and **showattr**, are installed when the z/OS NFS client and TCP/IP are installed. The target library NFSCUTIL is a DDDEF to an existing z/OS UNIX directory (/usr/lpp/NFS/IBM) and will contain the client commands for the z/OS NFS client after installation. There is no need to port the z/OS NFS client utilities as you would for the remote NFS clients which use the z/OS NFS server.

Follow these installation procedures:

1. Delete any previous versions of the **mvslogin**, **showattr**, and **mvslogout** commands and their source code from your client workstation.
2. Retrieve the **mvslogin**, **mvslogout**, and **showattr** commands (in tarbin file format for AIX or UNIX, or in source code format for any other platform) from the *prefix*.NFSTARB data set, where prefix is an installation-specified variable. Use FTP with a binary transfer to send the tarbin or executable files to a client workstation for UNIX or AIX (no character conversion should be made). Use FTP with text transfer to send the source code of the three commands to a client workstation for any platform.
3. Use the tar utility to extract the files only if the client uses AIX or UNIX.
4. Compile the source code only if it is not in executable code format. (You might need to modify the code for your specific client machine.)
5. Make executable code versions of the commands available to all clients.

Recommendation: We recommend placing these commands on a LAN server (possibly in /usr/local/bin) that is available to many workstations, rather than installing them on each client workstation.

Table 32 is a list of all files stored in *prefix*.NFSTARB data set related to the server's client commands (**mvslogin**, **mvslogout**, and **showattr**):

Table 32. Files in the prefix.NFSTARB data set to download to clients

File Name	Download as:	Description	Client Environment
prefix.NFSTARB(GFSAWAIX)	client.tarbin (or any name)	Binary file	AIX, UNIX
Source Code for Commands			
prefix.NFSTARB(GFSAWMNT)	gfsawmnt.h	C header files	All [1]
prefix.NFSTARB(GFSAWSHO)	gfsawsho.h		
prefix.NFSTARB(GFSAWRP6)	gfsawrp6.h		
prefix.NFSTARB(GFSAWRS6)	gfsawrs6.h		
prefix.NFSTARB(GFSAWAXD)	gfsawaxd.c	C modules	All [1]
prefix.NFSTARB(GFSAWLIN)	gfsawlin.c		
prefix.NFSTARB(GFSAWLOU)	gfsawlou.c		
prefix.NFSTARB(GFSAWMOU)	gfsawmou.c		
prefix.NFSTARB(GFSAWSHA)	gfsawsha.c		
prefix.NFSTARB(GFSAWJCL)	makefile	All [1]	

Note:
1. For AIX or UNIX, you do not need to download every individual file if you download the GFSAWAIX file.

The following sample client screens show how to retrieve and create the **mvslogin**, **mvslogout**, and **showattr** client commands for the following platforms.

- AIX, Sun Solaris, and Linux; see page 200.

Appendix H, “Retrieval of source code for client enabling commands,” on page 501 shows how to retrieve the necessary source code to install the client commands on any platform except for an AIX or UNIX workstation.

Retrieving commands for AIX, Sun Solaris, and Linux

Figure 27 on page 201 shows how to retrieve the necessary files to install the client commands on workstations with AIX, Sun Solaris, or Linux. Use the "Make" section that is appropriate for your platform and omit the others. Use MVSHOST1 as the name of the z/OS host, and smith is a z/OS user ID.

Note: The compiler name can be changed to match the installed compiler name.

```
 $ ftp mvshost1
 Connected to mvshost1
 220-FTPSERVE at MVSHOST1, 01:44:24 on 6/02/07
 220 Connection closes if idle for more than 5 minutes.
 Name (mvshost1:w42dept): smith
 <Press ENTER key>
 331 Send password please.
 Password: password
 230 smith is logged on.
 ftp> bin
 200 Representation type is IMAGE.
 ftp> get 'prefix.nfstarb(gfsawaix)' client.tarbin
 200 Port request OK.
 125 Sending data set PREFIX.NFSTARB(GFSAWAIX)
 250 Transfer completed successfully.
 local: client.tarbin remote: 'prefix.nfstarb(gfsawaix)'
 213504 bytes received in 2.4 seconds (87 Kbytes/s)
 ftp> quit
 221 Quit command received. Goodbye.  $
 $ mkdir mvsnfs.client
 $ cd mvsnfs.client
 $ tar -xvf ../client.tarbin
 tar: record size = 20 blocks
 x ./makefile, 8234 bytes, 17 tape blocks
 x ./gfsawaxd.c, 13474 bytes, 27 tape blocks
   ⋮
 $
 $ touch *.*

(Following are Make sections for each supported platform - use the appropriate one

SUN:
$ make sun
gcc -D SUN -D _UTILS -D SOLARIS -lnsl -lsocket -o showattr gfsawsha.c gfsawaxd.c
gcc -D SUN -D _UTILS -D SOLARIS -lnsl -lsocket -o mvslogin gfsawlin.c gfsawmou.c
gcc -D SUN -D _UTILS -D SOLARIS -lnsl -lsocket -o mvslogout gfsawmou.c gfsawlou.c

Sun binaries for NFS Tools are made

LINUX:
make linux
gcc -D LINUX -D _UTILS -o showattr gfsawsha.c gfsawaxd.c
gcc -D LINUX -D _UTILS -o mvslogin gfsawlin.c gfsawmou.c
gcc -D LINUX -D _UTILS -o mvslogout gfsawmou.c gfsawlou.c

Linux binaries for NFS Tools are made

ZLINUX:
$ make zlinux
cc -D LINUX -D _UTILS -m31 -o showattr gfsawsha.c gfsawaxd.c
cc -D LINUX -D _UTILS -m31 -o mvslogin gfsawlin.c gfsawmou.c
cc -D LINUX -D _UTILS -m31 -o mvslogout gfsawmou.c gfsawlou.c

Binaries for zlinux NFS Tools are made
AIX:
$ make aix
gcc -D AIX -D _UTILS -D AIX_RT -o showattr gfsawsha.c gfsawaxd.c
gcc -D AIX -D _UTILS -D AIX_RT -o mvslogin gfsawlin.c gfsawmou.c
gcc -D AIX -D _UTILS -D AIX_RT -o mvslogout gfsawmou.c gfsawlou.c

AIX binaries for NFS Tools are made
 $ ./mvslogin mvshost1 smith
 (MVSNFS must be operational. Password Required on host side.)
 GFSA988I Remote host does not have AF_INET6 interface.
 GFSA973A  Enter MVS password: password
 GFSA955I  smith logged in ok.
 $
```

Figure 27. Retrieving the client enabling commands for AIX, Sun Solaris, and Linux

After retrieving the files of client enabling source code for the AIX or UNIX environment, follow these steps as shown in Figure 27 on page 201 to create the executable commands:

1. To make sure the source files have the current date. Some platforms do not have correct time stamps and cause make files to fail; issue the following command.

   ```
   touch *.*
   ```

2. To create the executable commands **mvslogin**, **mvslogout**, and **showattr**, issue one of these commands for the environment that IBM supports.

   make aix
   Creates executables for AIX.

   make sun
   Creates executables for Sun.

   make linux
   Creates executables for Linux.

   make zlinux
   Creates executables for zLinux.

Note: In order to build zLinux, you MUST have a 32-bit build environment installed on your zLinux system. If a 32-bit environment is not available, see "Porting on different compilers and operating systems" on page 204 for information on building 64-bit versions of the Client enabling commands.

Porting the mvslogin, mvslogout, and showattr commands

The z/OS NFS server supports any client machine that has an NFS client software package implemented according to the Sun NFS protocol. See "Tested clients for the z/OS NFS server" on page 14 for a list of supported platforms.

Note: NFS supports the authentication procedures of PCNFSD Version 1 and Version 2 protocols. If a PC client supports PCNFSD and keeps the UID and GID to each mount point base, you do not need to port the **mvslogin** command. See Appendix I, "PCNFSD protocol," on page 503 for details on PCNFSD support.

To port the **mvslogin**, **mvslogout**, and **showattr** commands successfully, you should understand the following:

- C language - The source code for these commands is written in C.
- System calls for your client machine's operating system - For example, the FAT file system under DOS only allows up to eight characters for file names, and up to three characters for file name extensions. AIX and UNIX do not have this restriction. Therefore, while mvslogout is a valid file name in an AIX or UNIX environment, it is too long to be a valid file name in a FAT file system under DOS.

  As another example, the way that you get mount information varies for different platforms. The **mount** command is in the following (or similar) format:

  ```
  mount <server>:<remote file system> <local mount point>
  ```

  The minimum information for porting the client enabling commands is:

  1. Server name
  2. Remote file system (high-level qualifier)
  3. Local Mount point
  4. UID and GID

  The system calls to get the information for porting the client enabling commands are platform-dependent. If you cannot find the information in the following types of documents for the platform, you must call the support telephone number for the platform and ask to speak with their NFS development department:

  – Operating system development toolkit
  – TCP/IP development toolkit
  – NFS development toolkit
- The source code for **mvslogin**, **mvslogout**, and **showattr**.

  For example, mvslogin tells the server the z/OS user id and its associated client UID number. This client UID number is expected to be passed to the server for all further client requests to the NFS. If the client user does not specify the z/OS user id and password on the mvslogin command, the z/OS login ID is taken from the login ID on the workstation with no password assumed. If authentication for this default login ID from the workstation fails, then mvslogin prompts the user to enter the z/OS login password.

Figure 28 on page 204 shows the common source files for the **mvslogin**, **mvslogout**, and **showattr** commands on all platforms:

Seven .c files

gfsawaxd.c
XDR encode and decode routines for attributes service.

gfsawclt.c
NFS protocol calls for **mvslogin** and **mvslogout**.

gfsawlin.c
Main program to generate **mvslogin** command.

gfsawlou.c
Main program to generate **mvslogout** command.

gfsawmcl.c
Create the client handle and initialize it.

gfsawmou.c
XDR protocol definitions for **mvslogin** and **mvslogout**.

gfsawsha.c
Main program to generate **showattr** command **mvslogout**.

Four .h files

gfsawmnt.h
Protocol definitions for **mvslogin** and **mvslogout**.

gfsawsho.h
Attribute definition and procedures.

gfsawrp6.h
IPv6 RPC library definitions.

gfsawrs6.h
IPv6 RPC support functions.

One makefile

Figure 28. Common source files

Porting on different compilers and operating systems

Procedures for porting vary for different C compilers and operating systems. Differences can occur during compiling, linking, and run time.

Compiling

The following items might vary for your client machine's operating system:

- Different set of compilation flags

  There are different sets of compilation flags based on compilers or operating systems. For example:

  – AIX (on RS/6000) has the unique flags _BSD, _SUN, and BSD_INCLUDES.

  – DOS compilers have different compiler models, which require the corresponding compiler flag (for example, -AL and -AS).
- Include files in different directories

  Because the include files can be installed differently based on the operating systems and their toolkits, the include files could be in different directories.
- Include file has a different name

Include files for the same or similar functions could have the same or similar file names. For example, DOS uses the file name “string.h”, and the other platforms use “strings.h”.

- System variables in different include files

  The system variables are usually in different include files, based on the operating systems. For example, to access the mount table some AIX and UNIX clients use mntent.h.
- System variables have different names

  The variables related to operating systems could have different variable names. For example; some AIX and UNIX clients use getuid to get the real UID.
- System variables have different structure

  The structures related to operating systems could be different. For example, DOS FAT file systems have file name length restrictions which cause them to have a different directory structure from AIX or UNIX.
- System variables not supported

  Some system variables are supported by one operating system but not another.
- Sequence of include files

  Some include files are based on the precedence of another include file. The prerequisite include file has to come before the other include files. For example, some Programming Libraries offer types.h which is based on C compiler sys/types.h. Therefore, #include <sys/types.h> should be before #include <types.h>.
- Mount information varies depending on the client operating system

  The information about mount points provided by vendors of the client operating systems and client TCP/IP products varies and might not always be complete. To find the mount information:

  1. Search through the documentation (for example, the TCP/IP development toolkit and the installation and administration guides).
  2. If you cannot find the mount information in the documentation, contact the vendor that offers the TCP/IP development toolkit.
- 32-bit mode and 64-bit mode

  IBM supports the compilation of the client enabling commands in 32-bit mode on all supported platforms and in 64-bit mode on AIX, Sun, Linux, and z/Linux.

  IBM has tested the client enabling commands in 32-bit mode and 64-bit mode on both 32-bit and 64-bit capable systems using the standard gcc (GNU Compiler collection) 4.0.x compiler command for Linux, gcc 3.4.x compiler command for Sun, and gcc 4.0.x compiler command for AIX. Compiler support has been expanded to include Sun Studio 11 for Solaris 10 and XLC v8 for AIX 5.3.x.x.

  The IBM-supplied makefile for the NFS client enabling commands provides the following keywords to enable the use of 64-bit addressing on various platforms:

AIX64	for the AIX platform
LINUX64	for the Linux platform
SUN64	for the SUN platform
ZLINUX64	for the z/Linux platform.

  The IBM-supplied makefile for the NFS client enabling commands also provides the following keywords to override some of the default values:

  CC
  Allows the default compiler used by the makefile for the target platform to be overridden with a different compiler name. For example, “make cc=gcc sun” builds the utilities for the SUN Solaris platform using the gcc compiler.

CFLAGS Allows the default compiler options to be used for the target platform to be overridden with a different set of compiler options. The options string must be placed in double quotes, because the string can include any characters, including blanks.

Notes:

1. 64-bit mode is not currently supported for the z/OS client.
2. IBM cannot test all possible compiler/option combinations. Any compilation or execution failures experienced when the default compiler/option values are overridden are the customer's responsibility to resolve.
3. The required system run-time libraries for 64-bit support must be available on the platform.

Linking

After linking the programs together, check for attention messages and error messages. The following items might vary for your client machine's operating system:

- Different set of linker/loader flags

  Some programs require a different set of linker/loader flags.
- Library files in a different directory

  The library files required to complete the linkage could be in a different directory.
- Library files have different names

  Depending on how the client machines' operating systems are installed, the library files might have different names.
- Different libraries required

  The system variables could be in different libraries for the different operating systems.
- Compiler is not compatible with the system toolkit

  Some operating systems support multiple versions of C compilers. Some C compilers, however, might not match the various system toolkits.
- Different library model required

  The library models have to match with the compilation time.
- System variables not supported

  Some system variables are supported during compile time but not supported by the link time.

Running commands

After the compilation and linkage are successful, run the command to see if the result is as expected. If not, figure out the difference of the result or failure. The difference or failure can be in the following areas:

- Definition of the standard C variables is different

  The definition of the standard C variables could be different for the different operating systems or compilers. Some special handling might be required. For example:
  - **int** is 2 bytes long for DOS but it is 4 bytes long for the z/OS NFS, AIX, and Sun.
  - **tab** has a different value, causing the spacing of the output to be different.
- Definition of function calls is slightly different

Although a given function is supported, it might work slightly differently. For example, the “mount table” has a different format in AIX from Sun Solaris.

- Library functions might have a defect

  Some functions in the C libraries do not function as the documentation describes. You might report the problems or write your own functions to replace them.

NFS v4 protocol name mapping

Using NFS v4 protocol (NFSv4) name mapping, a user can map owner and group names on a single DNS domain (INET environment) or on multiple DNS domains (CINET environment) to z/OS USS uid and gid numeric values. **nfsv4domain**(*NFSv4_default_domain*) specifies the "pseudo" NFSv4 domain for the NFSv4 name mapping. The "pseudo" NFSv4 domain allows various NFSv4 Clients from various network domains to seamlessly access the server provided that these NFSv4 Clients are also configured with the same "domain,"

Advantages: NFSv2 and NFSv3 protocol has been limited to the use of the UNIX-centric user identification mechanism of numeric user id (uid and gid). However, for NFS to move beyond the limits of large work groups, the NFS v4 protocol changed the user identification to be string based. It provides:

- The owner and group names to be administered on a DNS domain basis
- Flexibility (support of multiple internal naming schemes).

NFSv4 Name Mapping requires:

- The same owner and group names to be defined on both the server and client.
- The owner and group names must be defined to RACF with appropriate uid and gid values on z/OS.
- **nfsv4domain**(*NFSv4_default_domain*) attribute should be appropriately set. When **nfsv4domain** is omitted, the participating NFSv4 Client's domain must match one of the Server's network domain for the proper NFSv4 name mapping.

Name resolution is not supported through any global name server such as LDAP.

The NFS client and NFS server can have their own local representation of owner and owner_group attributes that is used for local storage or presentation to the end user. Owner and owner_group attributes are transferred between the NFSv4 client and server in the form of "user@dns_domain". To provide a greater degree of compatibility with NFSv2 and NFSv3, which identified users and groups by 32-bit unsigned uids and gids, the owner and group strings that consist of decimal numeric values can be given a special interpretation by clients and servers.

Examples of owner, owner_group attributes syntax:

- vndrcvr@storage.tucson.ibm.com
- sys@storage.tucson.ibm.com
- 100 (numeric string "100").

z/OS NFS server has supported NFSv4 since V1R7. z/OS NFS server V1R7, V1R8, V1R9 only support the limited interpretation (i.e. only numeric strings). z/OS NFS server V1R10 added full name@domain strings. The inability of z/OS NFS server prior to V1R10 to interpret the owner and owner_group attributes caused problems with NFSv4 root support (See z/OS NFS APAR OA22311).

- z/OS NFS server has supported <root> suffix in Export List since z/OS NFS server V1R8
- Root support widely uses the functions (chown, chgrp) demanding full name@domain strings.

The z/OS NFS server performs the following:

- Mapping inbound owner/owner_group attributes to local representation

  Usage of RPC uid and gid as local representation in case of unsuccessful interpretation.
- Mapping local representation to outbound owner/owner_group attributes

  Usage of z/OS NFS server uid and gid cache in case of the absence of the local representation in the RACF database.
- Mapping owner and group names to a single DNS domain or multiple DNS domains

Inbound owner/owner_group processing

Inbound owner/owner_group attributes may be present in:

- CREATE, NVERIFY, OPEN, SETATTR, VERIFY NFSv4 operations
- Access Control List NFSv4 (ACL) attribute.

Owner/owner_group attributes can be in the form of:

- "user@dns_domain" strings
- Special strings, like anonymous user/group strings or superuser strings
- Numeric string (for example, "100").

There are special strings representation:

- Client anonymous user
- Client superuser.

owner attribute for anonymous user consists of: String "nobody" with the at sign "@" and the domain

owner_group attribute for anonymous user consists of: String "nobody" with the at sign "@" and the domain

owner attribute for client superuser consists of: String "root" with the at sign "@" and the domain

owner_group attribute for client superuser can be anyone.

The z/OS NFS server uses the implicit interpretation on the base of hardcoded strings "nobody" and "root" (without the reference to RACF database) to designate a client anonymous user and client superuser, except for owner_group attribute for client superuser.

The z/OS NFS server stores anonymous uid and gid:

- At z/OS NFS server start up, the RACF database is queried for user with "nobody" name
- If that user is found in the RACF database, its RACF uid and gid are used
- Otherwise 65534 (not ‘-2’) is used.

z/OS NFS server maps:

- Anonymous owner string to RACF uid of "nobody"
- Anonymous owner_group string to RACF gid of "nobody"
- Superuser owner string to 0
- Superuser owner_group string in the standard way (using z/OS NFS server internal cache and RACF database). z/OS NFS server does not use default name for superuser group name.

The subsequent processing of UID=0:

- If the z/OS NFS server is in EXPORTS mode and EXPORT list entry includes <root> suffix, UID keeps the value
- Otherwise, UID maps to the stored RACF UID of "nobody"

Outbound owner and owner_group attributes processing

Outbound owner and owner_group attributes may be present in:

- GETATTR, READDIR NFSv4 operations

Owner and owner_group attributes can be sent to the client in the form of:

- "user@dns_domain" strings
- Special strings like anonymous user/group strings and superuser strings
- Numeric string (for example, "100")

Translation of UID, GID to owner and owner_group attributes uses (in the following order):

1. Internal z/OS NFS server cache of UIDs, GIDs
2. RACF database

For invalid translation z/OS NFS server uses numeric string representation for outbound owner and owner_group attributes.

Special strings "nobody" and "root" generation :

- If UID is equal to the stored uid of "nobody", z/OS NFS server maps it to anonymous owner string using the default "nobody" name.
- If GID is equal to the stored gid of "nobody", z/OS NFS server maps it to anonymous owner_group string using the same default "nobody" name.
- If UID is equal to 0, z/OS NFS server maps it to superuser owner string using default "root" name.
- If GID is equal to 0, z/OS NFS server maps it from RACF cache and derives the group name from the RACF database.

z/OS NFS server uses internal z/OS NFS server uid and gid cache and RACF database to find user/group names and their UIDs/GIDs.

During mapping inbound owner and owner_group attributes to numeric identifiers:

- The z/OS NFS server gets UID by user name from the RACF database.
- The z/OS NFS server gets GID by group name from the RACF database.

During mapping numeric identifiers to outbound owner and owner_group attributes:

- The z/OS NFS server searches for the numeric identifier (UID or GID) in internal z/OS NFS server cache.
- If the numeric identifier is found in z/OS NFS server cache then there is no reason to obtain it from the RACF database (invalid translation uses numeric string representation).
- Otherwise the z/OS NFS server attempts to obtain a name by numeric identifier from the RACF database.
- If the name is not found in the RACF database, the z/OS NFS server puts numeric identifier into internal z/OS NFS server cache and uses the numeric string representation.

Domain processing in NFSv4 Name Mapping

The domain portion of the owner or owner_group attribute refers to a DNS domain name. For example,

```
user@us.ibm.com
```

For inbound mapping, the z/OS NFS server validates the domain portion of the owner or owner_group attribute .

- • The z/OS NFS server identifies the domain to be used to verify the domain portion of owner/group strings coming from the client. It does so as follows:

  If the **nfsv4domain** attribute was set, the z/OS NFS server uses it as its domain for NFSv4 name mapping; otherwise, the z/OS NFS server gets the domain from the TCP/IP stack which connects with the client; if the z/OS NFS server cannot get the domain from TCP/IP stack, it uses the default server domain. The z/OS NFS server compares this domain with the domain portion of the owner/group string that the server has received from the client.
- • If the domain portion of owner/group string is not valid, z/OS NFS server attempts to derive UID and GID from the RPC packet.

For outbound mapping z/OS NFS server puts the domain portion to the owner or owner_group attribute

- For an invalid translation, z/OS NFS server uses numeric string representation for outbound owner and owner_group attributes (without domain).

Users can override the DNS domain(s) with the z/OS NFS server or z/OS NFS client attribute **nfsv4domain**(*NFSv4_default_domain*).

Chapter 11. Network File System operation

This topic describes how to start and stop the z/OS Network File System, and describes the operands of the MVS **modify** and **display** commands that are related to the z/OS NFS server. The operands for collecting diagnostic information are also described.

Starting the z/OS NFS client

If you want to use the z/OS NFS client, do the following:

- Define the z/OS client as a file system in the z/OS UNIX BPXPRMxx member of SYS1.PARMLIB. Start the z/OS UNIX address space, which will cause the BPXPRMxx member to be used. As part of the z/OS UNIX startup, the z/OS NFS client will be started in an z/OS UNIX colony address space. Wait until this message appears before proceeding:

  ```
  BPXI004I OMVS INITIALIZATION COMPLETE
  ```

  See *z/OS UNIX System Services Planning* for more information.
- Ensure that TCP/IP and PORTMAP are active.

During z/OS UNIX file system initialization, the z/OS NFS client is started and run in the z/OS UNIX colony address space. The FILESYSTYPE parmlib statement for the z/OS NFS client must be present in the SYS1.PARMLIB(BPXPRMxx) parmlib member in order to start the z/OS NFS client. BPXPRMxx can specify optional component trace options for the NFS client, as shown in "Using NFS client component trace PARMLIB member CTINFC*nn*" on page 300. For more information on z/OS UNIX file system, see *z/OS UNIX System Services File System Interface Reference*.

If the z/OS NFS client fails to initialize, a write to operator (WTO) message is issued to the operator console. The following conditions can cause z/OS NFS client initialization to fail.

- The z/OS NFS client is not started in a standalone colony address space.
- The z/OS NFS client is already started. Multiple instances of the z/OS NFS client on a single z/OS system is not supported.
- Using a security product that is downlevel, the z/OS NFS client requires RACF 2.2 or later.
- An incorrect parameter is specified in the installation parameters.
- Unicode services is not installed or is not active.

When the z/OS NFS client initializes, messages like these example messages are displayed on the operator's console.

```
$HASP373 NFSCR    STARTED
BPXI004I OMVS INITIALIZATION COMPLETE
GFSC700I z/OS NETWORK FILE SYSTEM CLIENT
(HDZ11VC) STARTED
```

If the z/OS NFS client is stopped, canceled, or for any other reason terminates, z/OS UNIX issues the following message:

```
BPXF032D FILESYSTYPE type TERMINATED. REPLY 'R' WHEN READY TO RESTART.
         REPLY 'I' TO IGNORE
```

© Copyright IBM Corp. 1991, 2010

To restart the z/OS NFS client, specify 'R' in reply to the message. Replying with 'I' will cause z/OS UNIX to ignore the termination of the z/OS NFS client. If 'I' was replied and you still wish to restart the z/OS NFS client, use the SET command as follows. If z/OS UNIX has been initialized but the z/OS NFS client is not active, issue a SETOMVS RESET=(xx) command to the BPXPRMxx member of SYS1.PARMLIB that defines the z/OS NFS file system. z/OS UNIX will then start the z/OS NFS client.

Stopping the z/OS NFS client

The z/OS NFS client is started when the z/OS UNIX file system is initialized and is persistent until z/OS UNIX is stopped. To stop the z/OS NFS client gracefully the system operator can use the modify operator command **omvs,stoppfs** specifying the NFS client, as follows:

```
f omvs,stoppfs=NFS
```

If this command fails to gracefully shut down the z/OS NFS client, the operator can force an abnormal termination using the operator command **cancel** with the z/OS NFS client address space name; for example:

```
cancel  mvsnfsc
```

It is imperative and necessary to stop the z/OS NFS client gracefully so it can save its important data (RPC transaction ID) for the subsequent restart.

Attention: The destruction of the z/OS NFS client address space can cause unpredictable abnormal z/OS UNIX address space behavior. If a z/OS UNIX process tries to access the NFS client data during its address space destruction, then an OC4 protection exception in the z/OS UNIX BPXVCLNY load module can occur.

To bring down the NFS client during shutdown, follow these steps:

Step 1. Stop the NFS client. It runs in a colony address space; to terminate it, enter either of the following:

- `f omvs,stoppfs=nfs`
- `CANCEL <nfsv>` (if STOPPFS did not work)

Step 2. Stop TCPIP, because it is a registered blocker of OMVS SHUTDOWN.

Step 3. Terminate z/OS UNIX System Services

```
f omvs,shutdown
```

Starting component tracing for the z/OS NFS client

To start recording diagnostic information for the z/OS NFS client in z/OS component trace buffers, follow these steps:

1. Decide on the trace options to use. These can be in a CTINFC*nn* member of SYS1.PARMLIB to be specified on the TRACE CT command, or individual options to be specified when prompted in response to the TRACE CT command.
2. From the master console or another console with master authority, issue the TRACE CT command as follows:

```
►►──TRACE CT,ON,COMP=──mvsnfsc──┬─────────────────┬──────────────────►◄
                                └─,parm=CTINFCnn─┘
```

where

mvsnfsc
: Specifies the name of the procedure in your system that PROCLIB used to start up the client.

parm=CTINFCnn
: Specifies that diagnostic information for the NFS client be recorded in z/OS component trace buffers using trace options specified in member CTINFC*nn* of SYS1.PARMLIB. To use the default trace options for the z/OS NFS client, specify *parm=CTINFC00* for the default SYS1.PARMLIB member CTINFC00.

If you do not specify a CTINFC*nn* PARMLIB member on the CTRACE CT command, the following message will prompt you to enter trace operands:

```
* id ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
```

In response to the message, reply with the identification number *id* from the message, an external writer program if any to receive the records, and a trace option or options to use from the list below. Here is the response syntax:

where:

```
►►──R(eply)──id──OPTIONS=──(name,name...)──┬──────────────────────────┬──────►
                                           └─,WTR=──┬─procname───┬─┘
                                                    └─DISCONNECT─┘

►──┬─,CONT─┬───────────────────────────────────────────────────────────►◄
   └─,END──┘
```

id Specifies the identification number from the ITT006A message.

name,name...
: Specifies a trace option or options for tracing NFS client records. The possible options are:

Buffer
: Buffer management (BUFNODE)

CB_Mgmt
: Control Block Management (creation, initialization, modification, deletion).

Detail
: Detailed Trace Record. This is used for low level debug.

Dispatch
: Unit of work is dispatched from a queue to resume processing.

Entry
: Entry into a function

Exit
Exit from a function.

FFDC
First Failure Data Capture. This option is on by default and cannot be turned off.

General
General Trace Record.

Lock_Release
Control block lock release

Lock_Request
Control block lock requests

Lock_Resume
Control block lock request resumes after lock request either succeeds or fails.

Msg
Existing NFS Trace Error, Warning and Informational Records. This option is on by default and cannot be turned off.

Network
Network communication related trace.

NFS_Request
Request sent to NFS server.

NFS_Return
Return from NFS request.

Resume
Unit of work resumed due to availability of resource (for example, latch acquired).

Schedule
Unit of work is scheduled onto another queue (for example, ipcqueue of this or another task, array queue.).

Suspend
Unit of work must suspend due to unavailability of resource (for example, waiting for a latch).

Trap
For use in special temporary trap code created to aid in the analysis of a problem.

USS_Request
Request issued to z/OS UNIX System Services.

USS_Return
Returned from z/OS UNIX System Services request.

In addition to these basic options, you can also enter the following shorthand values to specify multiple record types:

All
All record types.

Call
Entry and Exit record types.

Lock
Lock_Request, Lock_Result, and Lock_Release record types.

NFS
NFS_Request and NFS_Return record types.

Task_Flow
Suspend, Resume, Schedule, and Dispatch record types.

USS
USS_Request and USS_Return record types.

Notes:

1. An option can be turned off by preceding the option value with a minus sign (for example, OPTIONS=(-GENERAL)).
2. Options are processed from left to right, first processing all values to turn on options and then processing all values to turn off options. Thus all options except Network can be turned on with the following specification: OPTIONS=(ALL, -NETWORK).
3. If an option value of -ALL is specified, only the minimum set of options remains active (FFDC and MSG).

WTR=procname | DISCONNECT
Connects or disconnects the component trace external writer and the trace. *procname* identifies the name of the member that contains the source JCL that invokes the external writer. The member can be a SYS1.PROCLIB cataloged procedure or a job. The procname in the WTR parameter must match the procname in a previous TRACE CT,WTRSTART command.

WTR=DISCONNECT disconnects the writer and the trace. The component continues tracing and placing the trace records in the address-space buffer, but stops passing trace records to the external writer.

You must also specify a TRACE CT,WTRSTART or TRACE CT,WTRSTOP command to start or stop the writer.

CONT or END
Specifies that the reply continues on another line. The system reissues the same prompting message. You then can continue the reply. You can repeat any parameters on the continuation line, except END. Repeated parameters are strung together. They do not overlay each other. You must specify END to complete the response. END identifies the end of the REPLY.

Starting the z/OS NFS server

Make sure that z/OS UNIX is customized to be able to start automatically during IPL.

If you want to use the z/OS NFS server, TCP/IP and PORTMAP need to be started and active on your system. Then, start the z/OS NFS server.

You might also need to start up NAMESRV to map machine names into their corresponding Internet addresses. If you use the dynamic host configuration protocol (DHCP), NAMESRV is required.

Notes:

1. PORTMAP is synonymous with portmapper, which is a program provided by z/OS Communications Server that maps registered server programs to port numbers. NAMESRV is the cataloged procedure of the Domain Name Server

which is provided by Communications Server that maps a host name to an internet address or an internet address to a host name. See *z/OS Communications Server: IP Configuration Guide* and *z/OS Communications Server: IP Configuration Reference*for information on configuring PORTMAP, starting PORTMAP, configuring the Domain Name Server, and starting NAMESRV automatically with z/OS Communications Server.

2. The z/OS Portmapper does not support IPv6. Therefore, when using IPv6 addresses, the z/OS server host must be configured with RPCBIND, not the Portmapper. RPCBIND supports both IPv6 and IPv4. As of z/OS V1R8, Portmapper should only be used for IPv4 only systems. Otherwise, RPCBIND should be used.

The z/OS NFS server does not support file persistence. That is, when the server is restarted, files cannot be accessed using old file handles.

To start the z/OS NFS server, enter the **start** command from a console. Enter the command as follows:

```
►►──START──mvsnfs──┬───────────────────────────────────────────────────────────────────────────┬──►
                   │          ┌─info──┐                                                      │
                   └─,parms='─┼─error─┼─┬─────────┬─┬────────────┬─┬──────────┬─┬──────────┬─'─┘
                              └─warn──┘ └─,altsym─┘ └─,ctrace=nn─┘ └─,dsps=nn─┘ └─,version─┘

►──┬────────────────────────┬─────────────────────────────────────────────────────────────────►◄
   │            ┌─ignore─┐  │
   └─startfail=─┴─dump───┴──┘
```

where

mvsnfs Specifies the name of the procedure in your system that PROCLIB used to start up the server.

info The first parameter specifies the level for diagnostic messages (the default is *info*).

altsym If *altsym* is specified in the second parameter, the semicolon (;) is used as the comment symbol in the attributes (with one exception: If the fn_delimiter(;) attribute uses a semicolon, the fn_delimiter semicolon is treated as a delimiter between the file name and the attributes that follow, not as a comment symbol) and exports data sets. Otherwise, the pound sign (#) is used as the comment symbol.

For example, suppose you have some data sets with a high-level qualifier of *#USER05*, and you want client users to be able to read them. First, you would modify the exports data set and attributes data set by using ';' as the comment symbol rather than '#'. Next you would specify `#user05 -ro` as an entry in the exports data set. Finally, you would specify the **altsym** parameter when you enter the **start** command.

ctrace=nn

Specifies that diagnostic information for the NFS server be recorded in z/OS component trace buffers, using trace options specified in member CTINFS*nn* of SYS1.PARMLIB. To use the default trace options for the z/OS NFS server, specify **ctrace=00** for the default SYS1.PARMLIB member CTINFS00. If the **ctrace** operand is not specified, then the default SYS1.PARMLIB member CTINFS00 is used.

dsps=nn

Specifies the size of the data space to be allocated for the NFS component trace buffers, where *nn* equals the number of megabytes to be allocated for each trace buffer. NFS uses 3 trace buffers in rotation. The value of *nn* can be between 10 and 600. If a value outside this range is specified, it will be adjusted accordingly. The default value is 10 (representing 10 MB).

version

Causes the server to place a GFSA947I message on behalf of each module in the server into the server log and component trace, at startup. Similar to using the **MODIFY** *mvsnfs***, version=all** operator command.

startfail

Specifies the action the z/OS Network File System Server should take if it encounters a terminating error during startup, or later during execution which did not otherwise create a dump.

- **ignore**: No special action is taken. The z/OS Network File System server proceeds with termination as it has done prior to z/OS NFS V1R11. This is the default setting if the parameter is not specified.
- **dump**: For terminating errors, an SVC dump is created before terminating, if no dump was otherwise created.

Note: No dump is produced if the z/OS Network File System server is stopped by the operator stop request. A dump is generated if the z/OS Network File System server shuts down because of a TCP/IP termination.

These parameters override the parameter settings in the server startup procedure.

When you enter **start**, the following console message appears; if you installed HDZ1B1N, then this FMID is displayed in the GFSA348I message text.

```
GFSA348I z/OS NETWORK FILE SYSTEM SERVER
(HDZ1B1N) STARTED.
```

Starting multiple servers

You can start more than one z/OS NFS server on a single z/OS system, but they must be at the same release level. If a server is running, attempts to start another release level of the server on the same system will fail.

Note: The NFS server and NFS client can be at different release levels on the same system; there is no requirement for server and client release levels to match.

Within a sysplex, you can have different levels of NFS servers and clients on different systems. However, byte-range locking and file share reservations are not communicated across systems within the sysplex. Therefore, in sysplex environments where byte range locking and share reservations are required for accessing MVS data sets with NFS, IBM recommends that only one NFS server be started.

Stopping the z/OS NFS server

When shutting down your system, follow these sequential steps.

Step 1. Stop the z/OS NFS server

Step 2. Shut down the TCP/IP server

Step 3. Shut down z/OS UNIX if it is running

Use the **stop** command to shut down the z/OS NFS server. All current input/output (I/O) operations are completed and all OPEN data sets are closed. This command can be entered at any time. Enter the command as follows, where *mvsnfs* is the name of the procedure in your system that PROCLIB used to start up the server.

```
►►──STOP──mvsnfs──────────────────────────────────────────────────────►◄
```

You can also shut down the server by entering the **modify** command.

```
►►──┬─MODIFY─┬──mvsnfs,stop───────────────────────────────────────────►◄
    └─F──────┘
```

The operator's console displays messages like the following messages.

```
GFSA329I SERVER SHUTDOWN IN PROGRESS.
GFSA330I SERVER SHUTDOWN COMPLETE.
$HASP395 MVSNFS   ENDED
```

As a last resort, you can cancel the server by entering the **cancel** command.

```
►►──CANCEL──mvsnfs────────────────────────────────────────────────────►◄
```

Starting the z/OS NFS NSM and z/OS NFS NLM

If you want to use the z/OS NFS Network Status Monitor (z/OS NFS NSM) and the z/OS NFS Network Lock Manager (z/OS NFS NLM), you must specify the NLM site attribute for the z/OS NFS server. See Table 24 on page 134 for details. If you specify the NLM site attribute, NLM and NSM will start when the NFS server is started, and will stop when the NFS server is stopped. The attribute's default value, NONLM, specifies that the NFS server will run without NLM or NSM.

After a z/OS NFS server restart, the z/OS NFS server sends an SM_NOTIFY request to each NFS client that supports NLM. The NFS client should reclaim locks as soon as it receives the SM_NOTIFY request.

Starting component tracing for the z/OS NFS server

To start recording diagnostic information for the z/OS NFS server in z/OS component trace buffers, follow these steps:

1. Decide on the trace options to use. These can be in a CTINFS*nn* member of SYS1.PARMLIB to be specified on the TRACE CT command, or individual options to be specified when prompted in response to the TRACE CT command.
2. From the master console or another console with master authority, issue the TRACE CT command as follows:

```
►►──TRACE CT,ON,COMP=──mvsnfs──┬──────────────────┬─────────────────────►◄
                               └─,parm=CTINFSnn──┘
```

where

mvsnfs Specifies the name of the procedure in your system that PROCLIB used to start up the server.

parm=CTINFSnn
: Specifies that diagnostic information for the NFS server be recorded in z/OS component trace buffers using trace options specified in member CTINFS*nn* of SYS1.PARMLIB. To use the default trace options for the z/OS NFS server, specify *parm=CTINFS00* for the default SYS1.PARMLIB member CTINFS00.

If you do not specify a CTINFS*nn* PARMLIB member on the CTRACE CT command, the following message will prompt you to enter trace operands:

```
* id ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND.
```

In response to the message, reply with the identification number *id* from the message, an external writer program if any to receive the records, and a trace option or options to use from the list below. Here is the response syntax:

where:

```
►►──R(eply)──id──OPTIONS=──(name,name...)──┬────────────────────────────┬──────►
                                           └─,WTR=──┬─procname───┬─┘
                                                    └─DISCONNECT─┘

►──┬─,CONT─┬────────────────────────────────────────────────────────────►◄
   └─,END──┘
```

id Specifies the identification number from the ITT006A message.

name,name...
: Specifies a trace option or options for tracing NFS server records. The possible options are:

Buffer
: Buffer Management (that is, BUFNODE)

CB_Mgmt
: Control Block Management (creation, initialization, modification, deletion).

Detail
: Detailed Trace Record. This is used for low level debug.

DFP_request
: Call to MVS data set system function.

DFP_return
: Return from MVS data set system function.

Dispatch
: NFS Client worker thread posts results back to Cross memory thread.

Entry
Entry into a function

Error
Existing NFS Trace Error Record. This option is on by default and cannot be turned off.

Exit
Exit from a function.

FFDC
First Failure Data Capture. This option is on by default and cannot be turned off.

General
General Trace Record.

Info
Existing NFS Trace Info Record.

Lock_Release
Control Block Lock Release

Lock_Request
Control Block Lock Requests

Lock_Resume
Control block lock request resumes after lock request either succeeds or fails.

Network
Network communication related trace.

Resume
Cross memory thread receives result from NFS Client worker thread and resumes processing, or NFS worker thread receives work for processing

Schedule
Cross memory thread sends work to NFS Client worker thread and continues processing in parallel.

Suspend
Cross memory thread sends work to NFS Client worker thread and waits for result, or NFS worker thread waits for work.

Trap
For use in special temporary trap code created to aid in the analysis of a problem.

USS_request
Request issued to z/OS UNIX System Services.

USS_return
Returned from z/OS UNIX System Services request.

Warning
Existing NFS Trace Warning Record.

In addition to these basic options, you can also enter the following shorthand values to specify multiple record types:

All
All existing NFS trace records.

Call
Entry and Exit record types.

Debug1
FFDC, Entry and Exit record types.

Debug2
FFDC, Entry, Exit and Network record types.

Debug4
FFDC, Entry, Exit, Network, Suspend, Resume, Schedule, Dispatch and General record types.

Debug9
All existing NFS trace records (equal to **All**).

Lock
Lock_Req, Lock_Result and Lock_Release record types

MVS
DFP_Request and DFP_Return record types.

NFS
NFS_Req and NFS_Rtn record types.

Task_Flow
Suspend, Resume, Schedule and Dispatch record types.

USS
USS_Request and USS_Return record types.

Notes:

1. An option can be turned off by preceding the option value with a minus sign (for example, OPTIONS=(-GENERAL)).
2. Options are processed from left to right, first processing all values to turn on options and then processing all values to turn off options. Thus all options except Network can be turned on with the following options specification: OPTIONS=(ALL, -NETWORK). If an options value of " -ALL" is specified, the options revert back to a minimum trace state .
3. A minimum trace state has been defined, which will be the default initial trace state and will also be the trace state that takes effect if "all" trace states are turned off. In this "MIN" state, ERROR and FFDC record tracing will remain active. They cannot be deactivated. Neither MIN nor -MIN can be specified in the OPTIONS list.

WTR=procname|DISCONNECT
Connects or disconnects the component trace external writer and the trace. *procname* identifies the name of the member that contains the source JCL that invokes the external writer. The member can be a SYS1.PROCLIB cataloged procedure or a job. The procname in the WTR parameter must match the procname in a previous TRACE CT,WTRSTART command.

WTR=DISCONNECT disconnects the writer and the trace. The component continues tracing and placing the trace records in the address-space buffer, but stops passing trace records to the external writer.

You must also specify a TRACE CT,WTRSTART or TRACE CT,WTRSTOP command to start or stop the writer.

CONT or END
Specifies that the reply continues on another line. The system reissues the same prompting message. You then can continue the reply. You can repeat

any parameters on the continuation line, except END. Repeated parameters are strung together. They do not overlay each other. You must specify END to complete the response. END identifies the end of the REPLY.

Note: It is no longer necessary to use the **flushctr** option on the Modify command to flush the last buffer being filled to the component trace before the external writer is stopped or before a dump is taken. This function is now done automatically when the Component Trace external writer is stopped.

Entering operands of the modify command for the z/OS NFS server

Besides the **start** and **stop** commands, you can enter the **modify** command at the console with thee operands shown in Figure 29 on page 223. All operands must be preceded by either **MODIFY** *mvsnfs*, or **F** *mvsnfs*, where *mvsnfs* is the name of the procedure in your system that PROCLIB used to start the server.

Each message is sent to a component trace buffer, or to the console or the data set that is pointed to by the NFSLOG1 and NFSLOG2 DD statements, or to any combination of those destinations. NFSLOG1 and NFSLOG2 are DD statements in the startup procedure of the z/OS NFS server. These z/OS NFS server DD statements specify data sets where all the messages for debugging or trace are stored during the processing of the z/OS NFS server.

All data set names entered from the console must be fully-qualified and without quotation marks.

This is a summary of the modify command with the relevant operands:

```
►►──┬─MODIFY─┬──mvsnfs,addds=ddname(dsname)──────────────────────────►◄
    └─F──────┘
```

```
|  ►►──┬─MODIFY─┬──nfsserver,bufferusage───────────────────────────────►◄
       └─F──────┘
|
```

```
|  ►►──┬─MODIFY─┬──nfsserver,bufhigh──┬───────────────────┬────────────►◄
       └─F──────┘                     └─=(xx──┬─────┬──)──┘
                                              └─,yy─┘
|
```

```
|  ►►──┬─MODIFY─┬──mvsnfs,cbsniff──┬────────┬──────────────────────────►◄
       └─F──────┘                  └─=nnn───┘
```

```
►►──┬─MODIFY─┬──mvsnfs,debug=──┬─deleg────┬──────────────────────────►◄
    └─F──────┘                 └─delegoff─┘
```

```
►►──┬─MODIFY─┬──mvsnfs,DlyDTimeout=n─────────────────────────────────►◄
    └─F──────┘
```

```
►►──┬─MODIFY─┬──mvsnfs,exportfs──────────────────────────────────────►◄
    └─F──────┘
```

```
►►──┬─MODIFY─┬──mvsnfs,freeds=ddname─────────────────────────────────►◄
    └─F──────┘
```

```
►►──┬─MODIFY─┬──mvsnfs,freeze=──┬─on──────┬──────────────────────────►◄
    └─F──────┘                  ├─off─────┤
                                ├─onhfs───┤
                                └─offhfs──┘
```

```
►►──┬─MODIFY─┬──mvsnfs,list=──┬─mounts──┬────────────────────────────►◄
    └─F──────┘                └─dsnames─┘
```

```
►►──┬─MODIFY─┬──mvsnfs,mapfile──┬──────────┬─────────────────────────►◄
    └─F──────┘                  └─(dsname)─┘
```

```
►►──┬─MODIFY─┬──mvsnfs,racf_cache_reset──────────────────────────────►◄
    └─F──────┘
```

Figure 29. Summary of the modify command (Part 1 of 3)

```
▶▶──┬─MODIFY─┬──mvsnfs,release=data_set_name──┬──────────┬──────────────────▶◀
    └─F──────┘                                └─(member)─┘

                                     ┌─off──┐
▶▶──┬─MODIFY─┬──mvsnsf,──sockhang=──┼─on───┼─────────────────────────────────▶◀
    └─F──────┘                       └─trap─┘

▶▶──┬─MODIFY─┬──mvsnfs,status────────────────────────────────────────────────▶◀
    └─F──────┘

▶▶──┬─MODIFY─┬──mvsnfs,swapmhdb──────────────────────────────────────────────▶◀
    └─F──────┘

▶▶──┬─MODIFY─┬──mvsnfs,swapldb───────────────────────────────────────────────▶◀
    └─F──────┘

▶▶──┬─MODIFY─┬──mvsnfs,unmount=mount_point_name──────────────────────────────▶◀
    └─F──────┘

▶▶──┬─MODIFY─┬──mvsnfs,unmntall──────────────────────────────────────────────▶◀
    └─F──────┘

▶▶──┬─MODIFY─┬──mvsnfs,unmnthfs──────────────────────────────────────────────▶◀
    └─F──────┘

▶▶──┬─MODIFY─┬──mvsnfs,unmntmvs──────────────────────────────────────────────▶◀
    └─F──────┘

▶▶──┬─MODIFY─┬──mvsnfs,flushlog──────────────────────────────────────────────▶◀
    └─F──────┘

▶▶──┬─MODIFY─┬──mvsnfs,listlock=file_name────────────────────────────────────▶◀
    └─F──────┘

▶▶──┬─MODIFY─┬──mvsnfs,log=──┬─error─┬───────────────────────────────────────▶◀
    └─F──────┘               ├─info──┤
                             └─warn──┘

| ▶▶──┬─MODIFY─┬──mvsnfs,smf=──┬─off─┬─,─┬─none─────────┬─────────────────────▶◀
      └─F──────┘               └─on──┘   ├─all──────────┤
                                         └─subtype_list─┘
```

Figure 29. Summary of the modify command (Part 2 of 3)

```
►►──┬─MODIFY─┬──mvsnfs,switchlog──────────────────────────────────►◄
    └─F──────┘
```

```
►►──┬─MODIFY─┬──mvsnfs,v4delg=──┬─off─┬───────────────────────────►◄
    └─F──────┘                  └─on──┘
```

```
►►──┬─MODIFY─┬──mvsnfs,version──┬───────────┬──────────────────────►◄
    └─F──────┘                  ├─=all──────┤
                                └─=module───┘
```

Figure 29. Summary of the modify command (Part 3 of 3)

Addds operand

This operand is used to specify a replacement for one of the NFS control data sets. This applies to either the lock data sets or the mount handle data base data sets. When the z/OS NFS server determines that one of the control data sets has become unusable, it will write a warning message to the operator console requesting that the broken data set be replaced. Rename or delete the broken data set, allocate a new data set with the original name, and issue the MODIFY Addds command to enable the new data set. If the ddname does not match one of the valid control data set ddnames that has been freed, an error message is written. If the system programmer chooses not to replace the broken data set, the server will continue processing with the remaining data set. If a different data set name is used, the corresponding data set name in the NFS server startup procedure must be changed. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,addds=ddname(dsname)──────────────────────►◄
    └─F──────┘
```

Where:

ddname
is the ddname of the NFS server control data set that is to be replaced. The valid ddnames are: FHDBASE, FHDBASE2, LDBASE, LDBASE2.

dsname
is the name of the lock data set or mount handle data set to be enabled for use by the z/OS NFS server.

Note: The current NFS server control data set must be freed with the **MODIFY Freeds** command before the **MODIFY Addds** command can be issued against it.

bufferusage operand

This operand reports the current data buffer utilization (in Mbytes). Enter the operand as follows:

```
►►──┬─MODIFY─┬──nfsserver,bufferusage──────────────────────────────────►◄
    └─F──────┘
```

bufhigh operand

This operand specifies the storage limit for data buffers for the NFS client. Enter the operand as follows:

```
►►──┬─MODIFY─┬──nfsserver,bufhigh──┬─────────────────────┬──────────────►◄
    └─F──────┘                     └─=(xx──┬───────┬──)─┘
                                           └─,yy───┘
```

Cbsniff operand

This operand is used to periodically look (sniff) for a z/OS NFS Server control block chain breakage associated with z/OS UNIX file systems. The sniffer inspects the z/OS UNIX mount control blocks, checking for allocation and linked-list consistency, and creates a dump if it detects a problem (the z/OS NFS Server continues to run after the dump is taken). Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,cbsniff──┬────────┬──────────────────────────────►◄
    └─F──────┘                  └─=nnn───┘
```

Where:

nnn
is a value of 0 or a number of seconds from 2 to 86400 (24 hours) specifying the time interval at which the sniffer is to run. If the specified value is larger than the maximum in this range, it is set to 86400. If it is 1, it is set to 2 (to prevent the sniffer from running too often). If it is negative, it is set to 0.

If the sniffer value is set to 0, the sniffer is turned off; otherwise it will be run immediately and then set to the timer period (in seconds) as specified.

If Cbsniff is issued without the equals sign ('='), the sniffer runs one time without altering the period of the sniffer (even if it is off at the time). If it does not identify a problem, any timer period is started anew.

If the sniffer identifies a problem, it generates an SDUMP and turns itself off (no further sniffers unless turned on explicitly).

The sniffer default is 0. It must be explicitly started every time the z/OS NFS Server is started or restarted, including for a SYSPLEX failover, or after the sniffer has triggered and created a dump.

DlyDTimeout operand

This command sets the minimum delay detection time value in sec before the delay detection mechanism observes a delay in an external call/function and prints message GFSA1030E on the console. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,DlyDTimeout=n──────────────────────────────────────►◄
    └─F──────┘
```

Valid values are 0 and a range of 5 to 60 seconds. Any value of DlyDtimeout from 1 to 4 seconds is converted to 5 seconds. If DlyDTimeout is set to 0 the delay detection mechanism is turned off. The Default value is 10 seconds.

Exportfs operand

This operand causes the exports data set to be re-read and the internal exports list to be rebuilt without stopping and restarting the server. Checklist data specified on the **dirsuf** parameter in the exports data set is included in the update. This operand can be run at any time. Any changes in the **sec** keyword of the export list will be **immediately** enforced at server startup or EXPORTFS time, regardless of the state of any preexisting mount points. If client IP addresses have been changed and the z/OS NFS server is running in NODHCP mode, the exportfs command will **not** rebuild the exports list correctly. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,exportfs───────────────────────────────────────────►◄
    └─F──────┘
```

Freeds operand

This operand is used to free one of the NFS control data sets so it can be replaced. This applies to either the lock data sets or the mount handle data base data sets. Enter the operand as follows.

```
►►──┬─MODIFY─┬──mvsnfs,freeds=ddname──────────────────────────────────────►◄
    └─F──────┘
```

Where:

ddname

is the ddname of the NFS server control data set that is to be freed. The valid ddnames are: FHDBASE, FHDBASE2, LDBASE, LDBASE2.

Notes:

1. Only the currently inactive data set of the mount handle data base, or lock data base, pair can be freed. Therefore, if the active data set is the one to be freed, it is necessary to first swap the data set pair. This can be done with a swapmhdb, or swapldb, command. This command swaps the active and inactive data sets in the database. Once this is done, it is then possible to free the previously active (now inactive) data set.
2. The freed data set must be renamed or deleted before a new data set of the same name can be allocated.

Freeze operand

This operand suspends or resumes processing of user mount requests. For the NFS V4 protocol, this operand also suspends or resumes processing of lookup operations from the NFS server into z/OS UNIX file systems or MVS data sets. You can enter the *freeze=on* command at any time for maintenance purposes. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,freeze=──┬─on─────┬─────────────────────────────────►◄
    └─F──────┘                  ├─off────┤
                                ├─onhfs──┤
                                └─offhfs─┘
```

- If you select **on**, these messages appear.

  ```
  GFSA901I  Mounts Processing Suspended.
  GFSA771I  z/OS UNIX mounts suspended
  ```

  Future mount requests by client users are refused for both z/OS UNIX file systems and z/OS conventional MVS data sets and the message "Permission Denied" displays on their monitors. After issuing a *freeze=off* operand, mount processing resumes for both z/OS UNIX file systems and z/OS conventional MVS data sets, and these messages appear.

  ```
  GFSA902I  Mounts Processing Resumed.
  GFSA772I  z/OS UNIX mounts resumed.
  ```

  Client users can again mount z/OS directories as normal.
- If you select **onhfs**, future mount requests by client users for z/OS UNIX file systems are refused.
- If you select **offhfs**, mount processing resumes for z/OS UNIX file systems.

List operand

This operand displays a list of either the mount points or the MVS data sets that are currently active in the Network File System. This command can be entered at any time for maintenance or diagnostic purposes. The path name output of an z/OS UNIX file object might require multiple console messages. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,list=──┬─mounts──┬──────────────────────────────────►◄
    └─F──────┘                └─dsnames─┘
```

The **mounts** option returns a list of all mount points currently active in the system. The list of active mount point names and their associated *current use counts* are displayed, along with the list of clients which have active mounts to the mount point. If available, a client's hostname is displayed; if not, the client's IP address is displayed. For example, in the following list entry, the mount point /HFS/U/BLUE has a current use count of 3, indicating that three clients are currently accessing it. The hostnames of the two of the clients are displayed, but the third client's hostname is unavailable, so its IP address is displayed instead.

```
GFSA910I /HFS/U/BLUE ACTIVE =3 hostname1 hostname2 9.1.22.73
```

Note: Because there could be a large number of mounts or hosts for a single mount, NFS limits the console display of this message to a maximum of approximately 10 lines of the response. The entire response is recorded in the NFS log data set. Message GFSA907I is displayed on the console if not all of the information is shown.

The list of clients is always less than or equal to the current use count that is displayed on the ACTIVE parameter; if the number of clients is less than the current use count, then one or more of the clients have multiple active mounts to the mount point.

Current use counts indicate how many mount requests have been made without an unmount request for the same mount point regardless to which local directory the mount is attached. For example, suppose the high-level qualifier `JOHN` is mounted to the same local directory 12 times without any unmount. `ACTIVE =12` is shown. Now, suppose the high-level qualifier `JOAN` has been mounted to 15 different local directories but 5 of them have been unmounted. `ACTIVE =10` is shown, as in the following example:

```
GFSA910I JOHN ACTIVE =12.
...
GFSA910I JOAN ACTIVE =10.
...
```

With the NFS version 4 protocol, the current count reflects only mounts specified with the **mvsmnt** attribute, and the count is not decremented since NFS version 4 has no unmount function.

The **dsnames** option returns a list of all data sets that are currently active. Currently active means that the data set is either in use or has been opened for accessing but has not been closed due to timeout even though the file is not in use. A list of data sets and data set members such as these display.

```
GFSA912I CHRIS.TEST.
GFSA912I SMITH.PAYROLL(JULY).
GFSA912I /HFS/U/PAYROLL.
```

z/OS UNIX file names are truncated after 126 characters.

Mapfile operand

This operand causes the side files data to be read again and rebuilt without stopping and restarting the server or remounting of mount points. If mapfile operand has a parameter (particular side file name), only the specified side file will be updated.

The default is to refresh all side files (default from NFSATTR or previously specified by any mount command). Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,mapfile──┬──────────┬──────────────────────►◄
    └─F──────┘                  └─(dsname)─┘
```

racf_cache_reset operand

This operand erases all UIDs and GIDs from the NFS RACF cache. Note that if you issue this operand during z/OS NFS server shutdown, the command is rejected with GFSA949I (ECMDER) console error message. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,racf_cache_reset──────────────────────────────────►◄
    └─F──────┘
```

Release operand

This operand forces the Network File System to close and deallocate the MVS data set or PDS/PDSE member specified. Any locks which exist will also be released and any waiting locks for the data set or member will be cancelled. Any waiting byte range lock requests (for the specified data set) will be purged. File share reservations will also be released.

For a z/OS UNIX file, any locks held will be released and any waiting locks will be canceled. File names specified with a prefix must be specified in single quotes with the prefix (/hfs or /mvs) followed by the z/OS UNIX path name or the MVS data set name, respectively. Note that the currently active prefix settings apply for determining whether a z/OS UNIX file or an MVS data set is specified. The command syntax is as follows.

Attention: The NFS and NLM protocols do not provide any means for the NFS server to notify the NFS client that the file (or MVS data set) and associated locks have been released. Use of the RELEASE command can lead to data corruption because the NFS client will be unaware of the fact that its file and locks are no longer held. This command should only be used as a last resort when it is not possible to release the file any other way.

NFS v4 Delegation: The Release operand initiates the recall of any delegations associated with the file. This is done asynchronously. The release command completes immediately, not waiting for the recall to complete. If an NFS client has a write delegation, it is the client's responsibility to properly recover from the stale file error received when the client attempts to write back any modified buffers. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,release=data_set_name──┬──────────┬──────────────►◄
    └─F──────┘                                 └─(member)─┘
```

where

data_set_name
Is the name of an MVS data set or an z/OS UNIX file. If *data_set_name* references a PDS or a PDSE data set (does not include the member name), a syntax error will result.

For z/OS UNIX file names, specify the release operand as follows:

```
f mvsnfs,release='/hfs/u/jones' (hfs path specified in quotes)
```

For MVS data sets, specify the release operand as follows:

```
f mvsnfs,release=data_set_name (MVS dataset name not specified in quotes)
```

- If the object you specify is active, you receive a message as follows.

```
GFSA914I data.set.name(member) DEALLOCATED.
```

- If the object you specify is not active, you receive a message as follows.

```
GFSA915I data.set.name(member) NOT ALLOCATED.
```

smf operand

This operand enables changing the smf site attribute settings for the SMF Record Type 42 without requiring a Server Restart. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,smf=─┬─off─┬─,─┬─none─────────┬──────────────────────►◄
    └─F──────┘              └─on──┘   ├─all──────────┤
                                      └─subtype_list─┘
```

Status operand

This operand displays the status of the server's active subtasks. You can enter this command for diagnosis purposes at any time that the server is running. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,status────────────────────────────────────────────►◄
    └─F──────┘
```

This is a sample message listing.

```
   GFSA900I MOUNT PROCESSING ACTIVE.
   GFSA751I SMF PROCESSING SUSPENDED FOR USER LOGOUT.
   GFSA753I SMF PROCESSING SUSPENDED FOR FILE TIMEOUT.
   GFSA781I mvsnfs SAF PROCESSING ENABLED.
   GFSA903I TASK 5C580 TCB 8D1888 PROGRAM = GFSALEGT =
NFSTSK02
   GFSA903I TASK 5C7A0 TCB 8D10C8 PROGRAM = GFSALEGT =
NFSTSK01
   GFSA903I TASK 5C9C0 TCB 8D1378 PROGRAM = GFSALEGT =
NFSTSK00
   GFSA903I TASK 5CE00 TCB 8D1378 PROGRAM = GFSAXPRT =
TRANSPORT
```

Swapldb operand

This operand is used to swap the active and inactive lock data base data sets. Once this is done, it is then possible to free the previously active (now inactive) lock data base data set. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,swapldb───────────────────────────────────────────►◄
    └─F──────┘
```

Swapmhdb operand

This operand is used to swap the active and inactive mount handle data base data sets. Once this is done, it is then possible to free the previously active (now inactive) mount handle data base data set. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,swapmhdb──────────────────────────────────────►◄
    └─F──────┘
```

Unmount operand

This operand unmounts a mount point that is currently active. The path name for a z/OS UNIX file object cannot exceed 126 bytes. File names specified with a prefix must be specified in single quotes with the prefix (/hfs or /mvs) followed by the z/OS UNIX path name or the MVS data set name, respectively. For example: `F mvsnfs,unmount='/HFS/u/jones'`. Note that the currently active prefix settings apply for determining whether a z/OS UNIX file or an MVS data set is specified. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,unmount='mount_point_name'────────────────────►◄
    └─F──────┘
```

- If the mount point is active, the server responds.

  ```
  GFSA916I mount_point UNMOUNTED.
  ```

- If the mount point you specify does not exist when you enter the command, the server displays messages similar to the following messages.

  ```
  GFSA917I SMITH NOT MOUNTED.
  GFSA917I /HFS/U/JONES NOT MOUNTED.
  ```

Once the mount point is removed, client users are unable to access this mount point, and they get the "Stale NFS File Handle" message. Client users have to enter the **umount** command to end the stale file handle problem.

Unmntall operand

This operand causes the server to immediately unmount all mount points without stopping and restarting the NFS server. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,unmntall──────────────────────────────────────►◄
    └─F──────┘
```

Unmnthfs operand

This operand causes the server to immediately unmount all z/OS UNIX mount points without stopping and restarting the NFS server. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,unmnthfs──────────────────────────────────────►◄
    └─F──────┘
```

Unmntmvs operand

This operand causes the server to immediately unmount all MVS data set mount points without stopping and restarting the NFS server. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,unmntmvs──────────────────────────────────────►◄
    └─F──────┘
```

v4delg operand

When the z/OS NFS server starts up, **v4delg** is set to off. Set **v4delg** to on if you want to enable NFS v4 MVS data set access delegation. The **v4delg** operand is used to resume or suspend delegation. This operand has two parameters, ON and OFF. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,v4delg=──┬─off─┬──────────────────────────────►◄
    └─F──────┘                  └─on──┘
```

where

v4delg=on
Enables NFS v4 MVS dataset access delegation.

v4delg=off
Disables NFS v4 MVS dataset access delegation.

Entering operands of the modify command for diagnosis

These operands of the **modify** command can help you to collect data for diagnosing problems.

Note: All operands must be preceded with either:

```
MODIFY mvsnfs,
```

or

```
F mvsnfs,
```

where

mvsnfs Specifies the name of the procedure in your system that PROCLIB used to start up the server.

Each message is sent to the console, the component trace, or the data set that is pointed to by the NFSLOG1 or NFSLOG2 DD statements, or a combination of those destinations.

Debug operand

This operand provides the ability to turn on extra debug controls to capture special diagnostics for specific functional areas in the z/OS NFS Server. Use this operand only when requested by IBM Support. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,debug=──┬─deleg────┬─────────────────────────►◄
    └─F──────┘                 └─delegoff─┘
```

Where:

deleg Monitors the z/OS NFS Server NFS v4 delegation functions for the occurrence of various errors. If any of these errors are detected, a dump is generated to capture the associated NFS trace data before it is lost.

delegoff
Turns off the special z/OS NFS Server NFS v4 delegation function monitoring. The writing of NFS trace records is not changed.

The default setting is for all of the debug controls to be off.

Flushlog operand

flushlog is an operator command that lets you flush the NFS message log to disk. This command enables a TSO/E user to browse all the log records that have been written by the NFS. Enter the operand as follows.

```
►►──┬─MODIFY─┬──mvsnfs,flushlog─────────────────────────────────────►◄
    └─F──────┘
```

Listlock operand

This operand lists client NFS processes that hold locks, share reservations or delegations for a specified file on the z/OS Network File System server. The list of lock holders can be used to diagnose problems with locking, share reservation or delegation conflicts. The listlock operand lets you specify an MVS data set, PDS or PDSE member, or z/OS UNIX file, and writes a message (GFSA791I or GFSA792I) for each unique userid and client host pair that holds locks, share reservations or delegations for it. Note that the currently active prefix settings apply for determining whether a z/OS UNIX file or an MVS data set is specified. The general command syntax is as follows:

```
►►──┬─MODIFY─┬──mvsnfs,listlock='file_name'─────────────────────────►◄
    └─F──────┘
```

where

file_name
Is the name of an MVS data set, z/OS UNIX file, PDS member, or PDSE member. File names specified with a prefix must be specified in single quotes with the prefix (/hfs or /mvs) followed by the z/OS UNIX path name or the MVS data set name, respectively. For example, if the HFS(/hfs) site attribute was specified, LISTLOCK='/hfs/u/user' will indicate the z/OS UNIX file /u/user.

- For an MVS data set, PDS member, or PDSE member:
  - If *file_name* references a z/OS conventional MVS data set for the first time, message GFSA921I is issued.

  - If *file_name* references a PDS or a PDSE data set (that does not include the member name), GFSA918I is issued.
  - If *file_name* references an invalid member name of a PDS or a PDSE data set, GFSA919I is issued.
  - If *file_name* is found and there are no locks, share reservations or delegations to list, message GFSA793I is issued.
  - When an NFS client closes a file, delegation is returned to the NFS Server. Assuming there is no local user access request, the delegation is kept by the NFS Server, and not returned to MVS. In this case, message GFSA790I is issued.
- For a z/OS UNIX file, if there are no locks to list, message GFSA793I is issued.
- If the specified file does not have any locks, message GFSA793I is issued to report that no locks exist.
- Since the lock information may only be reported to the server log data set or component trace buffer, and not back to the console, message GFSA794I is issued to indicate that the listlock function completed successfully.
- Response messages are sent to the console if there are 10 or fewer locks to be reported. Otherwise, the response messages are only sent to the NFS log data set.
- Before reading the output of this command, you need to flush the log using the flushlog operand.

Log operand

Use the **log** operand to set the level of NFS logging messages to be collected. Enter the operand as follows:

```
►►──┬─MODIFY─┬──mvsnfs,log=─┬─error────┬─────────────────────────────►◄
    └─F──────┘              ├─info─────┤
                            ├─warn─────┤
                            └─memstats─┘
```

where

log=error
Collects only error messages.

log=info
Collects error, attention, and informational messages.

log=warn
Collects error message and attention messages.

log=memstats
Requests the NFS Server to write memory usage statistics to the NFS Log.

The diagnostic message level can also be set in the execution statement of the NFS startup procedure. For more information, see Appendix K, “Capturing diagnostic information using z/OS NFS log data sets and from other components,” on page 523.

Smf operand

When the z/OS NFS server starts up, *smf* is set to off. You need to set *smf* to on if you want to collect the SMF records. The Network File System suspends writing SMF records when it detects an SMF error. The *smf* operand is used to resume or suspend the collection of SMF data. This operand has two parameters, ON and OFF. Enter the operand as follows.

```
►►──┬─MODIFY─┬──mvsnfs,smf=─┬─off─┬─[,─┬─none─────────┬─]──────────────────►◄
    └─F──────┘              └─on──┘    ├─all──────────┤
                                       └─subtype_list─┘
```

where

smf=on Resumes the collection of SMF records.

smf=off Suspends the collection of SMF records.

smf=none
No SMF records are to be produced.

smf=all All SMF NFS type 42 records are to be produced.

smf=subtype_list
A list of levels that are delimited by commas. At least one of the levels (user | file | audit) must be specified for a list, and the remaining levels are optional. Where:

- file: Produces file usages SMF records (subtype 7).
- user: Produces user session SMF records (subtype 8).
- audit: Produces file creation, removal, and rename records (subtype 26).

Sockhang operand

When the z/OS NFS server starts up, *sockhang* is set to off. Set *sockhang* to on if you want the server to create a dump if it detects a potential socket hang condition. This optimizes the ability to detect the situation and capture the necessary diagnostic trace data before the trace data is lost.

When *sockhang* is set to on, tracing levels Network, MVS (DFP_request and DFP_return), and Task_Flow (Dispatch, Resume, Schedule, and Suspend) are set on to ensure that the necessary diagnostic data is collected when a socket hang is first detected. When *sockhang* is subsequently set to off, tracing levels Network, MVS, and Task_Flow are reset to off; any previous settings of Network, MVS, and Task_Flow before *sockhang* was turned on must be re-set as needed. Detail level tracing provides the ideal setting to further isolate the source of a socket hang condition, but with a greater effect on system performance. Enter the operand as follows:

```
                                    ┌─off──┐
►►──┬─MODIFY─┬──mvsnsf,─sockhang=───┼─on───┼──────────────────────────────►◄
    └─F──────┘                      └─trap─┘
```

where:

sockhang=off
Does **not** create a dump if a socket hang is detected.

sockhang=on
Creates a dump if a socket hang is detected.

sockhang=trap
Creates a dump if a socket reference count discrepancy is detected.

Switchlog operand

When the log data set being used to collect z/OS NFS server debug trace record becomes full or unusable, you can use **switchlog** to switch the trace recording to another log data set. Enter the operand as follows.

```
►►──┬─MODIFY─┬──mvsnfs,switchlog──────────────────────────────────────►◄
    └─F──────┘
```

where

switchlog
Switches the z/OS NFS server debug trace recording from the current log data set to the next log data set. Prior to the switch, the in-storage debug trace information is flushed to disk and the current log data set is closed. Message GFSA930I is issued to report the log switch, including the previous and new log data set DD names, and debug trace recording continues in the next log data set.

Version operand

To aid in diagnostics, the *version* operand identifies the precise release and maintenance levels of the z/OS NFS server modules, including the latest APAR installed. Enter the operand as follows.

```
►►──┬─MODIFY─┬──mvsnfs,version──┬─────────┬───────────────────────────►◄
    └─F──────┘                  ├─=all────┤
                                └─=module─┘
```

where

version Returns information about the release level of the z/OS NFS server and the latest APAR installed on the server.

version=all
Returns a list of all the modules in the z/OS NFS server with their current level information. At the end of the list is information about the latest APAR installed.

version=module
Returns information about the APAR level of the specified module.

Displaying NFS trace information

NFS component trace status and active options can be displayed by using the MVS **DISPLAY TRACE, COMP=** *mvsnfs* command for the NFS server or the **DISPLAY TRACE, COMP=** *mvsnfsc* command for the NFS client. For more information on the DISPLAY TRACE command, refer to *z/OS MVS System Commands*, SA22-7627.

Chapter 12. Installation-wide exit routines for the z/OS NFS server

This topic helps you write installation-wide exits or replaceable modules that customize NFS processing. It contains product-sensitive programming interfaces that are provided by the server.

If there is no customization made to the installation-wide exit routines, we recommend that you remove them for better performance.

Requirements for NFS

Table 33 summarizes the installation-wide exit routines that are provided by the NFS.

Table 33. List of installation-wide exits

Source Module Name	Load Module Name	Parameter List Mapping Macro	Description
GFSAUXL	GFSAUXL	GFSAULOG GFSAUDSA	Login security exit
GFSAUXF	GFSAUXF	GFSAUSEC GFSAUDSA	File security exit

GFSAUDSA is a sample skeleton for a user storage block.

The installation-wide exit routines are shipped with the NFS and they contain the dummy skeleton code. The source modules reside in the prefix.NFSSAMP data set and the macros reside in prefix.NFSMAC data set. (The value of prefix depends on the installation.) Before modifying or replacing these exits, you should review the functions and processing of these exit routines carefully. These are basic requirements for all the NFS exit routines:

- Exit routines must reside in an authorized program library and be accessible by the z/OS LOAD macro. NFS and installation-wide exit routines receive control in problem state key 8. Installation-wide exits are run as an APF-authorized task, because the Network File System is APF-authorized. As with any APF-authorized program, your exits should not be link-edited with APF-authorization. Only the main task, NFS, should have that link edit attribute.
- Exit routines must be link-edited with AMODE(31).

  The installation-wide exit routines are entered in AMODE(31) and can reside above or below the 16M line depending on the requirements of the installation-wide exits themselves.
- Exit routines must be reentrant.
- Exit routines must follow the standard z/OS register save and restore convention. The standard z/OS registers convention is:
  - Register 1 contains the address of the exit parameter list.
  - Register 13 contains the address of the caller's save area.
  - Register 14 contains the caller's return address.
  - Register 15 contains the address of the entry point for this exit routine. The server does not use return codes stored in register 15, but includes a parameter in the parameter list for exits that supply return codes.

© Copyright IBM Corp. 1991, 2010

Notes:

1. Address parameters have null value (0) if the related data does not exist.
2. The length of each field can be found in the macros shipped. Field length can be changed in the future.

Sample link-edit JCL

Use the sample JCL shown in Figure 30 to assemble and link-edit the GFSAUXL and GFSAUXF load modules:

```
//jobname  JOB (job_and_user_accounting_information)
//EXITASM  PROC M=
//ASM      EXEC PGM=ASMA90,
//              PARM='RENT'
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DISP=SHR,DSN=source_library_name(&M)
//SYSLIB   DD   DISP=SHR,DSN=SYS1.MACLIB
//         DD   DISP=SHR,DSN=source_macro_library_name
//SYSLIN   DD   DISP=SHR,DSN=obj_library_name(&M)
//SYSUT1   DD   UNIT=SYSDA,
//              SPACE=(32000,(30,30))
//PEND
//         EXEC EXITASM,M=GFSAUXL
//         EXEC EXITASM,M=GFSAUXF
//stepname EXEC PGM=HEWL,
//         PARM='MAP,LIST,RENT,REUS'
//SYSPRINT DD SYSOUT=A
//SYSUT1   DD DSN=&SYSUT1,UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD  DD DSN=apf_library_name,DISP=OLD
//USER     DD DSN=obj_library_name,DISP=SHR
//SYSLIN   DD *
   INCLUDE USER(GFSAUXL)
   MODE    AMODE(31)
   ENTRY   GFSAUXL
   NAME    GFSAUXL(R)
   INCLUDE USER(GFSAUXF)
   MODE    AMODE(31)
   ENTRY   GFSAUXF
   NAME    GFSAUXF(R)
/*
```

Figure 30. Sample link-edit JCL for the NFS. This example uses High Level Assembler. To use Assembler H, replace `ASMA90` on the `EXEC` statement with `IEV90`. The rest of the JCL would be the same.

Storage blocks of the server exits

This section discusses how to use storage blocks of the NFS installation-wide exits.

Global exit block (GXB)

The GXB is obtained once during system initialization by the server login exit. The exit returns a word to the server. This word is referred to as ‘the address of the GXB’, but the system initialization exit might store any value in the word. The address of the GXB is returned to the server in the parameter list and passed back to the installation-wide exits in each subsequent call. This block contains user installation-wide exit data that is needed to communicate with the server.

The GXB can contain an area to save the user data needed for all sessions. The usage of this block is determined by the exit. Access to the Global Exit Block must be controlled by the user-written installation-wide exits to ensure that updates are serialized and do not interfere with each other. This block is shared with the file security Installation-Wide Exit. The format of the GXB is entirely controlled by the login and file security installation-wide exits.

User exit block (UXB)

During a *Start of New User Session* request or a *User Login Request* or an *RPCSEC_GSS User Session Request* (see Note below), the exit can obtain a User Exit Block. The exit returns a word to the server. This word is referred to as 'the address of the UXB', but the system initialization exit might store any value in the word. The address of the UXB is returned to the server in the parameter list (depends on which request comes first), and is passed back to the installation-wide exits on each subsequent call related to this combination of machine and user IDs.

Note: An RPCSEC_GSS User Session is the first RPCSEC_GSS request coming into the NFS from a unique combination of the user as identified by the user's Kerberos Principal and the Internet address.

The UXB can contain an area to save the user data needed for this session. The usage of this block is determined by the exit. The exit is responsible for obtaining, and freeing access to these storage areas. This block is used by the login installation-wide exit and file security installation-wide exit. The format of the UXB is entirely controlled by the login and file security installation-wide exits.

Login installation-wide exit

The exit routine can invoke a customized authorization facility. The server mainline code can be set to perform Security Authorization Facility (SAF) checking, by specifying the **security** attribute in the attributes table.

If **security(saf)** or **security(safexp)** is specified in the attributes table and the exit routines exist, these exit routines get control first, and then SAF security checking gets control. If the exit routines fail the request, the entire request fails. If the exit routines process the request successfully, then the request is processed by the SAF checking. Similarly, if the SAF checking fails the request, the entire request fails.

If neither **security(saf)** nor **security(safexp)** is specified in the attributes table and the exit routine exists, this exit routine determines whether the request is successful or fails.

Figure 31 on page 242 shows the logic flow that determines which login checking routines are used.

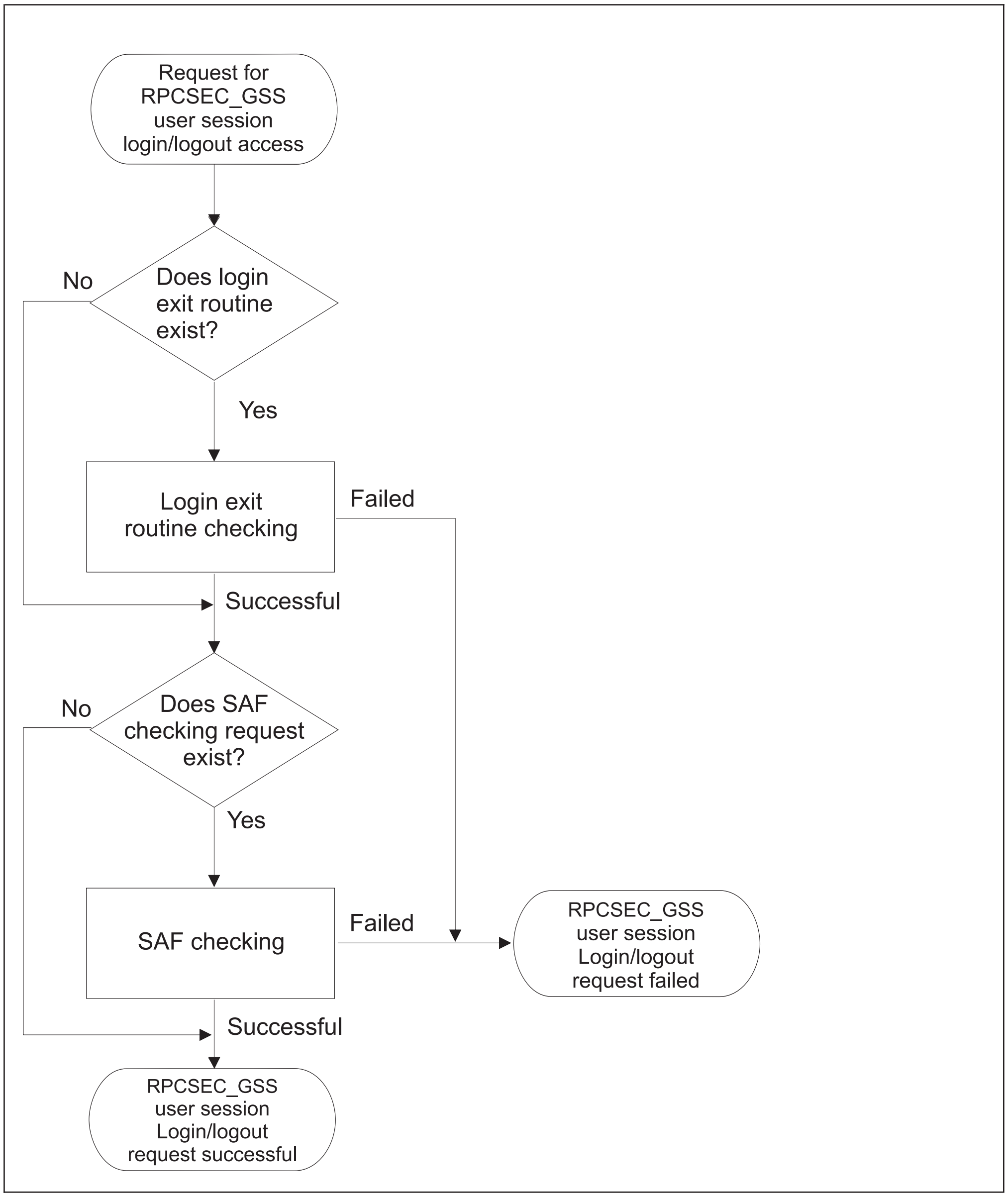

Request for RPCSEC_GSS user session login/logout access
No
Does login exit routine exist?
Yes
Login exit routine checking
Failed
Successful
No
Does SAF checking request exist?
Yes
SAF checking
Failed
RPCSEC_GSS user session Login/logout request failed
Successful
RPCSEC_GSS user session Login/logout request successful

Figure 31. Determining which login checking routines are used

The login installation-wide exit has a parameter list which is passed from the server to the installation-wide exit. The login installation-wide exit can be invoked for any of the following conditions:

System initialization
: Performed once during the initialization of the server and allows a Global Exit Block to be obtained. The GXB address is always returned to the installation-wide exits (see GXB in "Requirements of the login exit" on page 243). If this request fails, both the login installation-wide exit and the file security installation-wide exit are marked as non-existent.

New user session
: Performed when a unique combination of UNIX UID and Internet address is detected. The exit might obtain a User Exit Block for use by later calls if the UXB does not exist.

Login
: Performs user verification when a client tries to use either an **mvslogin**

command or a PCNFSD request or an RPCSEC_GSS user session request (see 241). The exit might obtain a User Exit Block for use by later calls if the UXB does not exist.

Logout
Performs cleanup when a client tries to use the **mvslogout** command or a timeout occurs. Timeout is the value specified in the logout attribute in the attributes table. On a logout, the UXB is released. Logout can also be initiated by the login request. For RPCSEC_GSS request, logout is only initiated when a timeout occurs since mvslogin/mvslogout is no longer required.

System termination
Occurs once during the termination of the server, and causes the Global Exit Block to be freed.

Requirements of the login exit

Besides the requirements stated in "Requirements for NFS" on page 239, the login installation-wide exit routine must be link-edited with the name of GFSAUXL.

Options of the login exit

The login installation-wide exit routine can perform these functions:

- Obtain a Global Exit Block (GXB)
- Obtain a User Exit Block (UXB)

  If the User Exit Block already exists, its address is passed to the installation-wide exit routine; otherwise, 0 is passed. The installation-wide exit routine can accept or reject the request. If the request is accepted and the User Exit Block does not exist, you can allocate the User Exit Block and return the address in the LEDSXS parameter list.

Structure of the login exit message

The message supplied by the login exit is sent to the NFS server log data set and component trace, if activated. The message has the label name LEDSXD and consists of these two fields:

LEDSXCNT
1 byte message length, excluding the NULL character (X'00').

LEDSXMGS
81 byte NULL character string. The installation-wide exit can fill in this message field with the message ended by one or more bytes of X'00'.

If the format of the message is incorrect, both the login installation-wide exit and the file security installation-wide exit are marked as nonexistent, and a message (GFSA990I or GFSA991E) is sent to the z/OS NFS server log data set and to the console.

Contents of the login exit parameter list

The parameter list is mapped by macro and DSECT GFSAULOG. Table 34 on page 244 describes the contents of each field.

For configurations that use Internet Protocol Version 4, the client IP address is in field LEDSIA. If your configuration uses IPv6, refer to field LEDSIA6 for the client IP address. When an IPv6 address is provided in LEDSIA6, LEDSIA takes a value of -1.

If the **dhcp** site attribute is specified for the NFS server to use dynamic client IP addresses, the contents of the client IP address field will be correct when the exit parameter list request is processed. However, the exit must not have any dependency on the persistence of the IP address value beyond the duration of the individual request. That IP address can change between requests.

Table 34. Format of login installation-wide exit routine parameter list

Field Name	Description	Contents
LEDSRQ	Request code	System request code set by the server before calling this installation-wide exit, for these conditions: **4** System initialization **8** System termination **12** Start of new user Session **16** User login request **20** Logout has been requested
LEDSRC	Return code	Codes generated and returned by the calls.
LEDSM	Client machine name	Character string ended by single byte containing X'00'.
LEDSIA	Client IP address	Number (32-bit Internet address). Contains -1 (X'0xFFFFFFFF) if an IPv6 address is provided in the LEDSIA6 field.'
LEDSU	UNIX Client user ID number	Number
LEDSG	UNIX Client group ID number	Number
LEDSMU	z/OS user ID	Character string padded with a blank at the end of the user ID,conforming to z/OS standards.
LEDSMG	z/OS group name	Character string padded with a blank at the end of the group name, conforming to z/OS standards.
LEDSXS	Address of UXB	Size and content are installation-dependent, generated at the start of a new user session or a user login request.
LEDSXG	Address of GXB	Size and content are installation-dependent, generated at system initialization.
LEDSXD	Message supplied by this exit routine	Message structure (see “Structure of the login exit message” on page 243).
LEDSVERS	z/OS version number	Number (*0x17* for z/OS V1R7)
LEDSIA6	Client IP address	Number (128-bit Internet address, for Internet Protocol V6 users)
Note: Character strings are in upper-case EBCDIC.		

Login exit parameter list

This section describes how the Global Exit Block (GXB) or User Exit Block (UXB) is constructed and used between the login installation-wide exit and the NFS. A request code is set by the server before each call to this installation-wide exit routine. The installation-wide exit routine provides a return code for each event.

Request codes to the login exit

Table 35 on page 245 shows the login installation-wide exit request codes and their meanings:

Table 35. Request codes to the login exit

Code	Meaning
4 (X'04')	System initialization
8 (X'08')	System termination
12 (X'0C')	Start of new user session
16 (X'10')	User login request
20 (X'14')	User logout request

IBM might add new request codes in a future level of the server. To provide for this, consider making your exit set a return code 0 if it does not recognize the request code.

Return codes from the login exit

Table 36 shows the login installation-wide exit routine return codes and their meanings:

Table 36. Return codes from the login exit

Code	Meaning
0 (X'00')	Request successful
4 (X'04')	Request unsuccessful

System initialization routine of the login exit

The system initialization routine is called each time the server address space starts, and can acquire and initialize the GXB. Table 37 shows the codes and fields that are used for this routine:

Table 37. Codes and fields for system initialization

When Invoked	Field Name	Description	Contents
On entry	LEDSRQ	Request code	**4**
On exit	LEDSXD	Exit-supplied error message	Message Structure
On exit	LEDSRC	Return code	**0** Initialization successful **4** Stop the NFS.
On exit	LEDSXG	Global Exit Block	Address

Start of new user session routine of the login exit

The start of new user session routine is called when a new client machine-user combination is recognized by the server. The exit might acquire and initialize a UXB. This is needed only for system authentication, not for RPCSEC_GSS authentication. Table 38 shows the codes and fields that are used for this routine:

Table 38. Codes and fields for start of new user session

When Invoked	Field Name	Description	Contents
On entry	LEDSRQ	Request code	**12**
On entry	LEDSM	Client machine name	EBCDIC character string
On entry	LEDSIA	Client IP address	Number
On entry	LEDSU	UNIX Client User ID number	Number

Table 38. Codes and fields for start of new user session (continued)

When Invoked	Field Name	Description	Contents
On entry	LEDSG	UNIX Client Group ID number	Number
On entry	LEDSXS	User Exit Block	0
On entry	LEDSXG	Global Exit Block	Value
On entry	LEDSIA6	Client IP address	Number (128-bit Internet address, for Internet Protocol V6 users)
On exit	LEDSXD	Exit-supplied error message	Message Structure
On exit	LEDSRC	Return code	**0** New session established **4** New session not established
On exit	LEDSXS	User Exit Block	Value, if return code in LEDSRC is 0

User login request routine of the login exit

This routine is called when the **mvslogin** command or PCNFSD request or RPCSEC_GSS user session request is used. The installation security system should be called to determine if the user is properly authorized. Table 39 shows the codes and fields:

Table 39. Codes and fields for user login request

When Invoked	Field Name	Description	Contents
On entry	LEDSRQ	Request code	**16**
On entry	LEDSM	Client machine name	Character string
On entry	LEDSIA	Client IP address	Number
On entry	LEDSU	UNIX Client User ID number	Number
On entry	LEDSG	UNIX Client Group ID number	Number
On entry	LEDSMU	z/OS User ID	Character string
On entry	LEDSMG	z/OS Group Name	Character string
On entry	LEDSXS	User Exit Block	Address or 0
On entry	LEDSXG	Global Exit Block	Address
On entry	LEDSIA6	Client IP address	Number (128-bit Internet address, for Internet Protocol V6 users)
On exit	LEDSXD	Exit-supplied error message	Message Structure
On exit	LEDSRC	Return code	**0** *Login* successful **4** *Login* failed
On exit	LEDSXS	User Exit Block	Value

User logout request routine of the login exit

This routine is used at logout to return the User Exit Block storage obtained at the start of the session and to perform any related logout processing. The codes and fields used are shown in Table 40.

Table 40. Codes and fields for logout request

When Invoked	Field Name	Description	Contents
On entry	LEDSRQ	Request code	**20**
On entry	LEDSM	Client machine name	Character string
On entry	LEDSIA	Client IP address	Number
On entry	LEDSU	UNIX Client User ID number	Number
On entry	LEDSG	UNIX Client Group ID number	Number
On entry	LEDSMU	z/OS User ID	Character string
On entry	LEDSMG	z/OS Group Name	Character string
On entry	LEDSXS	User Exit Block	Address or 0
On entry	LEDSXG	Global Exit Block	Address
On entry	LEDSIA6	Client IP address	Number (128-bit Internet address, for Internet Protocol V6 users)
On exit	LEDSXD	Exit-supplied error message	Message Structure
On exit	LEDSRC	Return code	**0** *Logout* successful
On exit	LEDSXS	User Exit Block	**0** UXB is released

System termination routine of the login exit

This routine is used at server termination to release the GXB storage. All users are automatically logged off. The codes and fields used are shown in Table 41.

Table 41. Codes and fields for system termination

When Invoked	Field Name	Description	Contents
On entry	LEDSRQ	Request code	**8**
On entry	LEDSXG	Global Exit Block	Address
On exit	LEDSXD	Exit-supplied error message	Message Structure
On exit	LEDSRC	Return code	**0** Exit termination successful
On exit	LEDSXG	Global Exit Block	**0** GXB is released

File security installation-wide exit

The file security installation-wide exit routine verifies that a user is authorized to access a data set or data set member with the access mode requested. If the request from allocation, write, read, or access does not have permissions set up, then the exit routine gets control.

This exit applies only to MVS data set access, not to z/OS UNIX file access.

The permissions set up by the file security exit can be overridden by the SAF checking. If the exits allow access and there is no SAF checking, the permissions remain in effect until logout. The server does not call again for the same access before logout. The server gets the access mode or permissions before any of the other three types of calls.

If **security(saf)** or **security(safexp)** is specified in the attributes table and the exit routine exists, this exit routine gets control first, and then SAF security checking gets control. If the exit routines fail the request, the entire request fails. If the exit routines process the request successfully, then the request is processed by the SAF checking. Similarly, if the SAF checking fails the request, the entire request fails. If the SAF checking is successful, the file permissions from the SAF checking are set up for the request.

If neither **security(saf)** nor **security(safexp)** is specified in the attributes table and the exit routine exists, this exit routine determines the permissions.

Figure 32 on page 249 shows the logic flow determining which file security checking routines are used.

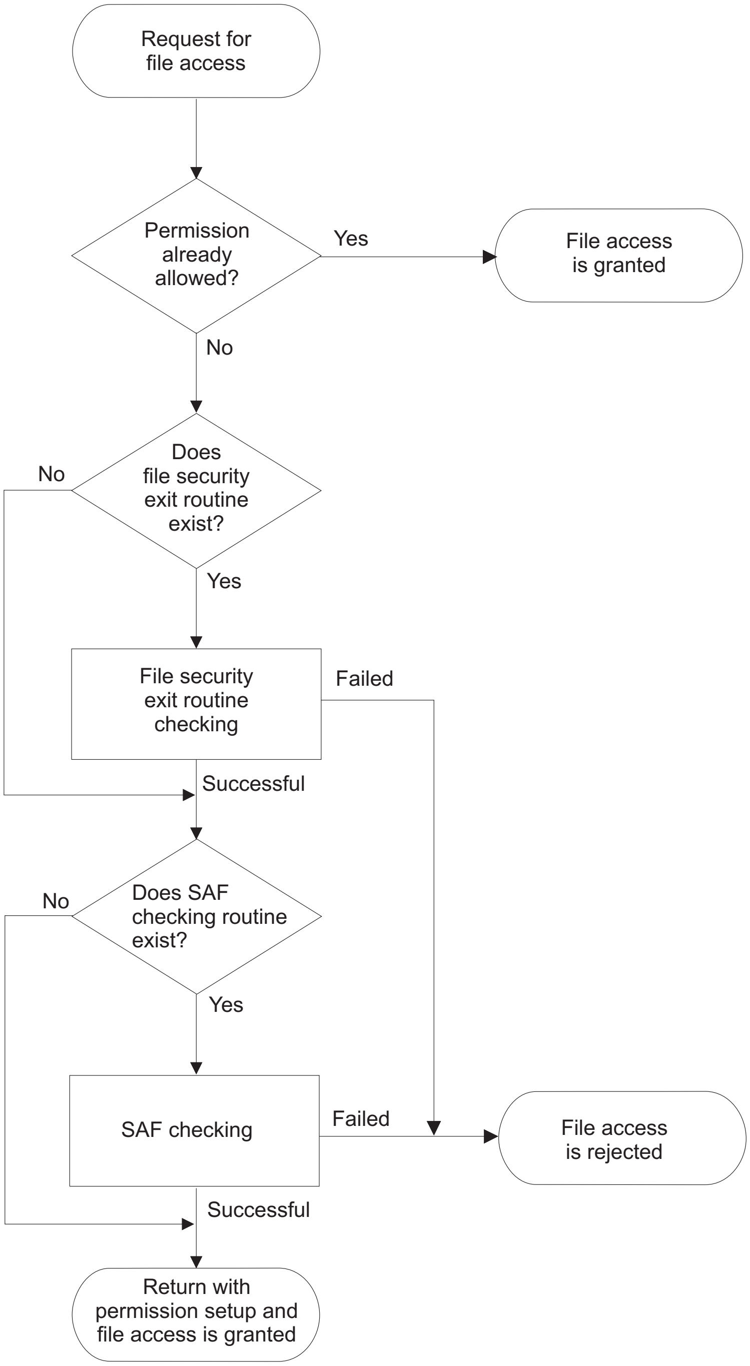

Request for file access
Permission already allowed?
Yes
File access is granted
No
Does file security exit routine exist?
No
Yes
File security exit routine checking
Failed
Successful
Does SAF checking routine exist?
No
Yes
SAF checking
Failed
File access is rejected
Successful
Return with permission setup and file access is granted

Figure 32. Determining which file security checking routines are used

The file security installation-wide exit uses the parameter list which is used by the server installation-wide exit. This exit is called for at these times.

Data Set Allocation
When a client user creates, renames, or deletes a data set.

Data Set Write
When a client user first writes to an MVS data set or data set member.

Data Set Read
When a logged in user tries to first read from an MVS data set or data set member.

Getting Access Mode or Permissions

This function is called first to set up permission for the read, write, create, delete, or rename data set request. The server needs to obtain the access mode or the permissions that a user has for a data set. This function is only called when SAF is not being used.

The following show access modes and what they permit.

Allocate	Read, write, create, delete, or rename the data set.
Write	Read or write the data set
Read	Read the data set

A return code is set by the installation-wide exit indicating whether the request is allowed. The file security installation-wide exit is not called at server startup or shutdown.

Requirements of the file security exit

Besides "Requirements for NFS" on page 239, the following requirements must be fulfilled.

- The file security installation exit must be link-edited with the name of GFSAUXF.
- The login exit must exist (GFSAUXL).
- If the GXB or UXB are to be used, the login exit must obtain them.
- The UXB and GXB are shared with the login installation-wide exit.

Structure of the file security exit message

The message supplied by the file security exit is sent to the NFS server log data set and component trace, if activated. The message has the label name FEDSXD and consists of the following two fields.

FEDSXCNT
1 byte message length, excluding the NULL character (X'00').

FEDSXMGS
81 byte NULL character string. The installation-wide exit can fill in this message field with an error message ended by one or more bytes of X'00'.

If the format of the message is incorrect, the file security installation-wide exit is marked as nonexistent, and a message (GFSA990I or GFSA991E) is sent to the NFS server log data set and to the console, and to component trace if activated.

Contents of the file security exit parameter list

The file security installation-wide exit parameter list is mapped by macro and DSECT GFSAUSEC. Table 42 on page 251 describes each field.

For configurations that use Internet Protocol Version 4, the client IP address is in field FEDSIA. If your configuration uses IP version 6, refer to field FEDSIA6 for the client IP address. When an IPv6 address is provided in FEDSIA6, FEDSIA takes a value of -1.

If the **dhcp** site attribute is specified for the NFS server to use dynamic client IP addresses, the contents of the client IP address field will be correct when the exit parameter list request is processed. However, the exit must not have any dependency on the persistence of the IP address value beyond the duration of the individual request. That IP address can change between requests.

Table 42. Format of the parameter list for the file security installation-wide exit

Field Name	Description	Contents
FEDSRQ	Request code	System request code set by the server before each call to this exit for these conditions. **4** Validate allocate request **8** Validate write request **12** Validate read request **16** Return the current settings of the security permissions
FEDSRC	Return code	Codes are generated and returned by the exit routine.
FEDSM	Client system name	Character string ended by a single byte containing X'00'.
FEDSIA	Client IP address	Number (32-bit Internet address). Contains -1 (X'0xFFFFFFFF) if an IPv6 address is provided in the FEDSIA6 field.'
FEDSU	Client user ID number	Number
FEDSG	Client group ID number	Number
FEDSMU	z/OS user ID	Character string padded with a blank at the end of the user ID, conforming to z/OS standards.
FEDSDN	MVS data set name	Character string ended by a single byte containing X'00'. This conforms to z/OS standards.
FEDSMN	MVS data set member name	Character string ended by a single byte containing X'00'. This conforms to z/OS standards.
FEDSXS	Address of UXB	Size and contents are installation dependent. Generated at the start of a user session or user login by the *login* exit routine.
FEDSXG	Address of GXB	Size and content are installation dependent. Generated at server initialization by the *login* exit routine.
FEDSTYPE	File Type	Address of fullword integer defining the data set organization: **0** Not used **1** Sequential **2** Partitioned **3** Direct **4** Reserved **5** VSAM (if unable to classify further) **6** VSAM ESDS **7** VSAM RRDS **8** VSAM KSDS **9** Reserved **10** Reserved
FEDSRPM	UNIX Permissions	UNIX file modes of type mode_t as defined by POSIX.
FEDSXD	Message supplied by this exit routine	Address of the message structure (see “Structure of the file security exit message” on page 250).
FEDSVERS	z/OS version number	Number (*0x17* for z/OS V1R7)

Table 42. Format of the parameter list for the file security installation-wide exit (continued)

Field Name	Description	Contents
FEDSIA6	Client IP address	Number (128-bit Internet address, for Internet Protocol version 6 users)
Note: Character strings are in upper-case EBCDIC.		

File security exit parameter list

These sections each describe how the data in the parameter list is used by the server and the file security installation-wide exit routine. A request code is set by the server before each call to this installation-wide exit routine.

Each topic describes an event, for which some fields are set on entry. The file security installation-wide exit provides a return code for each event.

Request codes to the file security exit

Table 43 shows the request codes to the file security installation-wide exit:

Table 43. Request codes to the file security exit

Code	Meaning
4 (X'04')	Validate allocate request
8 (X'08')	Validate write request
12 (X'0C')	Validate read request
16 (X'10')	Return the current settings of the security permissions

Return codes from the file security exit

Table 44 shows the request codes returned by the file security installation-wide exit:

Table 44. Return codes from the file security exit

Code	Meaning
0 (X'00')	Request successful
4 (X'04')	Request unsuccessful

Validate allocate request routine of the file security exit

This routine is called when a user allocates, renames, or deletes a data set or data set member. The codes and fields used are shown in Table 45.

Table 45. Codes and fields for validate allocate request

When Invoked	Field Name	Description	Contents
On entry	FEDSRQ	Request code	**4**
On entry	FEDSM	Client machine name	Character string
On entry	FEDSIA	Client IP address	Number
On entry	FEDSU	Client user ID number	Number
On entry	FEDSG	Client group ID number	Number
On entry	FEDSMU	z/OS user ID	Character string
On entry	FEDSDN	MVS data set name	Character string
On entry	FEDSMN	MVS data set member name	Character string

Table 45. Codes and fields for validate allocate request (continued)

When Invoked	Field Name	Description	Contents
On entry	FEDSXS	User Exit Block	Address
On entry	FEDSXG	Global Exit Block	Address
On entry	FEDSTYPE	File type	Number
On entry	FEDSIA6	Client IP address	Number (128-bit Internet address, for Internet Protocol version 6 users)
On exit	FEDSXD	Exit-supplied error message	Message Structure
On exit	FEDSRC	Return code	**0** Access allowed **4** Access denied

Validate write request routine of the file security exit

This routine is called when a user writes to an MVS data set or data set member. The codes and fields used are shown in Table 46.

Table 46. Codes and fields for validate write request

When Invoked	Field Name	Description	Contents
On entry	FEDSRQ	Request code	**8**
On entry	FEDSM	Client machine name	Character string
On entry	FEDSIA	Client IP address	Number
On entry	FEDSU	Client User ID number	Number
On entry	FEDSG	Client Group ID number	Number
On entry	FEDSMU	z/OS user ID	Character string
On entry	FEDSDN	MVS data set name	Character string
On entry	FEDSMN	MVS data set member name	Character string
On entry	FEDSXS	User Exit Block	Address
On entry	FEDSXG	Global Exit Block	Address
On entry	FEDSTYPE	File type	Number
On entry	FEDSIA6	Client IP address	Number (128-bit Internet address, for Internet Protocol version 6 users)
On exit	FEDSXD	Exit-supplied error message	Message Structure
On exit	FEDSRC	Return code	**0** Access allowed **4** Access denied

Validate read request routine of the file security exit

This routine is called when a user reads from an MVS data set or data set member. The codes and fields used are shown in Table 47.

Table 47. Codes and fields for validate read request

When Invoked	Field Name	Description	Contents
On entry	FEDSRQ	Request code	**12**
On entry	FEDSM	Client machine name	Character string

Table 47. Codes and fields for validate read request (continued)

When Invoked	Field Name	Description	Contents
On entry	FEDSIA	Client IP address	Number
On entry	FEDSU	Client user ID number	Number
On entry	FEDSG	Client group ID number	Number
On entry	FEDSMU	z/OS user ID	Character string
On entry	FEDSDN	MVS data set name	Character string
On entry	FEDSMN	MVS data set member name	Character string
On entry	FEDSXS	User Exit Block	Address
On entry	FEDSXG	Global Exit Block	Address
On entry	FEDSTYPE	File type	Number
On entry	FEDSIA6	Client IP address	Number (128-bit Internet address, for Internet Protocol version 6 users)
On exit	FEDSXD	Exit-supplied error message	Message Structure
On exit	FEDSRC	Return code	**0** Access allowed **4** Access denied

Return security permissions routine of the file security exit

This routine is called to query a data set access mode or permission. The codes and fields used are shown in Table 48.

Table 48. Codes and fields for return security permissions

When Invoked	Field Name	Description	Contents
On entry	FEDSRQ	Request code	**16**
On entry	FEDSM	Client machine name	Character string
On entry	FEDSIA	Client IP address	Number
On entry	FEDSU	Client user ID number	Number
On entry	FEDSG	Client group ID number	Number
On entry	FEDSMU	z/OS user ID	Character string
On entry	FEDSDN	MVS data set name	Character string
On entry	FEDSMN	MVS data set member name	Character string
On entry	FEDSXS	User Exit Block	Address
On entry	FEDSXG	Global Exit Block	Address
On entry	FEDSTYPE	File type	Number
On entry	FEDSIA6	Client IP address	Number (128-bit Internet address, for Internet Protocol version 6 users)
On exit	FEDSXD	Exit-supplied error message	Message Structure
On exit	FEDSRC	Return code	**0** Exit processing successful **4** Exit failed

Table 48. Codes and fields for return security permissions (continued)

When Invoked	Field Name	Description	Contents
On exit	FEDSRPM	New permissions in mode_t format	UNIX file modes of type mode_t as defined by POSIX

Part 3. Performance Tuning

Chapter 13. Performance tuning in the NFS environment 259
What is performance tuning? 259
How is performance characterized? 259
What is the NFS environment? 260
How to tune for performance 260
Impact of the NFS protocol on performance . . . 262

Chapter 14. Optimizing the NFS environment 265
Network performance tuning 265
NFS client system performance tuning 267
NFS server system performance tuning. 268
z/OS constraints 269

Chapter 15. Evaluating z/OS NFS performance 271
Evaluating throughput 271
Single process throughput 271
Multiple process throughput 272
Multiple client throughput 272
Evaluating NFS command response time 272
Evaluating CPU utilization 272
Collecting server usage data 273

Chapter 16. Tuning the z/OS NFS server . . . 275
Data set creation attributes 275
Block size and record length 275
Record format 276
Data set organization and data set type. . . . 276
B37/D37/E37 ABEND handling 277
| NFS server cache monitoring and reporting . . 277
Processing attributes 277
Character translation 278
File size determination 278
Data set timeout specification 279
Accessing migrated files. 280
Asynchronous z/OS UNIX processing 280
Site attributes 281
Buffer usage and caching 281
Ordering out-of-sequence data. 282
Storage considerations 284
Subtasking 284

Chapter 17. Tuning the z/OS NFS client. . . . 287
Caching 287
Dynamicsizeadj. 287
Bufhigh 288
Biod 288
Readahead 288
Delaywrite 288
Vers 289
Wsize and rsize 289

© Copyright IBM Corp. 1991, 2010

Chapter 13. Performance tuning in the NFS environment

This topic explains performance tuning within the context of the NFS client-server environment. It provides guidelines on the performance expectation of the underlying hardware and software products on which an NFS client-server implementation is dependent. These guidelines specifically note the limitations to performance tuning and describe processes to tune z/OS NFS.

What is performance tuning?

In general, performance tuning improves the price to performance ratio for a system or set of services by reallocating the available computing, network, or storage resources. The reallocation of these resources not only improves the performance for a particular load of work, but also accommodates an increase in the amount of work to be performed with minimal acquisition of additional resources. The information acquired from performance tuning can also be an important basis for long range capacity planning.

The following might be some possible reasons to do performance tuning.

- Access more data over existing networks
- Improve response time for particular applications or groups of users
- Better utilize the available storage capacity
- Minimize cost for additional services or functional capability

How is performance characterized?

Performance is the manner in which a process, system, processor, network, or device behaves for a particular load or unit of work. To measure, or quantify, performance, we monitor the length of time for a unit of work to complete. If units of work are being shared, we monitor the amount of time waiting for a resource to be available to perform a unit of work. A unit of work is a specific activity or action that we expect a process, system, processor, network, or device to perform. This could be something as granular as an I/O request, sending or receiving a buffer of data over a network, or processing an NFS request. We are frequently interested in the performance of a particular set of work activities which we will refer to as a load of work, or workload.

When a particular workload has been identified for performance measurement, we can determine the performance metrics, or units of measurement, that are relevant to that workload. Some examples of the following performance metrics that might be used in reference to performance tuning for z/OS NFS are:

throughput	Number of units of work completed per unit of time.
response time	Elapsed time necessary to complete a unit of work.
CPU time	Amount of time the processor spends executing instructions.
instructions per byte	Total number of CPU instructions to process data divided by the number of bytes of data processed.
aggregate throughput	Sum of the throughput measurements for multiple processes.
NFS operations per second (NFSops)	Number of NFS requests processed by an NFS server divided by the total time in seconds to process the requests.

© Copyright IBM Corp. 1991, 2010

Measuring performance metrics can be as simple as using your watch to time the execution of a particular command or as complex as using specialized hardware and software tools to monitor and extract a diversity of performance metrics. Chapter 15, “Evaluating z/OS NFS performance,” on page 271 will address some methods that can be used to evaluate the performance of z/OS NFS. The method selected will depend on the complexity of the workload and the monitoring tools available at your installation.

What is the NFS environment?

The NFS environment includes the NFS client system(s), the mix of networks available, the NFS server system(s), and the manner in which they are configured. While this guide is intended for z/OS NFS, it is important to know the performance limitations within the NFS environment to determine the necessity of tuning z/OS NFS.

NFS client systems range from single user desktop personal computers to large scale processors with many users. These NFS client systems typically support multiple applications as well. Clearly, the NFS client system resources will be shared between its users and/or applications. These resources include available physical storage, memory, processing capability, and network access. The NFS client is one application, with a possibility for many users, that must share the memory, processing, and network resources in exchange for providing access to additional physical storage on other systems. The degree to which the NFS client application must share such resources will affect performance for NFS client users and applications.

The NFS server system, like the NFS client system, must share resources with other users and applications. NFS server application performance will be affected by the amount of contention over system resources and by the priority established for the NFS server application. The overall performance of an NFS server is also influenced by the number of NFS clients for which it provides services.

The network(s) over which NFS clients access NFS servers also affect overall performance in the NFS environment. Such networks can be homogenous, consisting of a single network medium, or heterogeneous, consisting of a mix of network mediums. Each network medium type has an expected maximum capacity, or *bandwidth*. For instance, the capacity of a Fast Ethernet Ring network may be 16 megabits per second (Mbps) or 4 Mbps, and the capacity of a Fiber Distributed Data Interface (FDDI) network is 100 Mbps. When different network mediums are combined in a more complex network environment, the capacity for a fixed route over the network is limited by the network segment with the smallest capacity. For example, a route over a network consisting of both 4 and 16 Mbps Token Rings and a FDDI backbone will have a maximum capacity equivalent to that of the Token Ring, or 4 Mbps. When bridges, routers, and gateways are included in a network configuration, their capacity must also be considered. Such devices must also be considered when tuning performance in a network environment, particularly if a device does not support increased network buffer sizes.

How to tune for performance

Given the complexity of the NFS environment, it is important to establish a methodology for tuning performance. The following steps provide a guideline that highlights particular areas relevant to the NFS environment. Implementing the guideline may involve more than one person or support organization.

Identify Performance Requirements: Before you begin performance tuning in general, and particularly z/OS NFS, determine those areas where performance is unsatisfactory. This is a good time to establish more precise performance requirements. As users and application requirements are identified, it can be advantageous to rank or group them according to their requirements.

Know the NFS Environment: In the previous section, the NFS environment was discussed. It is very important to fully understand the performance of the existing NFS environment, particularly that of the network. Such analysis can eliminate unnecessary tuning of the NFS client and server systems.

Establish Performance Objectives: Once the performance requirements and the NFS environment are known, you are in a position to define and prioritize your performance objectives. The performance objectives should be specified in a manner that is quantifiable and measurable. Keep in mind that you will need an executable workload or test scenario to evaluate the effectiveness of your performance tuning.

Define Workloads and Test Scenarios: You may already have workloads or test scenarios depending on how requirements and/or objectives were defined. However, these may be unwieldy or impractical to use for your performance tuning purposes. Therefore, it is advantageous to spend some time initially to define some simplified test cases that can be executed in a repeatable fashion and with as much control as is feasible. Simple test cases are also useful in diagnosing performance problems, particularly to locate an area within the NFS environment that may be impacting performance.

Select Monitoring Tools: While you may only be interested in the performance of z/OS NFS, you will find it useful to have access and familiarity with a variety of performance hardware and software monitoring tools. Not only will such knowledge assist you in collecting data to evaluate performance, but it will also help you to identify areas that may be impacting the performance of z/OS NFS. Minimally, a set of monitoring tools must be identified to collect the data upon which performance tuning decisions will be made and to determine the effectiveness of tuning.

Collect Performance Data: At this point you should be ready to begin collecting performance data. Initial measurements will be the starting point, or *baseline*, that will be used to evaluate the effect of your tuning. Since there can be a significant variation in network and system performance, it is prudent to repeat a performance measurement to establish the degree of variation inherent in the measurement. Doing this will provide a sense of whether or not future tuning is really affecting performance or simply normal variation.

While it may be convenient to collect performance data when systems and networks are idle, this is probably not practical. However, it is useful to collect data during peak and low activity periods. If user or application requirements are related to a specific time period, or sensitive to other system and network activity, data should be collected for these periods of time, as well. Remember that redistributing the workload may be the most cost effective approach to performance tuning.

Evaluate Performance Data: This may seem like an obvious step. However, keep in mind that the NFS environment may be quite complex. The more complex the NFS environment and your test cases are, the more data there will be to evaluate. You may also have both client and server data to evaluate as well as network data.

As you begin evaluating performance data you have collected, look for areas or opportunities where performance could be improved. Before attempting to tune z/OS NFS, you should determine if performance is primarily impacted by the NFS server system, the NFS client system, or the network itself. In fact, you may determine that additional data must be collected prior to any performance tuning. You may also discover evidence that configurations and parameter settings within the evaluation environment are not optimal.

The output of this step is a list of changes that you believe will positively affect performance. As you make this list, identify the impact or cost associated with a change. It is also useful to identify any resources that are heavily utilized or that have contending requirements. This will help you to prioritize and ultimately select the changes you will make or recommend.

Tune Your NFS Environment: Make one change from your list of possible tuning changes at a time. Measure and evaluate the effect of that change before making any other changes. Pay particular attention to any impact on heavily utilized resources that may have been previously identified in addition to newly exposed resources that now appear to be impacted. Since performance tuning typically involves a trade off in resource utilization, make sure you have not inadvertently caused a performance problem elsewhere. Also, before deciding to implement a change, consider whether any observed changes are due to the tuning change you've made or simply a result of normal measurement variation. Repeat this step as necessary.

Impact of the NFS protocol on performance

Command response time is of particular importance to NFS client users. The longer a user waits for the results of a particular command, the more important this will become. The nature of transparent access with the NFS protocol results in users not necessarily being aware of the impact on performance caused by the network and the NFS server system.

Also, while users are generally knowledgeable of the commands supported by the NFS client operating system, they may have no knowledge of the NFS commands, or procedures, that are executed as a result of one simple user command. In fact, one user command typically results in execution of multiple NFS commands.

Another impact on command response time is the NFS protocol itself; version 2 and version 3 of the NFS protocol are intended to be as stateless as possible. This means that a stateless server operates correctly without maintaining any protocol state information for its clients. A stateless protocol was originally selected to minimize the probability of data losses due to a server crash. In NFS version 4, some state information is introduced into the protocol (for example, the open/close, lock, and setclientid operations).

For NFS version 2 and version 3 protocols, the stateless nature of the NFS server places the responsibility of keeping track of NFS commands on the NFS client. To do this NFS client implementations generally wait a period of time for a response to a particular NFS command. If a response is not received within this period of time, the NFS client will retransmit the NFS command. This process is repeated for a fixed number of times or until a response is received.

NFS servers and clients have typical methods of queuing requests and responses as part of the underlying protocol layers. When these queues are full, incoming requests or responses are dropped. The NFS client and server do not know when responses or requests are dropped. Both rely on the stateless protocol whereby the client will eventually retransmit the NFS request again. Under these conditions the NFS client is waiting for a response that will never be received. Clearly, waiting to retransmit an NFS command, particularly multiple times, will negatively affect the response time for the initial user command.

To determine whether or not NFS client users are being impacted by the situation previously described, most NFS client implementations provide a **nfsstat** command to monitor NFS statistics. Figure 33 shows the output from the ***nfsstat -c*** command.

```
USER1:/u/user1:>nfsstat -c

Client rpc:
Connection oriented
calls      badcalls   badxids    timeouts   newcreds   badverfs   timers
0          0          0          0          0          0          0
nomem      cantconn   interrupts
0          0          0
Connectionless
calls      badcalls   retrans    badxids    timeouts   newcreds   badverfs
0          0          0          0          0          0          0
timers     nomem      cantsend
0          0          0

Client nfs:
calls      badcalls   clgets     cltoomany
0          0          0          0
Version 2: (0 calls)
null       getattr    setattr    root       lookup     readlink   read
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
wrcache    write      create     remove     rename     link       symlink
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
mkdir      rmdir      readdir    statfs
0 0%       0 0%       0 0%       0 0%
Version 3: (0 calls)
null       getattr    setattr    lookup     access     readlink   read
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
write      create     mkdir      symlink    mknod      remove     rmdir
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
rename     link       readdir    readdir+   fsstat     fsinfo     pathconf
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
commit
0 0%
Version 4: (0 calls)
null       getattr    setattr    lookup     access     readlink   read
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
write      create     mkdir      symlink    mknod      remove     rmdir
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
rename     link       readdir    statfs     finfo      commit     open
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
confirm    downgrade  close      lock       locku      lockt      setclid
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
renew      clid_cfm   secinfo    release_lo replicate
0 0%       0 0%       0 0%       0 0%       0 0%
```

Figure 33. Displaying NFS client rpc and NFS statistical information

When using the **nfsstat** command, users should be aware that results are cumulative. These statistics may be reset with the *-z* option of the nfsstat command. Also, the nfsstat-*c* command provides statistics for all NFS client activity, which makes it possible to access files on more than one NFS server. When using this command to query the NFS client statistics for z/OS NFS, make sure that all NFS client access is for z/OS NFS only.

The nfsstat-*c* command provides two types of statistics, *client rpc* and *client nfs*. The *client rpc* statistics provide an indication of NFS performance from the perspective of that particular NFS client. In general, if the *timeout* value is more than 0.2% of the total number of *rpc* calls, then some performance tuning is necessary.

If the *timeout* value is essentially equivalent to the *retrans* value, the NFS client is waiting on the NFS server. While you can increase the *timeout* option of the **mount** command to reduce the number of retransmissions, this will not improve the perceived responsiveness of the NFS server.

The *badcall* and *badxid* statistics are generated when information is lost or dropped somewhere between the NFS client and the NFS server. This can happen when processes are interrupted and are not necessarily indicative of a performance problem unless they are disproportionately high or persist. On the NFS client system the **netstat -s** command provides additional statistics that may be useful in tuning at the network level. For the z/OS NFS, the z/OS TCP/IP netstat command provides information that may be useful for tuning as well.

The client NFS statistics provide a distribution of the NFS procedure calls made by users and applications on that particular NFS client. For a typical NFS client you will probably see a larger percentage of *getattr* and *lookup* calls. These calls are made whenever an NFS file is initially accessed. Directory listing is a common user activity that will generate these NFS calls.

The write and read statistics can also provide insight into the manner of NFS access on an NFS client, such as read or write biases. If the percentage of read and write NFS calls is higher than the percentage of getattr and lookup NFS calls, the client is probably accessing relatively large files as opposed to accessing many smaller files. Conversely, if the percentage of getattr and lookup NFS calls is higher than the percentage of read and write NFS calls, you might want to investigate whether users are querying directories or accessing relatively small files. In the latter case you might further determine if such files should be stored on an NFS server or the NFS client system.

Chapter 14. Optimizing the NFS environment

This topic explores areas that may impact the overall performance within an NFS environment. The focus will be on various network and NFS client and server system parameters which may affect NFS performance. Specific product documentation should be consulted for additional detail.

Network performance tuning

Table 49 contains network performance tuning information.

Table 49. Network performance tuning symptom and action information

Symptom	Action
Data transfer rates significantly less than network capacity	**ACTION 1:** *Know your network topology* - "What is the NFS environment?" on page 260 briefly introduces the role of the network in the NFS environment. Clearly, transferring data over congested networks results in poor performance. Therefore, when a performance problem is encountered in the NFS environment, it is useful to determine whether or not the network is the source of the problem prior to investigating other alternatives. While there are products specifically designed to monitor and report on network activity, knowing your particular network topology may be sufficient for initial tuning. Figure 34 on page 266 shows a simple network topology with two 100 Mbps Fast Ethernet Rings connected to a Gbps Fast Ethernet backbone network. In this example, access from NFS client A, on a 100 Mbps Fast Ethernet Ring, to NFS server B, directly connected to the Gbps Fast Ethernet backbone, would be as limited by network bandwidth as access to NFS server A. On the other hand, access from NFS client B, directly connected to the Gbps Fast Ethernet backbone, would be limited by the load on the Gbps Fast Ethernet network and the capacity of NFS server B.

© Copyright IBM Corp. 1991, 2010

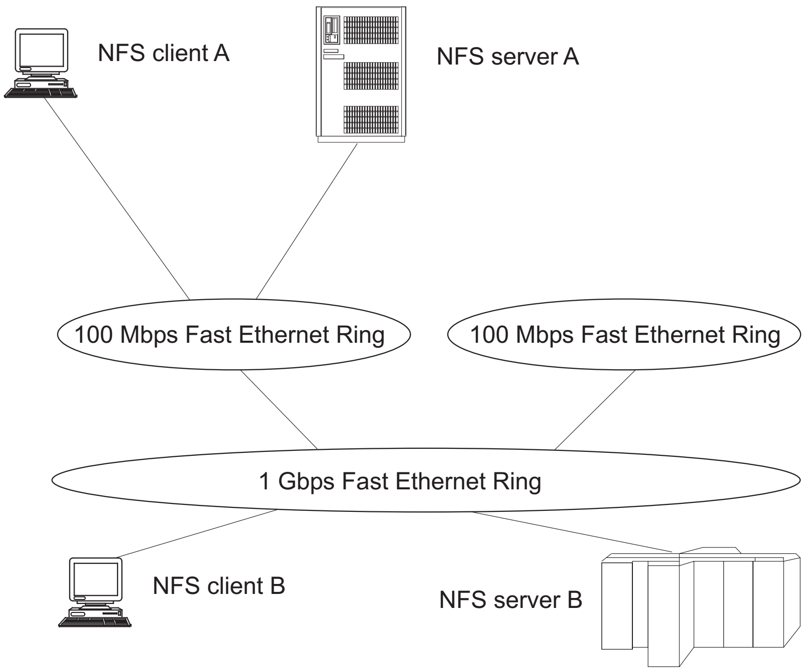

NFS client A
NFS server A
100 Mbps Fast Ethernet Ring
100 Mbps Fast Ethernet Ring
1 Gbps Fast Ethernet Ring
NFS client B
NFS server B

Figure 34. Sample network topology

There are other factors to consider in addition to the theoretical capacity of a network configuration. For instance, there are differences in the capability of network adaptors and network controllers. Overall network performance can also be influenced by the devices connecting network segments or subnets, particularly in terms of packet size limits and UDP checksum processing. While it may initially appear that there is no network constraint, there may be a device, adaptor, or network segment that is not performing at expected levels.

With this type of investigation, you may already have discovered that your primary constraint is a congested network. If so, you may elect to pursue such alternatives as:
- Increasing network bandwidth
- Changing network topology
- Scheduling applications during low system usage
- Modifying applications to reduce data transfer

ACTION 2: *Monitor network activity* - It may be necessary to monitor network activity over a period of time to determine what is causing a performance problem. In addition to whatever monitoring your network administrator may be able to provide, the **netstat -s** command, available on many NFS client platforms, may provide some insight as well. This command reports such statistics as bad checksums, dropped fragments, non-forwardable packets, various time-outs, and socket buffer overflows. In some cases, it may be necessary to use a network analyzer to determine exactly what is being sent over a network segment. While it can be difficult to analyze congested networks, network analyzers occasionally capture evidence that may indicate a problem with the NFS client or server system.

NFS client system performance tuning

Table 50 contains NFS client system performance tuning symptom and action information.

Table 50. NFS client system performance tuning symptom and action information

Symptom	Action
Excessive NFS retransmissions	A high number of NFS retransmissions can be detected with the **nfsstat -c** command. See section "Impact of the NFS protocol on performance" on page 262 for more information. A general rule of thumb is that the number of retransmissions should not be more than 2% of the total calls. Use **traceroute** *hostname* command to trace the route that IP packets take to a network host. This helps isolate where the bottleneck may occur.

Symptom	Action
	ACTION 1: *Increase network queue length* - If the **nfsstat -c** command shows evidence of bad calls in addition to timeouts, or the **netstat -s** indicates that packets or fragments are being dropped, increasing the network queue length may be advisable. Occasionally, a network analyzer will detect that the server has responded to an NFS request before the client retransmits the request. This indicates that the network queue on the NFS client system was full and the response was dropped. **ACTION 2:** *Increase maximum transmission unit* - The **nfsstat -in** command will display the maximum transmission unit (MTU) for each network interface active on the NFS client system. The MTU can be reset with the **ifconfig** command. This command is available on most UNIX NFS clients. For the z/OS NFS client, MTU can be reset in the TCP/IP profile. Increasing the MTU can improve overall performance and alleviate excessive IP fragmentation. However, with heterogeneous networks the MTU should not be set larger than the smallest acceptable MTU over the entire network route. For instance, when the network consists of a mix of Token Rings and Ethernets, the MTU is typically set at 1492 that is less than the 1500 MTU for an Ethernet and a 2000 MTU for Token Ring. Changing the network topology may allow you to increase the NFS client system MTU. **ACTION 3:** *Change socket buffer size* - Changing the buffer size is another way to tune the NFS client system. The AIX/6000 **no -a command** will display the current settings, in particular the **udp_sendspace** and **udp_recvspace**. The AIX/6000 **no -o udp_sendspace=32768** and **no -o udp_recvspace=32768** commands can be used to reset the **udp_sendspace** and **udp_recvspace**, respectively. For the z/OS NFS client, the **udpsendbfrsize** and **udprecvbfrsize** in the TCP/IP profile should be set respectively. The recommended value for Sun NFS V3 protocol is 65536. When processing with TCP protocol, it may be necessary to modify the **tcpsendbfrsize** and **tcprecvbfrsize**. **ACTION 4:** *Increase number of BIODs* - If data on an NFS server is typically accessed sequentially, or if there are many users on an NFS client system, it may be advantageous to increase the number of block I/O daemons (BIODs). This increases the NFS client processor utilization and should be weighed against other processor requirements on a multiple user system. If access is typically at random offsets within a file or file sizes are very small, you may even consider decreasing the number of BIODs. It should also be noted that some NFS client implementations honor BIODs differently to provide read or write bias. **ACTION 5:** *Increase timeout parameters* - If you have determined that NFS retransmissions are not caused by dropped NFS responses, you may consider increasing the timeout value. *This should be one of the last alternatives you select since this can have adverse effects.* If information is lost or dropped between the NFS client and the NFS server, increasing the timeout value can make performance worse. In this case the client would wait longer to decide to resend a request. In a heavily used NFS environment, increasing the timeout value also increases the likelihood that internal client or server buffers will become stagnant or stolen to service other network requests. You would probably only want to attempt this if you have strong evidence, probably from a network analyzer, that the NFS server is sending responses *after* the NFS client retransmits requests. The timeout values can be reset with the **mount** command.

NFS server system performance tuning

Regardless of whatever network and NFS client system tuning is done, some level of tuning or resource balancing should be addressed on the z/OS NFS server system. Typically, these systems have tuning and capacity planning procedures in place along with a high level of experience. Consequently, this section will be limited to a surface level discussion of the resource components to be considered when tuning the NFS server system.

z/OS constraints

Table 51 contains z/OS constraints information. These actions should be weighed against any increase in z/OS storage they may require.

Table 51. z/OS constraints symptom and action information

Symptom	Action
High CPU utilization with low aggregate throughput	**ACTION 1:** *Increase z/OS TCP/IP send and receive buffers* - There are other applications besides z/OS NFS which are dependent on the z/OS TCP/IP application. Some applications have different tuning requirements. The default values for **udpsendbfrsize** (or **tcpsendbfrsize**) and **udp udprcvbrfsize** (or **tcprcvbrfsize**) is 16 KB. It may be useful to increase these parameters up to 64 KB for better throughput with large files. **ACTION 2:** *Increase z/OS TCP/IP UDP queue* - Another area to consider is the z/OS TCP/IP UDP queue. As discussed in section "NFS client system performance tuning" on page 267, NFS responses are dropped when network adaptor queues are full. A similar situation occurs when the UDP queue of z/OS TCP/IP is full; in this case incoming NFS requests are dropped. The **noudpqueuelimit** keyword in the **assortedparms** section of the TCP/IP profile data set can be specified to enable the z/OS TCP/IP server to accept incoming UDP datagrams. Without specifying this keyword, the default queue length of 30 may not be sufficient. **ACTION 3:** *Modify NFS and TCP/IP dispatching priorities* - It is recommended that the z/OS NFS procedure have a relative dispatching priority less than that of TCP/IP. This is important because the MVS mean time to wait dispatching priorities are adjusted based on increased I/O activity. Since TCP/IP has higher network I/O than z/OS NFS, the TCP/IP dispatching priority is lowered. Assigning fixed dispatching priorities, with TCP/IP dispatched at a higher relative value than the z/OS NFS, can ensure this situation. **ACTION 4:** *Select transport protocols* - z/OS NFS supports TCP and UDP as transport protocols for the server, for both NFS Version 2 and Version 3 protocols. For the Version 4 protocol, NFS supports only TCP as a transport protocol. UDP is primarily used for its efficiency on high bandwidth, low latency networks, such as LANS. TCP is used for its efficiency on low bandwidth, high latency networks, such as WANS.
Constrained throughput with low CPU utilization	**ACTION 1:** - See "Subtasking" on page 284.

Table 51. z/OS constraints symptom and action information (continued)

Symptom	Action
High DASD utilization and NFS command response time	**ACTION 1:** *Evaluate placement of z/OS catalog data sets* - The performance of z/OS NFS will be impacted when serving files located on heavily utilized storage devices or devices behind congested storage controllers. Before deciding whether or not to move data or upgrade storage systems, make sure system catalogs are not on heavily utilized devices. Many of the NFS commands that are executed for z/OS conventional MVS data sets involve catalog access. In fact, listing directories, one of the most commonly executed commands on NFS client systems, accounts for a significant portion of the NFS **get** attribute and **lookup** commands. Such commands not only cause the z/OS catalog to be accessed but might also cause the entire file to be read depending on: • z/OS NFS processing options • Prior access by z/OS NFS • Whether or not files are DFSMS-managed **ACTION 2:** *DFSMS-managed data accessed from the network* - Files that are primarily accessed by way of z/OS NFS should be DFSMS-managed for improved performance. The z/OS NFS maintains file attributes for DFSMS-managed data sets when the z/OS NFS **nofastfilesize** processing option is specified. The **nofastfilesize** processing option provides exact file size determination rather than approximating file size as with the **fastfilesize** processing option. DFSMS management also provides improvements in terms of better storage utilization and specification of service levels. **ACTION 3:** *Evaluate placement of data accessed from the network* - While deciding whether or not to access DFSMS-managed files using z/OS NFS, consider also the placement of such files. Files with critical performance requirements should be placed on the appropriate devices by using the storage class parameters. If reallocation is necessary, you might also consider allocating sequential access method striped data sets for larger files or a z/OS UNIX file for smaller files. If you elect to maintain data set organization, you may choose to reblock the data set to a more optimal block size, such as half track blocking. You may even determine for some files that the best alternative is to store the files locally on the NFS client system.

Chapter 15. Evaluating z/OS NFS performance

This topic assists both NFS client users and NFS server system administrators in evaluating the performance of z/OS NFS. Test methods are provided to assist in collecting performance data for evaluation. The appropriateness and usage of these test cases and methods depend on the following conditions:

- Workload requirements to be evaluated
- Level of expertise of the evaluator
- Monitoring tools available

Evaluating throughput

Throughput refers to the rate at which data is transferred between the NFS client and server systems. Establishing throughput baselines for the Client-Server environments of interest will assist in determining any benefits achieved from tuning. Baselines and measurement techniques are also useful in diagnosing performance problems.

As discussed in "How is performance characterized?" on page 259, a set of work activities or workloads should be identified for measurement. For throughput the type of workload selected should reflect the type of NFS read and write access expected by typical users or specific applications. Consider the following points:

- Average file size or range of file sizes
- Sequential or random access
- Data set types and organizations to be accessed
- Requirements for ASCII/EBCDIC translation

Single process throughput

The easiest place to begin evaluating NFS read and write throughput is on a single client and a single process basis. While measuring in a controlled and isolated laboratory situation is advantageous, it may not be practical. Measuring the actual environment of interest can provide more meaningful information as well as establishing a methodology for monitoring performance in the future. Under such uncontrollable situations, it is important to determine range and normal variation in measurements with particular attention to peak and non-peak periods of activity.

Once the NFS environment has been selected for measurement, some simple techniques can be used to initially evaluate NFS read and write throughput. One of the easiest methods is to use commands available on the NFS client system, such as **copy** or **cp**, to generate NFS reads and writes. Copy commands along with **date**, **time**, or **timex** commands can be used to determine the elapsed time for the copy. Throughput can be readily calculated from the elapsed time and the size of the file.

Some additional considerations when measuring NFS throughput are the effects of graphical user interfaces (GUI), any overhead associated with opening files, and I/O to and from local physical storage. Measuring with and without a GUI will show the effect on NFS throughput caused by the GUI. Using a network analyzer can assist with isolating the time spent opening files from the time spent executing NFS reads or writes. Techniques such as copying to /dev/null can eliminate local physical I/O for NFS reads. For more complex requirements such as reading and

© Copyright IBM Corp. 1991, 2010

writing at random offsets, you may use an existing application to provide such access or write a program to execute NFS procedure calls at random offsets within a file. Program generated NFS procedure calls can also be useful when available local physical storage is insufficient for the file sizes to be measured.

Multiple process throughput

For NFS client systems with multiple users or applications you may want to determine aggregate throughput on a multiple process basis. If throughput on a single process basis has achieved the network capacity, then multiple process throughput will be limited by the same network capacity as well as any NFS client system limitations. A simple method for evaluating multiple process throughput is to simply execute multiple single process measurements simultaneously.

Multiple client throughput

Evaluating multiple client throughput requires the ability to propagate the measurement workload over more than one NFS client system, preferably the same type of system. Such measurements are more complex in that they require controlled execution and collection of results from multiple systems. Depending on your network configuration, remotely executing your single process workload on multiple systems is a relatively simple method to use to initially evaluate multiple client aggregate throughput.

Evaluating NFS command response time

A user is probably more interested in the response time for typical user commands, such as listing a directory, making or removing a directory, and creating or removing a file. However, when such commands are executed for NFS mounted file systems, the NFS command response times become a factor in user command response time. The transparent nature of the NFS protocol allows a user to define a set of commands as a measurement workload with a minimum amount of effort.

For those desiring more detail on NFS command response time, there are public domain programs as well as industry standard benchmarks. Both of these options require more experience on the part of the user and may have limitations on their usage. Such programs are typically written for the UNIX environment and may not support all NFS client and server systems.

Evaluating CPU utilization

Evaluating CPU utilization on an MVS system can be quite complex, particularly for a production system. The activity on such systems is already generally monitored and z/OS NFS is simply one more application to be monitored. High level monitoring can be accomplished from the Display Active (DA) display of the System Display and Search Facility (SDSF). This display provides an overview of the total activity on the system. Of particular importance are the CPU usage and relative dispatching priorities of TCP/IP and z/OS NFS. Monitoring tools such as the Resource Measurement Facility (RMF™) Monitor I and Monitor II can provide more detail on the performance characteristics of the MVS system.

Collecting server usage data

You can use the SMF records that z/OS NFS produces to keep track of how files are accessed and how long each NFS user session lasts. The following SMF records can be produced:

Record type 42 subtype 7	This record, written when a file times out, provides file usage statistics.
Record type 42 subtype 8	This record, written when a client user logs out of the z/OS NFS, provides user session statistics.
Record type 42 subtype 26	This record, written when a client creates, removes, or renames the file objects on the NFS mounted file system, provides the NFS Client's information

Chapter 16. Tuning the z/OS NFS server

Previous topics discussed performance tuning from a total system perspective. This topic discusses performance tuning at the z/OS NFS server level. In particular, this topic focuses on the z/OS NFS data set creation, processing, and site attributes.

Data set creation attributes

When creating new MVS data sets, the allocation parameters are the most readily accessible means for the end user to influence the performance of z/OS NFS server. They are also not well known outside of the z/OS user community. This is primarily due to the fact that MVS has a record oriented data structure and most NFS client systems have a byte oriented data structure. This section looks at how a variety of allocation parameters affect performance.

Block size and record length

Block size (BLKSIZE) specifies the maximum length, in bytes, of a physical block of storage in MVS. If BLKSIZE(0) is specified, the system will determine the optimal block size based on the maximum record length (LRECL) and the physical characteristics of the disk, or approximately half of a physical track. Half track blocking is optimal in that it reduces the amount of wasted storage on the disk.

The BLKSIZE is also important to the performance of the z/OS NFS, which performs physical I/O on a block basis. While half track blocking is generally optimal, for z/OS NFS server you should be aware of the relationship between the block size, the file record length, the network packet size and the NFS client **mount** write and read buffer sizes, *wsize* and *rsize*.

A network packet size will be negotiated between the NFS client and the NFS server when the NFS **mount** is processed. In general, the NFS server will determine the packet size for write operations, and the NFS client will determine the packet size for read operations. For z/OS NFS, this packet size is 8192 bytes, when processing with Sun NFS V2 protocol. For several NFS client implementations the default packet size is 8192. However, there are some implementations with a 4096 byte maximum packet size. The maximum packet size is 65536 bytes when processing with Sun NFS V3 protocol, and larger with the V4 protocol. However, z/OS NFS supports a maximum packet size of 32768 bytes for both the NFS V3 and V4 protocols.

The packet size can also be automatically reset to a smaller value when a constraint is detected. Such a reduction in packet size causes an increase in the number of NFS requests for the server to process the same amount of data. This increases the load on the server and may or may not have been the initial constraint causing the packet size reduction. This can have an adverse effect on z/OS NFS server performance, particularly when processing write requests.

Regardless of the packet size, the z/OS NFS server must reassemble the packets, possibly arriving out of sequence, into records. The z/OS NFS server buffers (see "Ordering out-of-sequence data" on page 282) and orders these packets until a block of data can be written to disk. Write requests, especially those for large files, are processed more efficiently when the likelihood of packets spanning records and blocks is reduced.

© Copyright IBM Corp. 1991, 2010

When the z/OS NFS server is processing read requests, the physical I/O is again on a block basis. Consequently, read throughput is improved by increasing the block size up to half track blocking.

If files are allocated with a default block size (BLKSIZE=0) on RAMAC 3 or Enterprise Storage Server (ESS™) 2105 DASD with 3390-3 format, the actual block sizes will probably vary between 24 and 27 KB, depending on the specified record length and record format. Table 52 shows the actual allocation block sizes for physical sequential data sets allocated on a RAMAC 3 or ESS 2105 DASD with a variety of record lengths and with both fixed and variable length records.

Table 52. Default block sizes for RAMAC 3 or ESS 2105 DASD

Record Format	Record Length (bytes)	Block Size (bytes)
FB	80	27,920
FB	1024	27,648
FB	4096	24,576
FB	8192	24,576
VB	84	27,998
VB	1028	27,648
VB	4100	27,998
VB	8196	27,998

Record format

Record format (RECFM) specifies the format and characteristics of the records in the data set. This section will be limited to a discussion of the performance considerations associated with fixed and variable length records.

There seems to be a natural affinity toward allocating MVS data sets with variable length records from the perspective of a byte oriented NFS client operating system. Such allocations enhance the feeling of transparent access and eliminate requirements to determine a fixed record length. However, the placement of incoming packets into records of varying lengths is more complex than placing packets into records with fixed lengths. Consequently, write performance for z/OS NFS is generally more efficient when MVS data sets are allocated with fixed length records. Record formats are specified with the RECFM data set creation attribute.

Data set organization and data set type

Data set organization (DSORG) specifies the organization of an MVS data set. The z/OS NFS data set organization attribute can be physical sequential (ps), direct access (da), or VSAM. z/OS NFS server supports three types of VSAM files: key-sequenced (KSDS), entry-sequenced (ESDS), and relative record (RRDS). However, keyed access and relative-number access are not supported by the NFS protocol.

The DSORG attribute is ignored for directory-oriented NFS client commands. The data set type (DSNTYPE) specifies whether a PDSE or PDS is to be created when the z/OS NFS receives a **mkdir** command from the NFS client. A PDSE is created when library is specified, and a PDS is created when PDS is specified. You cannot create another PDS (or PDSE) within a PDS (or PDSE).

Another MVS data set type is extended (EXT), which identifies an extended format data set. With z/OS NFS, these DFSMS-managed data sets are allocated based on specification of a data class (DATACLAS). An appropriate data class must have been defined on the MVS system for an extended format data set to be allocated. One of the values specified when creating an MVS data class is the volume count which determines the number of stripes allocated for an extended format data set. If automatic class selection (ACS) routines have been written to allocate extended format data sets based on such criteria as naming conventions, it would not be necessary to specify a DATACLAS to allocate an extended format data set through the z/OS NFS server.

The hierarchical file system (HFS) and z File System (zFS) are other data set types. They must be DFSMS-managed and be mounted within the z/OS UNIX subsystem. While an HFS or zFS file cannot be allocated using z/OS NFS, they can be accessed through the z/OS NFS server.

Data set organizations and data set types are generally selected based on user and application requirements. However, performance should be considered as one of these requirements. An important factor to consider when evaluating performance is the size of the files to be accessed. For smaller file sizes, we can differentiate between the overhead costs associated with creating or opening a file for output. As the file size increases, this impact becomes less of a factor in the transfer rates.

B37/D37/E37 ABEND handling

The z/OS NFS server has no control over available space for a data set on disk and cannot automatically extend to the available space on the disk. During writing to disk from the z/OS server, x37 Abends can occur more readily when data sets are created with small SPACE primary and secondary allocation units. For the z/OS server to avoid possible x37 Abends, data sets should be allocated with primary and secondary SPACE allocation units to fit the largest byte data stream from an NFS client. Due to asynchronous write between the z/OS NFS server and the NFS client, the NFS client may not get a no-space-on-disk error as a result of an x37 Abend on the z/OS server side if the z/OS server has to flush cached data and close/deallocate a data set at a time other than a write or commit request, such as after writetimeout expiration. For small data set allocations, there is a risk to lose data and get a no-space-on-disk error postponed due to z/OS server data caching.

NFS server cache monitoring and reporting

The NFS Server issues a message to indicate an MVS data buffer shortage each time the server's threshold limit is reached, and when the threshold is overcome. This message implies that some of the current requests may fail. This threshold is governed by the server site attributes that specify the storage limit (in bytes) for the buffers (**bufhigh** attribute) and a percentage of **bufhigh** storage limit before BUFHIGH watermark is reached (the **percentsteal** attribute). See “Buffer usage and caching” on page 281 for more information.

Processing attributes

Processing attributes are used to control how files are accessed from the NFS client system. Default values can be specified when the server is started. These defaulted values can be overridden at the NFS **mount** level or at the individual command level. The processing attributes, like the data set creation attributes, can affect the performance of the Network File System (NFS). This section will be limited to those processing attributes which have the most influence on performance.

Character translation

The processing attributes that specify whether or not data conversion occurs between ASCII and EBCDIC formats may have the most influence on the performance of z/OS NFS. The binary attribute, which specifies no data conversion, is the most efficient manner in which to access data by way of z/OS NFS. However, data conversion may be a requirement, particularly when data is shared with z/OS users or applications. Under such circumstances the text processing attribute would be specified instead of the binary processing attribute. The actual impact associated with data conversion will depend on the data to be converted.

In text processing mode for MVS data sets, end-of-line (EOL) terminators are inserted into the data at the end of each record to preserve the record format when data is viewed from the z/OS NFS client. However, text mode processing invokes an implicit ASCII to EBCDIC translation, which is not needed for a z/OS to z/OS mount. If, during text mode processing, this implicit data conversion (ASCII to or from EBCDIC translation with UNICODE and XLAT) is disabled using the srv_ccsid(1047) and cln_ccsid(1047) processing attributes on the mount command then the EOL terminators are converted (including EOL terminators such as LF, CR, NL) to an internal representation of the MVS data on the z/OS NFS server. The z/OS NFS server will use the EOL internal representation as record boundaries when writing to a data set. During reading of data from an MVS data set, the insertion of EOL terminators (in the internal representation on the z/OS NFS server) is used to create record boundaries. The data with EOL terminators are then converted to the external representation before replying to a client.

File size determination

The attributes discussed previously have primarily affected data transfer rates. Whether or not to determine the exact file size in bytes has more of an effect on user command response time. This is because the file size in bytes is one of the file attributes obtained by an NFS **Get Attribute** procedure. The NFS **Get Attribute** procedure is executed for every user command accessing an NFS mounted file system.

z/OS NFS provides processing attributes to specify estimated file size determination, fastfilesize, or exact file size determination, nofastfilesize. The accuracy of the file size with fastfilesize processing is best for binary processing of data sets with fixed length records. For other situations, or when exact file size is a requirement, the **nofastfilesize** processing attribute should be specified.

When the **nofastfilesize** processing attribute is specified, there are performance factors to consider. The most important factor is that file size in bytes is maintained by z/OS NFS for DFSMS-managed data sets with many data set organizations.

Figure 35 on page 279 compares the differences in response time between the z/OS NFS attribute support for DFSMS-managed data sets and the case when attributes are not cached for non-managed data sets. In this example, all data sets were members of a PDSE data set allocated with a variable block record format. The ls -l command was executed with the z/OS NFS **nofastfilesize** and **binary** processing attributes.

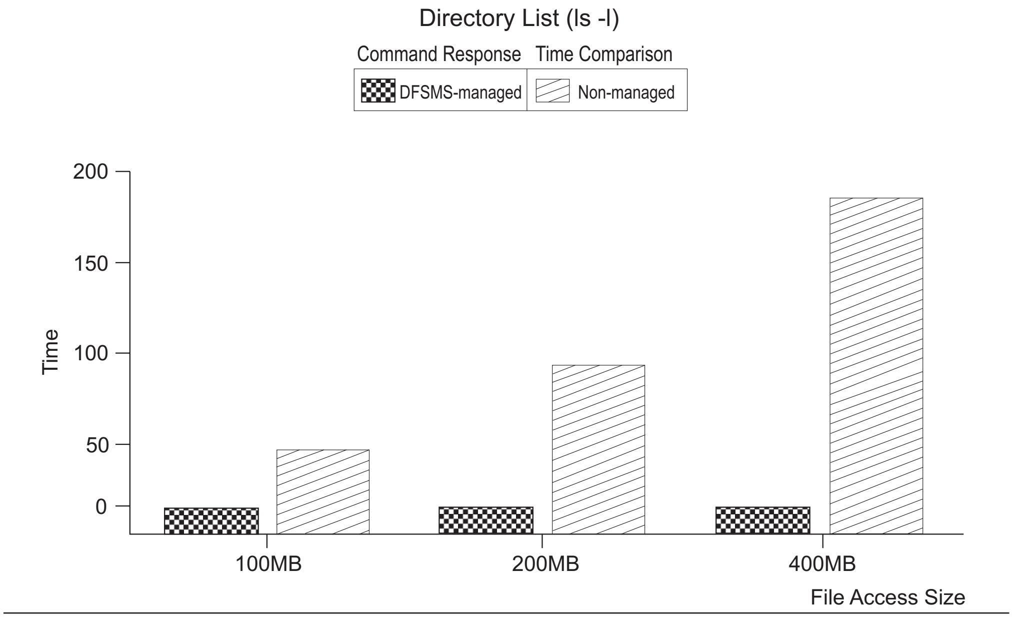


Figure 35. Directory list comparison between DFSMS-managed and non-managed

Data sets processed with the **nofastfilesize** attribute that are not DFSMS-managed will need to be read entirely to determine the exact file size in bytes when initially accessed by way of z/OS NFS. z/OS NFS will cache this information until the data set is modified (see section “Buffer usage and caching” on page 281) or the attribute time value has expired. The larger the file is, the greater the impact to command response time is. Figure 35 shows how the response time for a directory list command can be impacted when the file has to be read to determine the exact file size in bytes.

Maintaining the file size attribute by z/OS NFS also affects read throughput when accessing larger DFSMS-managed data sets with the **nofastfilesize** processing attribute. In the nofastfilesize case, reading the entire file to determine byte file size is not necessary. The impact on the end-to-end response time for the read is lessened by reducing the response time for the initial access to obtain file attributes.

Data set timeout specification

There are three timeout processing attributes that specify when z/OS NFS will release a data set following certain NFS operations. The **readtimeout** and **writetimeout** processing attributes specify how long z/OS NFS will wait after respective read and write operations before releasing a data set. The **attrtimeout** processing attribute specifies how long a data set will remain allocated after an NFS **Get Attribute** or **Lookup** operation. When accessing data sets that are not DFSMS-managed with the **nofastfilesize** processing attribute, it may be advantageous to increase the attrtimeout value so that z/OS NFS caches data set attributes for a longer period of time. However, all three processing timeout attributes must be within the range of the **maxtimeout** and **mintimeout** site attributes.

Note: When using the NFS version 4 protocol, these timeout values should be set to a value less than or equal to the lease time. Otherwise, it is possible for performance problems to occur when attempting to access MVS data sets.

Accessing migrated files

While migrating less frequently accessed data sets to tape improves MVS storage utilization, retrieving migrated data sets can impact the performance of z/OS NFS. z/OS NFS provides three processing attributes to specify how migrated data sets are to be handled by the server: **retrieve(wait)**, **retrieve(nowait)**, and **noretrieve**.

The **retrieve(wait)** attribute instructs the server to wait for a migrated data set to be recalled to a direct access storage device. The NFS client user or application process will not receive a response from z/OS NFS until the data set has been recalled. If the migrated file is relatively large, or migrated to tape, this can cause the NFS client to retransmit the request. The NFS client will retransmit the request a fixed number of times as specified by the **retrans** parameter of the **mount** command. If the recall of a migrated data set takes longer than the product of the timeo value on the **mount** command and one plus the **retrans** value for the **mount**, **timeo * (retrans +1)**, the initial command will most likely need to be executed again. Using the default values for the AIX/6000 **mount** command of seven-tenths of a second for **timeo** and three attempts for **retrans**, a recall would need to take less than 3 seconds. An alternative approach is to use the **retrieve(nowait)** processing attribute.

The **retrieve(nowait)** attribute instructs the server to recall a migrated data set and immediately return a "device not available" message to the client without waiting for the recall to complete. With this option, users attempting to access the file can continue to use their session for other activity rather than waiting an indeterminate time for the recall of a file. They can attempt to access the file again after allowing some period of time for the recall to have completed.

If it is critical to the user that a file be recalled before further work can continue, the **retrieve(wait)** processing attribute can be specified more selectively as part of the NFS client user command syntax along with the file name. While doing this provides the user more control over command execution, the user command may still timeout and have to be executed again.

Specifying the **retrieve(wait)** processing attribute as the system default may also impact the availability of z/OS NFS subtasks to process such requests. The last parameter of the **nfstasks** site attribute determines the number of z/OS NFS subtasks available to process such recall operations. See “Subtasking” on page 284 for more information about the **nfstasks** site attribute.

Another method of avoiding the unnecessary recall of migrated data sets is to specify the **noretrieve** processing attribute. With the **noretrieve** attribute, the server does not recall a migrated data set and a "device not available" message is returned to the client. This processing attribute is particularly useful when listing a directory with unknown contents.

Asynchronous z/OS UNIX processing

z/OS NFS with Sun NFS V2 protocol provides two processing attributes that only apply to z/OS UNIX processing. The **sync** attribute specifies that z/OS UNIX write requests should be committed to nonvolatile media (for instance, DASD) when received by the server. Some performance improvement can be obtained with the alternative **async** processing attribute. This type of processing causes z/OS UNIX write requests to be cached at a level within the z/OS UNIX architecture prior to the physical I/O. For more information about z/OS UNIX tuning, see “Ordering out-of-sequence data” on page 282 and “Subtasking” on page 284.

The **async** and **sync** attributes specified in the control file are ignored when z/OS NFS server is communicating with Sun NFS V3 and V4 protocols. The processing attribute is determined in the client application implementation. Asynchronous write is also recommended for faster throughput.

Site attributes

The site attributes control z/OS NFS resources. Tuning of these parameters should be done with caution. They are only specified at the start of the z/OS NFS server and cannot be modified by client users. This section is limited to those attributes which more directly affect performance.

Buffer usage and caching

The **bufhigh** site attribute specifies the storage limits (in bytes) of allocated buffers (see Figure 36). When the **bufhigh** storage limit has been reached, a percentage of the buffers will be reclaimed, and the amount of reclamation is determined by the **percentsteal** attribute. z/OS NFS uses this buffer area to cache file information, thereby satisfying requests more efficiently.

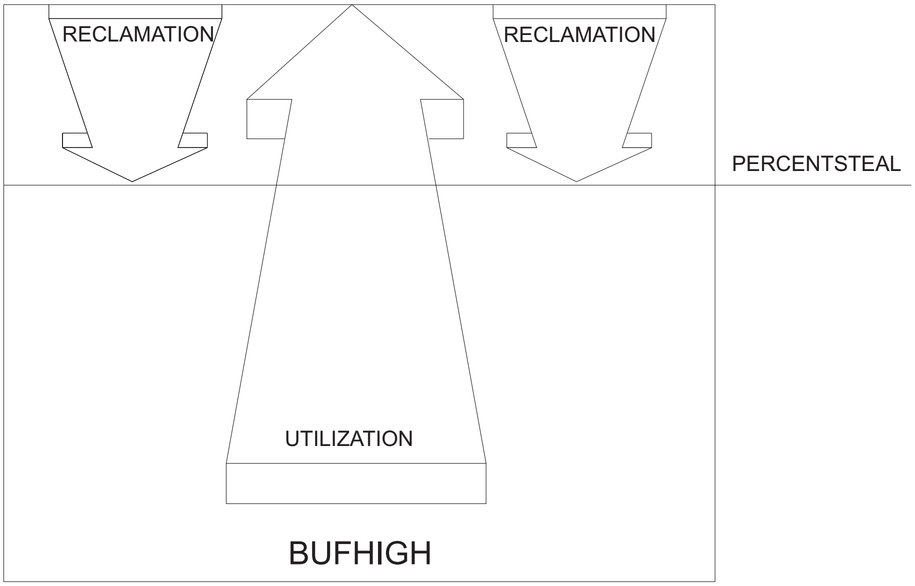


Figure 36. Bufhigh utilization with percentsteal

The NFS Server issues the message GFSA363I to indicate an MVS data buffer shortage each time the server's threshold limit is reached, and when the threshold is overcome. This message implies that some of the current requests may fail with ENOMEMORY error. This threshold is governed by two values set by the bufhigh server site attributes: the storage limit (in bytes, Kbytes, or Mbytes) of data buffers and the water mark (in percent) of the storage limit. There are additional console messages generated to pre-warn of the impending buffer shortage, before the critical threshold is actually reached. The benefit of these warning messages is that System Automation can be configured to act upon these 'Buffer Shortage' warning messages to inform the application administrators on the remote NFS client to react accordingly and possibly throttle the incoming workload.

There are two additional site attributes, **readaheadmax** and **maxrdforszleft**, that also affect buffer usage. The **readaheadmax** attribute specifies the number of bytes to be read to fill the internal buffers so that additional read requests for that file may be satisfied directly from cache. The **maxrdforszleft** site attribute defines the number of physical block buffers to cache after determining a file's size (see section “File size determination” on page 278). This information is cached to satisfy any subsequent read requests for the same file.

Keep in mind also that some information will be cached on a file handle basis, in other words, for every file accessed within the timeout periods. Tuning of the **bufhigh**, **readaheadmax**, and **percentsteal** values should be determined based on the following conditions:

- Number of files to be accessed
- Available region
- Amount of data to cache per file

For example, suppose that, on average, 1,000 physical sequential (DSORG=PS) files allocated on 3390 Model 3 DASD are accessed within the same timeout period through the z/OS NFS server. Further, assume that we would like two blocks of a file cached internally to satisfy read requests. If we assume an average block size of 25 KB (see “NFS client system performance tuning” on page 267 for more detail), we would need at least 50 MB of storage for internal buffers. With a 20 percent value for **percentsteal**, a 64 MB value for **bufhigh** would probably be more reasonable. For this example, a reasonable value for the **readaheadmax** attribute is the file block size, or 25 KB. The **readaheadmax** value should not be less than twice the maximum record length of files accessed by way of the z/OS NFS server.

Ordering out-of-sequence data

While the previous section primarily addressed satisfying read requests from cache, this section discusses the caching used by z/OS NFS to satisfy write requests. z/OS NFS provides two site attributes for this purpose: **cachewindow** and **logicalcache**. As with all site attributes, **cachewindow** and **logicalcache** can be modified only at server startup.

The **cachewindow** attribute specifies the window size, in terms of number of packets, used in logical I/O to buffer client block writes received out of sequence. Figure 37 on page 283 shows how out of order packets are buffered in a cache window until a block of data can be physically written to storage. Since out of sequence packets are an inevitability, a certain amount of buffering is necessary.

In a congested network environment, the nature of the underlying protocols should be taken into consideration when tuning the **cachewindow** attribute. Under such circumstances, NFS, UDP, and IP requests are more likely to timeout. When this situation is detected within the underlying protocols, the packet size can be reduced automatically. The packet size is generally halved until the server and client systems are balanced. The minimum packet size that can be negotiated is 512 bytes. Accessing the z/OS NFS server in such an environment can result in the default 8 KB read and write buffers being reduced to 512 bytes, or 16 packets instead of one packet. Under these circumstances the recommended value for **cachewindow** is 16 times the number of BIODs.

Another factor to consider in tuning the **cachewindow** attribute relates to the relationship between packet size, data set record length, and block size. If the packet size is typically greater than or equal to the data set block size, there is less of a need to buffer the packets. On the other hand, if the packet size is smaller than the block size and/or record length, a larger **cachewindow** value will be required to buffer the packets until a block I/O is executed.

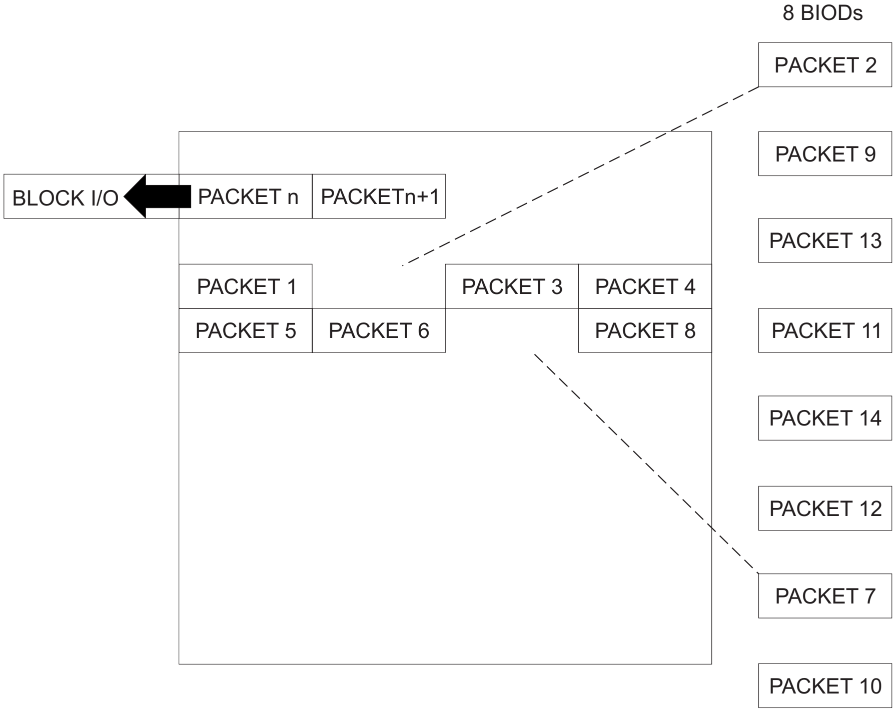


Figure 37. Relationship between cachewindow and BIODs

The **logicalcache** site attribute specifies the size (in bytes) of allocated buffers for all the cache windows combined (see Figure 38). The number of cache windows in use will depend on the number of file handles accessed within a given timeout period.

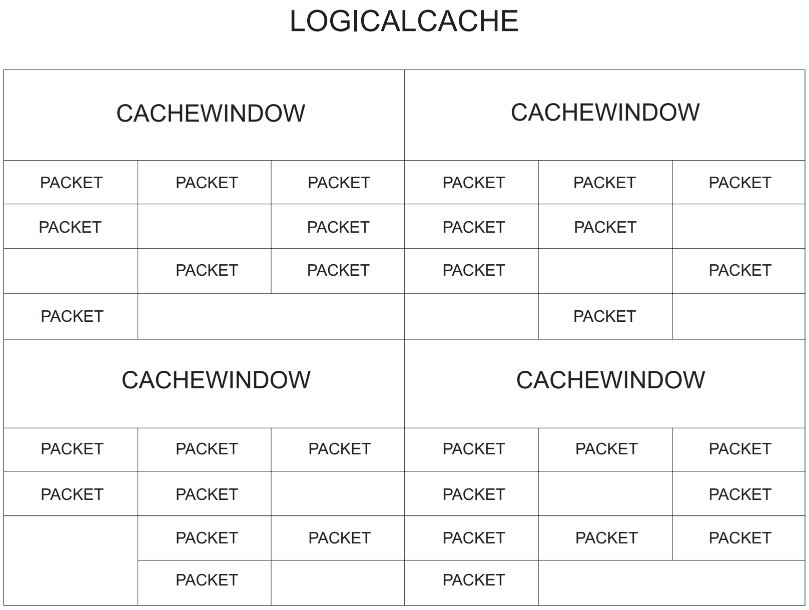


Figure 38. Logicalcache utilization for cachewindows

Tuning of the **logicalcache** and **cachewindow** values should be determined based on the following conditions:

- Number of files accessed for write
- Available region
- Typical packet size
- Data set block size
- Number of client Biods

The **cachewindow** and **logicalcache** attributes do not apply to z/OS UNIX processing.

For example, suppose you have AIX NFS client workstations on a lightly loaded network, with the default value of six Biods, writing to physical sequential MVS data sets allocated with 8 KB fixed length records. Further, assume that these files were allocated on 3390 Model 3 DASD with a 24 KB block size and that the network packet size is 8 KB. Then, a value of 12, twice the number of NFS client Biods, for **cachewindow** means that an out of sequence packet could be internally buffered within a 96 KB or four block range. If a packet is received outside this range, z/OS NFS must read data from storage in order to write the packet at the correct offset into the data set. z/OS NFS detects and ignores duplicate requests.

A value of 12 for **cachewindow** and a network packet size of 8 KB in our example results in a 96 KB buffer requirement per client actively writing to MVS data sets through z/OS NFS. A 5 MB value for **logicalcache** is sufficient to satisfy sustained write requests from 50 NFS client workstations at the same time.

Storage considerations

To avoid storage shortages, consider the following conditions:

- If a region size is specified, the **bufhigh + logicalcache** value should not be more than 90 percent of the region size. (The more **bufhigh**; the better performance.) With **bufhigh(4M)**, **logicalcache(1M)**, and the existing NFS core size, the minimum region size is 25M.
- If a region size is equal to 0, **bufhigh + logicalcache** value is limited by the supported virtual memory size on the installation. Region size can be set up as large as the size of the local page data sets.

The System Programmer should also consider if the other jobs on the installation use virtual memory.

Subtasking

The **nfstasks(n,m,o,t,u)** site attribute specifies the number of server processes to initiate on startup, where:

n is the number of processes started to handle asynchronous operations (z/OS MVS data set access) and short duration synchronous operations (SAF calls).

m is the number of processes which handle z/OS UNIX requests. Increase this value if your server requires many parallel z/OS UNIX accesses.

o is the number of processes that handle longer duration synchronous operations (concurrent NFS server recall requests.) Increase this value if your server supports lots of active recall operations.

t is the number of transport tasks dedicated to receiving TCP requests. If a large number of concurrent TCP requests are expected and the n,m values have already been increased to handle the expected request processing workload but the requests still seem to be taking excessively long to complete, then this value should be increased to relieve any possible bottleneck in the request receive processing.

u is the number transport tasks dedicated to receiving UDP requests. If a large number of concurrent UDP requests are expected and the n,m values have already been increased to handle the expected request processing workload but the requests still seem to be taking excessively long to complete then this value should be increased to relieve any possible bottleneck in the request receive processing.

If **nfstasks(n,m)** is specified, then the valid value range for *n* is from 4 to 99 and for *m* is from 4 to 100. The sum of *n* and *m* must be less than or equal to 100. If **nfstasks(n,m,o)** is specified, then the valid value range for *n* is less than or equal to 99 and for *m* is less than or equal to 100. The valid value range for *o* is less than or equal to 99. The sum of *m* plus *o* must be less than or equal to 100.

The absolute and relative value of *n* and *m* should be tuned for the expected system usage. If z/OS MVS data sets will be accessed, primarily, then *n* should be relatively high. If z/OS UNIX files will be accessed, primarily, then *m* should be relatively high. The absolute value of these will influence the amount of system resources consumed (higher values will make more system resources available to process NFS requests.

Based on system resources available below the 16 Mb line, the maximum *n* value may not be achievable. The precise maximum value will be system configuration dependent. If an 80A or 878 Abend is experienced during NFS server startup, use a smaller value for *n*.

When **nfstasks(n,m)** is specified, *m* represents both the processes for z/OS UNIX as well as the long duration synchronous operations. It is better to use the **nfstasks(n,m,o)** format to have better granularity of control and have the ability to specify a higher number of z/OS UNIX processes to handle more z/OS UNIX operations.

Based on system resources available below the 16 Mb line, the maximum *n* + *o* value may not be achievable. The precise maximum value will be system configuration dependent. If an 80A or 878 Abend is experienced during NFS server startup, use a smaller value for *n* + *o*.

Chapter 17. Tuning the z/OS NFS client

This topic discusses performance tuning at the z/OS NFS client level. Performance of the z/OS NFS client is affected by the following installation parameters that can be tuned (see Table 53). The default value of each parameter is specified in parentheses.

Table 53. Client installation parameters for tuning

attrcaching (Y)
biod (6)
bufhigh (32)
datacaching (Y)
delaywrite (16)
dynamicsizeadj (Y)
readahead (8)

Most of the installation parameters that are specified in member BPXPRM*xx* in SYS1.PARMLIB can be overridden for a mount point by using the **mount** command. The exceptions to this are **biod** and **bufhigh**, which require restarting z/OS UNIX.

Table 54 shows mount parameters that affect performance and that can be tuned. The default values are specified in parentheses.

Table 54. Mount parameters for performance and tuning

attrcaching (Y)
datacaching (Y)
delaywrite (16)
readahead (8)
rsize (8192 for V2) (No default for V3 or V4 - system determined)
wsize (8192 for V2) (No default for V3 or V4 - system determined)
vers (4)

Caching

Caching of file attributes (**attrcaching=y**) and data (**datacaching=y**) are recommended for performance. By caching the file data, all subsequent references to the cached data is done locally, avoiding the network overhead. Further authorization checking for subsequent access to the cached data or for other client users is done on the z/OS NFS client system. With attribute caching, z/OS NFS client will perform consistency checks to maintain valid cached data.

Note: attrcaching and datacaching are not supported for Kerberos mounts (for example, krb5, krb5i, or krb5p). They are only supported for system authentication (sys) mounts.

Dynamicsizeadj

The **dynamicsizeadj** parameter specifies the capability to have the packet size adjusted automatically when the remote procedure call (RPC) timeout has occurred.

© Copyright IBM Corp. 1991, 2010

For example, the **mount** command specifies, rsize=32K. If the RPC READ with a 32 K bytes request times out:

- The RPC request times out.
- The client cuts the size in half and sends the RPC READ with a 16 K request.
- The RPC request times out again.
- The client cuts the size in half and sends the RPC READ with a 8 K request.

Note: This process continues until the minimum size is reached.

Bufhigh

The **bufhigh** parameter specifies the maximum size in megabytes of allocated buffers for caching. z/OS NFS client uses this buffer area to cache file data, thereby satisfying requests efficiently.

When a client system is suffering high CPU utilization and low throughput, increase the **bufhigh** storage size. Increasing the **bufhigh** storage size by twice the default value (within available region) may improve system performance. Also, if the client system is processing mostly large files, it is good to increase the **bufhigh** storage size. Any changes to the **bufhigh** parameter require restarting z/OS UNIX.

Biod

The z/OS NFS client uses block I/O daemons (BIOD) to perform asynchronous read or write when **datacaching** is enabled. The **biod** parameter specifies the number of asynchronous block I/O daemons to use. If there are a lot of users on the NFS client system, it may be advantageous to increase the number of BIODs. The maximum number of BIODs should never exceed two times the number of client system processors. This will avoid excessive TCB switching. Tuning of this parameter should be done with caution. Any change to the **biod** parameter requires restarting z/OS UNIX.

Readahead

With caching enabled, read requests may be satisfied from cache. The **readahead** parameter specifies the maximum number of disk blocks to read ahead. Thus, additional read requests for the file may be satisfied directly from cache. When application is anticipating mostly sequential reads, increase the readahead to the maximum. If application is mostly processing random reads, use the default or a minimal readahead value.

Delaywrite

With caching enabled, write requests may be cached. The **delaywrite** parameter specifies the maximum number of disk blocks to cache. The z/OS NFS client will issue the WRITE RPC whenever the system load is low or if the cache data are in sequence. **delaywrite** increases WRITE performance by overlapping the disk I/O. This performance increase is most apparent with many small writes or with many random writes. When there is a high volume of write requests, set the **delaywrite** parameter to the maximum value(32). If there is a low volume of write requests, set the **delaywrite** to the default (16).

Vers

The **vers** parameter does not have to be specified unless you would like to override the default. The z/OS NFS client will communicate at the highest protocol level supported by the server, by default. Verify the protocol level for tuning of read or write buffer size (**wsize**, **rsize**). Use nfsstat −m /mountpoint command and look for **vers(n)** in the report or use rpcinfo −p servername command and look for "100003 4 TCP 2049 nfs" in the report.

Wsize and rsize

wsize and **rsize** specify the buffer size to use for read and write request. For NFS Version 2 protocol, **rsize** and **wsize** are a multiple of 512 bytes, up to a maximum of 8192 bytes.

For the NFS server with Version 3 (or later) protocol, **rsize** and **wsize** are negotiated between the z/OS NFS client and the NFS server. The maximum buffer size that is supported in z/OS NFS client system is 32 KB. The negotiated read and write buffer size will be the smaller of the buffer size supported by the server and 32 KB (min(server_info, 32KB)).

The z/OS NFS client system will reduce the read or write buffer size when there is constraint on the storage resource.

When using NFS Version 3 (or later) protocol the system normally determines the optimal buffer values. However, if the number of retransmissions or the number of time-out from **nfsstat** report is high, reduce the write buffer size (**wsize**) to 8 KB to avoid retransmission of large 32 KB data.

Part 4. Diagnosis and Messages

Chapter 18. Diagnosis and reporting of problems 293
Correcting input errors 293
Using keywords to identify a problem 294
  Component identification keyword 294
  Release level keyword 295
  Type-of-failure keyword 295
    ABENDxxx 295
    MSGGFShnnnt 295
    WAIT 296
    LOOP 296
    INCORROUT 296
    DOC SCnnnnnnnn 296
    PERFM 297
  Service level keyword 297
Using z/OS component tracing 297
  Component trace benefits 298
  Using NFS server component trace PARMLIB member CTINFS*nn* 299
    CTINFS00 member of SYS1.PARMLIB . . . 299
  Using NFS client component trace PARMLIB member CTINFC*nn* 300
    CTINFC00 member of SYS1.PARMLIB . . . 302
  Capturing NFS Server component trace information in an SVC dump 304
  Capturing NFS Client component trace information in an SVC dump 305
  Using a z/OS component trace external writer 305
    Step 1. Telling MVS to start a CT external writer 305
    Step 2. Telling NFS to connect to a CT external writer and start sending records . . 306
    Using IPCS to view records from an external writer 306
    Filtering NFS ctrace records in IPCS 307
Setting up a dump data set for abnormal ends . . 311
Searching the IBM database for APARs and PTFs 312
Contacting the IBM Support Center 312
Diagnostic aids 313
  First failure data capture 313
    Symptom data 313
    SVC dump 313
    Dump suppression 314
    Data capture suppression 314
    Invocation 314
  Errors and messages 314
  Debug trace data capture 315
  NFS client hang problem analysis 315
Environmental checklist 316

Chapter 19. Network File System messages . . 319
Server messages 319
Client messages 381
Client/Server Shared Messages 402
Messages from the client platform (AIX) 412

Chapter 20. Return codes 417

Chapter 21. Reason codes 427
Special reason codes (*xx* is 00-0F). 428
Reason codes from NFS Client or NFS Server modules (*xx* is 10-FF). 439
  USS JR*cccc* reason codes (0000-0FFF). 439
  Global reason codes (*yyyy* = 1000 - 3FFF) . . . 443
  Module specific reason codes (*yyyy* = 4000 - 4FFF) 447

© Copyright IBM Corp. 1991, 2010

Chapter 18. Diagnosis and reporting of problems

This topic is intended to help you diagnose NFS problems and use keywords to describe the NFS program failures.

Before you begin diagnostic procedures, verify that the suspected problem is not a result of failure caused by:

- z/OS Communication Server's TCP/IP (Transmission Control Protocol/Internet Protocol)
- Broken clients (for example, sending incorrect file handles or ignoring the server’s error return code)
- The network (for example, packets not arriving at the server, clients not receiving replies, or a gateway going down)
- The result of user input error.

For failures caused by z/OS Communication Server's TCP/IP, see *z/OS Communications Server: IP Diagnosis Guide* to diagnose the problem.

Correcting input errors

For a user input error, use this procedure to assist in correcting the error:

1. Look up the command and its attributes (or parameters), the attributes in the attributes data set, and the entries in the exports data set in this book or the appropriate client machine command documentation.
2. Examine the attributes (or parameters) specified by each command, specified in the attributes data set, and the DIRECTORY entries specified in the exports data set to verify that they are specified correctly. Check the z/OS NFS server log data set, the z/OS NFS client log data set, or both log data sets to find the error message if applicable.
3. If you find a user input error, reenter the command or restart the server after you correct the error.
4. If all attributes (or parameters) and directory statements appear to be specified correctly, you should use this topic to build a set of keywords that describe the error, and then contact IBM for help.

After determining that the suspected failure is neither a z/OS TCP/IP program error nor a user error, you need to develop a set of keywords that describe the program failure. A keyword is an agreed-upon word or abbreviation used to describe a single aspect of a program failure.

After you have selected a set of keywords, use the set to search IBMLink/Service to determine whether an authorized program analysis report (APAR) has already been recorded for the failure. The IBMLink/Service online database contains information about the resolution of APARs. If a program failure is identified in these databases with the same set of keywords, the search yields a description of the problem and usually a fix. If the failure is not known, use the keywords to describe the failure when you contact IBM for help, which is the final step before an APAR is generated. For more information about these services see *z/OS DFSMSdfp Diagnosis*.

© Copyright IBM Corp. 1991, 2010

Using keywords to identify a problem

In general, when you contact IBM, you are asked to identify the problem with a full set of keywords. A full set of keywords for the NFS identifies these specific keywords:

- Component identification keyword
- Release level keyword
- Type-of-failure keyword
- Service level keyword

This example displays a full set of keywords:

```
5695DF121 R1VS MSGGFSA865I UW12345
```

Notes:

1. 5695DF121 - Signifies the component identification keyword.
2. R1VS - R1VS is the release level keyword for the z/OS NFS server, FMID HDZ11VS. R1VC is the release level keyword for the z/OS NFS client, FMID HDZ11VC. See Table 55 for a list of the z/OS NFS FMIDs and their corresponding releases.
3. MSGGFSA865I - MSGGFSA is the type-of-failure keyword prefix for the z/OS NFS server. MSGGFSC is the type-of-failure keyword prefix for the z/OS NFS client.
4. UW12345 - Service level keyword.

Table 55. z/OS NFS FMIDs and release names

FMID	z/OS Release
HDZ11VS	Release 7 NFS server
HDZ11VC	Release 7 NFS client
HDZ118N	Release 8 NFS client and server
HDZ119N	Release 9 NFS client and server
HDZ1AN0	Release 10 NFS client and server
HDZ1B1N	Release 11 NFS client and server

These sections explain individual keywords and their relation to the full set of keywords used to describe a NFS program failure.

A search in IBMLink/Service using the first keyword, the NFS component identifier, alone would detect all reported problems for the entire program product. Each keyword added to the search argument makes the search more specific, thereby reducing the number of problem descriptions to be considered. Sometimes you can find a correction for a problem without using the full set of keywords. Each keyword after the first is optional.

Component identification keyword

The component identification number is the first keyword in a set. You should use this keyword whenever the Network File System is suspected of being the failing component. The component identification keyword should be used with at least one other keyword to search IBMLink/Service. Used alone, it produces a full listing of APARs against the Network File System.

1. The component identification number for the Network File System is 5695DF121.

2. Go to "Release level keyword."

Release level keyword

Using this keyword to identify the release level of the Network File System is optional in the IBMLink/Service search argument. However, it is required when you submit an APAR.

1. Issue the MODIFY mvsnfs VERSION command, as shown in "Version operand" on page 237.
2. In output message GFSA944I, find the release level, as shown by *R1xx* in the following example:
   ```
   GFSA944I z/OS Network File System Server release R1xx ....
   ```
3. See "Type-of-failure keyword."

Type-of-failure keyword

To identify the type of failure that occurred, select the one keyword from Table 56 which best describes the problem. Then follow the specific instructions for that keyword. If you are not certain which of two keywords to use for the type of failure, use the one that appears first in the list.

Table 56. Summary of type-of-failure keywords

Keyword	Type of Failure	Page
ABEND*xxx*	Network File System ends abnormally because of a system-detected error	295
MSGGFS*hnnnt*	Error indicated by a system message	295
WAIT	The program does not seem to be doing anything	296
LOOP	The program is doing something repetitively	296
INCORROUT	Output from the program is incorrect or missing	296
DOC SCnnnnnnnn	Documentation is incorrect or incomplete	296
PERFM	Performance of the program is degraded	297

ABENDxxx

Use this procedure when Network File System ends abnormally.

In this step, you build a type-of-failure keyword:

1. Replace the *xxx* part of the **abend***xxx* keyword with the abend code from either the SYSLOG message or the abend dump. For example, if the abend code is 0C4, the ABENDxxx keyword is ABEND0C4.
2. To continue, see "Service level keyword" on page 297 to identify the service level of Network File System.

MSGGFShnnnt

Use this procedure for any of these conditions:

- Network File System message indicates an unexpected error detected.
- Message is not issued under conditions that should cause it to be issued.
- Message is issued under conditions that should not cause it to be issued.

In this step, you build a type-of-failure keyword.

1. For the *hnnnt* portion of the MSGGFShnnnt keyword replace the *h* with an **A** if the message prefix was GFSA or with a **C** if the message prefix was GFSC.

Replace the *nnn* with the message number and replace the *t* with the type code. For example, if the message is GFSA865I, the MSGGFShnnnt keyword is MSGGFSA865I.

2. To continue, see "Service level keyword" on page 297 to identify the service level of the Network File System.

WAIT

Use this procedure when Network File System suspends activity while waiting for some condition to be satisfied without issuing a message to tell why it is waiting.

In this step, you build a type-of-failure keyword:

1. If a Network File System task was in a WAIT, use the ***wait*** keyword.

   ```
   5695DF121 Rccc WAIT
   ```

2. To continue, see "Service level keyword" on page 297 to identify the service level of the Network File System.

LOOP

Use this procedure when some part of the program repeats endlessly.

In this step, you build a type-of-failure keyword.

1. If a Network File System task was in a LOOP, use the ***loop*** keyword.
2. If a message repeats endlessly, use the MSGGFS*hnnnt* keyword at the same time with the LOOP keyword. For the *hnnnt* portion of the MSGGFShnnnt keyword replace the *h* with an A if the message prefix was GFSA or with a C if the message prefix was GFSC. Replace the *nnn* with the message number and replace the *t* with the type code.

   ```
   5695DF121 Rccc LOOP MSGGFShnnnt
   ```

3. To continue, see "Service level keyword" on page 297 to identify the service level of the Network File System.

INCORROUT

Do not use ***incorrout*** if another keyword applies. Use this procedure only for these conditions:

- Output was expected but not received (missing).
- Output is different from expected (incorrect).

In this step, you build a type-of-failure keyword.

1. Use the incorrout keyword.
2. If the output is in the form of an incorrect message, also use the type-of-failure keyword MSGGFS*hnnnt*. For the *hnnnt* portion of the MSGGFShnnnt keyword replace the *h* with an A if the message prefix was GFSA or with a C if the message prefix was GFSC. Replace the *nnn* with the message number and replace the *t* with the type code.

   ```
   5695DF121 Rccc INCORROUT MSGGFShnnnt
   ```

3. To continue, see "Service level keyword" on page 297 to identify the service level of the Network File System.

DOC SCnnnnnnnn

Use this procedure when a program problem appears to be caused by incorrect or missing information in the Network File System documentation. If the documentation error is minor (for example, incorrect punctuation or a misspelled word), you do not need to build a keyword string to describe it. Instead, submit a Reader's Comment Form from the back of the book.

If the error is serious, and of general concern to other users, continue with this procedure.

In this step, you build a type-of-failure keyword, which includes a document order number. No other keywords are needed:

1. Place the order number of the document after the DOC keyword, omitting the hyphens. If the suffix is one digit, precede it with a zero. For example, the keyword for a document with order number SC26-7029-01 would be DOC SC26702901.
2. Locate the page in the document at which the error occurs, and prepare a description of the problem. If you submit an APAR, you must include this information in the error description.
3. To continue, see "Service level keyword" to identify the service level of the Network File System.

PERFM

Performance problems can be related to system tuning. Use this procedure when the performance problem cannot be corrected by system tuning and performance is below stated expectations.

In this step, you build a type-of-failure keyword.

1. Use the ***perfm*** keyword.
   ```
   5695DF121 Rccc PERFM
   ```
2. To continue, see "Service level keyword" to identify the service level of the Network File System.

Service level keyword

Use this keyword to identify the service level of the module that failed. The service level is defined as the most current fix applied to the module.

1. Issue the MODIFY mvsnfs VERSION=module command, as shown in "Version operand" on page 237.
2. In output message GFSA945I, find the service level of the module, as shown by *service_level* in the following example:
   ```
   GFSA945I modulename service_level compile_date/time
   ```
3. Specify the service level keyword.

   For example, if the service level of the failure related module is UW12345, you would specify UW12345 as the service level keyword.
   ```
   5695DF121 Rccc MSGGFSA865I UW12345
   ```

You now have all the necessary information for an effective search of known problems in IBMLink/Service. Refer to "Contacting the IBM Support Center" on page 312 or "Searching the IBM database for APARs and PTFs" on page 312.

Using z/OS component tracing

To improve the recording and tracing of NFS diagnostic information, you can use component trace services. After activating component tracing and recreating the problem, the diagnostic information in the trace buffers can then be captured in a z/OS SVC dump or written to a DASD or tape data set and then viewed using the IPCS CTRACE command. Use of component trace services offers other advantages over using the server's log data sets, including performance improvements and virtual storage constraint relief.

To activate component tracing of NFS information, an operator can issue a TRACE CT command specifying the procedure name of the z/OS NFS server or client with individual trace options or a member of SYS1.PARMLIB that contains the trace options. The operator can choose different PARMLIB members (if they were previously created) by specifying CTRACE=*nn* on a START mvsnfs command, where *nn* is the suffix of a CTINFS*nn* PARMLIB member. When CTRACE=*nn* is not specified, the default PARMLIB member, CTINFS00, is used to initialize CTRACE for the server. If the specified PARMLIB member is absent or incorrect, CTRACE is activated with the default options in effect (CTRACE buffer size of 10M and FFDC,ERROR trace options). For details on these commands and their NFS trace options, see Chapter 11, "Network File System operation," on page 211. For details on the member of SYS1.PARMLIB, see "Using NFS server component trace PARMLIB member CTINFS*nn*" on page 299 and "Using NFS client component trace PARMLIB member CTINFC*nn*" on page 300.

The following sections describe how to use NFS component tracing in diagnosis procedures. Further information on using the IPCS CTRACE command can be found in *z/OS MVS IPCS Commands*. Additional information regarding the MVS DISPLAY, TRACE, and DUMP commands may be found in *z/OS MVS System Commands*.

When you use component tracing, NFS diagnosis information continues to be recorded in the NFS server log data sets, at the level specified on a MODIFY *mvsnfs*,LOG=*xxx* operator command. IBM recommends that you adjust the level of logging to avoid duplicating diagnostic records and to maximize the performance benefits of using component trace buffers.

If additional NFS diagnosis information is required using the log data sets for the z/OS NFS server, see Appendix K, "Capturing diagnostic information using z/OS NFS log data sets and from other components," on page 523 for further details.

Component trace benefits

Component trace support offers the following benefits over the NFS log data sets:

- For virtual storage constraint relief, trace buffers are stored to a z/OS data space. In a multi-server implementation, each server or client has its own data space initialized at startup.
- Performance is improved because in-memory tracing can now be activated without the overhead of I/O.
- The optional use of an external writer data set may extend the amount of trace data collected when accompanied with a z/OS dump.
- For performance, the I/O overhead of writing to a data set is performed under the started external writer address space. Removing the I/O path provides a more responsive NFS error log daemon.
- Use of the IPCS MERGE command provides the ability to merge an NFS server or client component trace with those of other NFS or clients servers and related z/OS products such as TCPIP and UNIX System Services to obtain a single image of system activity.
- Performance is improved over the previous NFS server and client tracing methodology with the ability to enable only those trace events of interest. Previously, the amount of tracing was cumulative.

Using NFS server component trace PARMLIB member CTINFS*nn*

The CTINFS00 member of SYS1.PARMLIB (Figure 39 on page 300) provides default component trace values for the z/OS NFS server. This member is used automatically when the z/OS NFS server is started. You can create and use additional copies of the member with different trace option values, placing them in SYS1.PARMLIB using the naming convention CTINFS*nn*.

Component tracing is active from the NFS server's start with trace options defined in the CTINFS*nn* PARMLIB member, where 00 is the default. If the specified PARMLIB member is incorrect or absent, component tracing functions in minimum state (MIN) with a minimum set of trace options (FFDC and MSG). To start a component trace when the NFS server is active, the operator issues a TRACE CT command on the master console or a console with master authority. The operator can specify trace options individually on the TRACE CT command, or can specify the name of a CTINFS*nn* PARMLIB member containing the desired trace options. Using a PARMLIB member on the TRACE CT command can help minimize operator intervention and avoid syntax or keystroke errors. The syntax and options for a component trace of the z/OS NFS server are shown in "Starting component tracing for the z/OS NFS server" on page 218.

Note: The minimum trace options of FFDC and MSG remain in effect after the TRACE CT,OFF,COMP=MVSNFS operator command, so tracing never stops while the server is active.

The parmlib member used at NFS server startup specifies the size of the trace buffers to be used. Three buffers of that size are created and used cyclically. If no value is specified, or no parmlib member is found at startup, or a syntax error is encountered while processing the parmlib member, and the dsps startup parameter is also not specified, a default buffer size of 10 MB is used. The buffer size may be set in the range of 600KB to 600MB. The maximum value of 600MB was chosen because the total size of the three buffers must stay below 2 GB.

Note: The z/OS NFS server does not support changing the buffer size after server startup.

If no parmlib member is found at z/OS NFS server startup, or it does not contain a tracing Options specification, the server trace options default to an options setting of MIN (the minimum tracing options – MSG and FFDC).

CTINFS00 member of SYS1.PARMLIB

Figure 40 on page 303 shows a copy of the CTINFS00 member of SYS1.PARMLIB that NFS provides for tracing the z/OS NFS server. This file is used for setting the initial startup trace settings. In the file:

ON/OFF specifies whether trace is turned on or off.

OPTIONS specifies the trace options that are to be applied.

```
/* ================================================================ */
/*                                                                  */
/* $MAC(CTINFS00)  COMP(5694DF121)                             @P01C*/
/*                                                                  */
/* Z/OS Network File System Server.                                 */
/* Sample CTRACE options.                                           */
/*                                                                  */
/* COPYRIGHT:                                                       */
/*PROPRIETARY V3 STATEMENT                                          */
/*Licensed Materials - Property of IBM                              */
/*"Restricted Materials of IBM"                                     */
/*5694-A01                                                     @P01C*/
/*Copyright IBM Corp. 2004                                          */
/*END PROPRIETARY V3 STATEMENT                                      */
/*                                                                  */
/*                                                                  */
/* ---------------------------------------------------------------- */
/*                                                                  */
/* CHANGE ACTIVITY:                                                 */
/*                                                                  */
/* $L77=NFS,HDZ11VS,040507,IBSMVB: Convert To CTrace           @L77A*/
/* $L77=NFS,HDZ11VS,041210,IBSKYL: Add FFDC                   @L777A*/
/* $P01=KAJ0199,HDZ11VS,041216,SJPLMB: Restore PID to 5694-A01 @P01A*/
/*                                                                  */
/* ---------------------------------------------------------------- */
/* DEFAULT CTINFS00 MEMBER                                          */
/* ================================================================ */
/*                                                                  */
TRACEOPTS
/* ---------------------------------------------------------------- */
/*   Optionally start external writer in this file (use both        */
/*   WTRSTART and WTR with same wtr_procedure)                      */
/* ---------------------------------------------------------------- */
/*        WTRSTART(wtr_procedure)                                   */
/*        WTRSTART(CTWTR)                                           */
/* ---------------------------------------------------------------- */
/*   ON OR OFF: PICK 1                                              */
/* ---------------------------------------------------------------- */
          ON
/*        OFF                                                       */
/* ---------------------------------------------------------------- */
/*        WTR(CTWTR)                                                */
/* ---------------------------------------------------------------- */
/*   OPTIONS: NAMES OF FUNCTIONS TO BE TRACED.                      */
/* ---------------------------------------------------------------- */
          OPTIONS(
                  'INFO'
                 ,'WARNING'
                 ,'ERROR'
                 ,'FFDC'
                  )
/*                                                                  */
```

Figure 39. z/OS NFS server component trace PARMLIB member CTINFS00

Using NFS client component trace PARMLIB member CTINFC*nn*

The CTINFC00 member of SYS1.PARMLIB (Figure 39) provides default component trace values for the z/OS NFS client. This member is used automatically when the z/OS NFS client is started. You can create and use additional copies of the member with different trace option values, placing them in SYS1.PARMLIB using the naming convention CTINFC*nn*.

Component tracing is active from the NFS client's start with trace options defined in the CTINFC*nn* PARMLIB member, where 00 is the default. If the specified PARMLIB member is incorrect or absent, component tracing functions in minimum state (MIN) with a minimum set of trace options (FFDC and MSG). The operator can specify trace options individually on the TRACE CT command, or can specify the name of a CTINFC*nn* PARMLIB member containing the desired trace options. Using a PARMLIB member on the TRACE CT command can help minimize operator intervention and avoid syntax or keystroke errors. The syntax and options for a component trace of the z/OS NFS client are shown in "Starting component tracing for the z/OS NFS client" on page 212.

Note: The minimum trace options of FFDC and MSG remain in effect after the TRACE CT,OFF,COMP=MVSNFSC operator command, so tracing never stops while a client application is active.

To start a component trace when the NFS client is active, the operator issues a TRACE CT command on the master console or a console with master authority. The syntax and options for a component trace of the z/OS NFS client are shown in "Starting component tracing for the z/OS NFS client" on page 212. The operator can specify the trace options individually on the command, or can specify the name of a CTINFC*nn* PARMLIB member containing the desired trace options. Using a PARMLIB member on the TRACE CT command can help minimize operator intervention and avoid syntax or keystroke errors.

If the NFS client is started by z/OS UNIX (specified in the BPXPRM*xx* parmlib member), you can designate a component trace parmlib member with options to be used at startup. Use the CTRACE=*nn* parameter, as shown in the following example:

```
FILESYSTYPE TYPE(NFS) ENTRYPOINT(GFSCINIT) ASNAME(mvsnfsc)
PARM(INITD,CTRACE=nn,...)
```

where:

- *mvsnfsc* is the name of the z/OS NFS procedure to be started.
- *nn* is the suffix of a CTINFC*nn* PARMLIB member.
- CTRACE must appear as the first or second parameter after INITD, in capital letters, separated by commas without blank spaces. If CTRACE=*nn* is not specified, the default PARMLIB member, CTINFC00, is used to initialize CTRACE for the client.

The component trace options specified in that member then control the component trace. These trace options can be overridden by a subsequent TRACE CT command if necessary, for example if a problem arises requiring a different set of trace options.

The parmlib member used at NFS Client startup specifies the size of the trace buffers to be used. Three buffers of that size are created and used cyclically. If no value is specified, or no parmlib member is found at startup, or a syntax error is encountered while processing the parmlib member, a default buffer size of 10 MB is used. The buffer size may be set in the range of 600KB to 600MB. The maximum value of 600MB was chosen because the total size of the three buffers must stay below 2 GB.

Note: The z/OS NFS client does not support changing the buffer size after client startup.

If no parmlib member is found at z/OS NFS client startup, or it does not contain a tracing Options specification, the client trace options default to an options setting of MIN (the minimum tracing options – MSG and FFDC).

CTINFC00 member of SYS1.PARMLIB

Figure 40 on page 303 shows a copy of the CTINFC00 member of SYS1.PARMLIB that NFS provides for tracing the z/OS NFS client. This file is used for setting the initial startup trace settings. In the file:

ON/OFF specifies whether trace is turned on or off.

OPTIONS specifies the trace options that are to be applied.

```
/* ================================================================ */
/*                                                                  */
/* $MAC(CTINFC00)  COMP(5694DF121)                            @L84A */
/*                                                                  */
/* Z/OS Network File System Client.                                 */
/* Sample CTRACE options.                                           */
/*                                                                  */
/* PROPRIETARY STATEMENT=                                           */
/*   LICENSED MATERIALS - PROPERTY OF IBM                           */
/*   THIS MACRO IS "RESTRICTED MATERIALS OF IBM"                    */
/*   5694-A01                                                       */
/*   COPYRIGHT IBM CORPORATION 2005                                 */
/*   SEE IBM COPYRIGHT INSTRUCTIONS                                 */
/* END PROPRIETARY STATEMENT                                        */
/*                                                                  */
/*                                                                  */
/* ---------------------------------------------------------------- */
/*                                                                  */
/* CHANGE ACTIVITY:                                                 */
/*                                                                  */
/* $L84=NFS,HDZ118N,051031,IBSKVV: Convert To CTrace           @L84A*/
/*                                                                  */
/* ---------------------------------------------------------------- */
/* DEFAULT CTINFC00 MEMBER                                          */
/* ================================================================ */
/*                                                                  */
TRACEOPTS
/* ---------------------------------------------------------------- */
/*   Optionally start external writer in this file (use both       */
/*   WTRSTART and WTR with same wtr_procedure)                      */
/* ---------------------------------------------------------------- */
/*        WTRSTART(wtr_procedure)                                   */
/*        WTRSTART(CTWTR)                                           */
/* ---------------------------------------------------------------- */
/*   ON OR OFF: PICK 1                                              */
/* ---------------------------------------------------------------- */
          ON
          BUFSIZE(10M)
/*        OFF                                                       */
/* ---------------------------------------------------------------- */
/*        WTR(CTWTR)                                                */
/* ---------------------------------------------------------------- */
/*   OPTIONS: NAMES OF FUNCTIONS TO BE TRACED.                      */
/* ---------------------------------------------------------------- */
/*        OPTIONS(                 */
/**********  FFDC IS ALWAYS ON *****/
/*                'FFDC'           */
/*               ,'ENTRY'          */
/*               ,'EXIT'           */
/*               ,'SUSPEND'        */
/*               ,'RESUME'         */
/*               ,'SCHEDULE'       */
/*               ,'DISPATCH'       */
/*               ,'USS_REQUEST'    */
/*               ,'USS_RETURN'     */
/*               ,'NFS_REQUEST'    */
/*               ,'NFS_RETURN'     */
/*               ,'CB_MGMT'        */
```

Figure 40. z/OS NFS client component trace PARMLIB member CTINFC00 (Part 1 of 2)

```
/*               ,'NETWORK'        */
/*               ,'GENERAL'        */
/*               ,'DETAIL'         */
/*               ,'TRAP'           */
/***** MSG IS ALWAYS ON ************/
/*               ,'MSG'            */
/*               ,'BUFFER'         */
/*               ,'LOCK_REQUEST'   */
/*               ,'LOCK_RESUME'    */
/*               ,'LOCK_RELEASE'   */
/*               ,'ALL'            */
/*               ,'CALL'           */
/*               ,'TASK_FLOW'      */
/*               ,'USS'            */
/*               ,'LOCK'           */
/*               ,'NFS'            */
/*                )                */
/*                                 */
```

Figure 40. z/OS NFS client component trace PARMLIB member CTINFC00 (Part 2 of 2)

Capturing NFS Server component trace information in an SVC dump

You can use component tracing to gather information for diagnosis when recreating prior problems related to the z/OS NFS server. To do so, follow these steps:

1. Have an operator issue a TRACE CT command, as described in "Starting component tracing for the z/OS NFS server" on page 218. Specify individual trace options on the command, or OPTIONS=All to include all trace types.
2. Recreate the reported problem. After the problem is recreated, disable tracing with the following command at the operator console:

   ```
   TRACE CT,OFF,COMP=MVSNFS
   ```

   which will convert the NFS server to MIN state of tracing.
3. Create an MVS dump

   To create a dump of the an NFS server address space and associated data space, enter the following command at the operator console:

   ```
   DUMP COMM=(any dump description title you choose)
   ```

   In response to operator message *nn IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND, reply with:

   ```
   R nn, JOBNAME=(MVSNFS),DSPNAME=('mvsnfs'.NFSCTRDS),CONT
   R nn,SDATA=(LPA,CSA,ALLNUC,GRSQ,LSQA,SWA,PSA,SQA,TRT,RGN,SUM)
   ```

   To include NFS component trace records in an SVC dump, specify the associated data space name '*mvsnfs*'.NFSCTRDS on the MVS DUMP command. The value of *mvsnfs* represents the server's procedure name.

   You can now use the IPCS command CTRACE COMP(MVSNFS) FULL to look at the trace records, where the default data set has been set to a captured dump data set. NFS provides both IPCS exits and format tables to facilitate this activity.

Capturing NFS Client component trace information in an SVC dump

You can use component tracing to gather information for diagnosis when recreating prior problems related to the z/OS NFS client. To do so, follow these steps:

1. Have an operator issue a TRACE CT command, as described in “Starting component tracing for the z/OS NFS client” on page 212. Specify individual trace options on the command, or OPTIONS=All to include all trace types.
2. Recreate the reported problem. After the problem is recreated, disable tracing with the following command at the operator console:

   ```
   TRACE CT,OFF,COMP=MVSNFSC
   ```

   which will convert the NFS server to MIN state of tracing.
3. Create an MVS dump

   To create a dump of the an NFS client address space and associated data space, enter the following command at the operator console:

   ```
   DUMP COMM=(any dump description title you choose)
   ```

   In response to operator message *nn IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND, reply with:

   ```
   R nn,JOBNAME=(MVSNFSC),DSPNAME=('MVSNFSC'.NFSCTRDS),CONT
   R nn,SDATA=(LPA,CSA,ALLNUC,GRSQ,LSQA,SWA,PSA,SQA,TRT,RGN,SUM)
   ```

   To include NFS component trace records in an SVC dump, specify the associated data space name *'MVSNFSC'*.NFSCTRDS on the MVS DUMP command. The value of *MVSNFSC* represents the client's procedure name.

   You can now use the IPCS command CTRACE COMP(MVSNFSC) FULL to look at the trace records, where the default data set has been set to a captured dump data set. NFS provides both IPCS exits and format tables to facilitate this activity.

Using a z/OS component trace external writer

z/OS Component Trace (CT) supports an external writer which can be used to write trace records to a DASD or tape data set in real time. That is, as NFS generates trace records those records can be placed on DASD or tape.

Activating a CT External Writer for a program is a two-step process. These steps can be done in either order, but the external writer will not start writing out trace records until they have both been done.

Step 1. Telling MVS to start a CT external writer

Before a CT external writer can start there must be a procedure in SYS1.PROCLIB that tells how to invoke the writer. An example of a CT external writer procedure that can be placed in SYS1.PROCLIB(CTWTR) is:

```
//CTWTR    PROC
//IEFPROC  EXEC PGM=ITTTRCWR
//TRCOUT01 DD   DSN=IBMUSER.CT1,DISP=NEW
```

If you are tracing in a multi-server or sysplex environment, the data set names on TRCOUTnn DD statements must be unique throughout the sysplex. An ENQUEUE error results if the data set names are not unique. *z/OS MVS Diagnosis: Tools and Service Aids* has a discussion about creating external writer procedures.

An example of values to use in allocating IBMUSER.CT1 is:

```
DATA SET NAME: IBMUSER.CT1

VOLUME SERIAL      ===> USRPAK    (BLANK FOR AUTHORIZED DEFAULT
GENERIC UNIT       ===>           (GENERIC GROUP NAME OR UNIT
SPACE UNITS        ===> CYLS      (BLKS, TRKS, OR CYLS)
PRIMARY QUANTITY   ===> 3         (IN ABOVE UNITS)
SECONDARY QUANTITY ===> 5         (IN ABOVE UNITS)
DIRECTORY BLOCKS   ===> 0         (ZERO FOR SEQUENTIAL DATA SET)
RECORD FORMAT      ===> VB
RECORD LENGTH      ===> 23472
BLOCK SIZE         ===> 23476
EXPIRATION DATE    ===>           (YY/MM/DD, YYYY/MM/DD
```

To start a CT external writer, enter the following command at the operator console:

```
TRACE CT,WTRSTART=CTWTR
```

When you see the following message, the CT external writer is now ready:

```
ITT110I INITIALIZATION OF CTRACE WRITER CTWTR COMPLETE.
```

Step 2. Telling NFS to connect to a CT external writer and start sending records

To have the NFS server start sending trace records to the CT external writer, enter the following command at the operator console:

```
TRACE CT,ON,COMP=MVSNFS
```

To have the NFS client start sending trace records to the CT external writer, enter the following command at the operator console:

```
TRACE CT,ON,COMP=MVSNFSC
```

In response to operator message *nn ITT006A SPECIFY OPERAND(S) FOR TRACE CT COMMAND reply with:

```
R nn,WTR=CTWTR,OPTIONS=(value(,value)),END
```

NFS will now start sending its trace records to the CT external writer, which in turn will start writing them to IBMUSER.CT1. A description of NFS trace option values (FFDC, Entry, Exit, etc.) is listed under “Starting component tracing for the z/OS NFS server” on page 218.

Using IPCS to view records from an external writer

Before you can use IPCS to look at trace records captured by a CT external writer, the remaining trace buffers must be queued to the external writer and the writer must be stopped in order to place an end-of-file record at the end of the data set.

For the z/OS NFS server, to stop NFS tracing issue the command:

```
TRACE CT,OFF,COMP=MVSNFS
```

For the z/OS NFS client, to stop NFS tracing and flush the remaining buffer to the external writer, issue the command:

```
TRACE CT,OFF,COMP=MVSNFSC
```

You can stop the writer when control is received back from the flush command. Issue the following command to stop the writer and make the data set available to IPCS:

```
TRACE CT,WTRSTOP=CTWTR
```

You can now use the IPCS CTRACE command CTRACE COMP(MVSNFSC) FULL to look at the trace records, where the default data set has been set to IBMUSER.CT1.

When writing NFS trace records from more than one NFS server or client into separate component trace data sets, allocate each data set on a separate volume for improved performance. Use the IPCS COPYTRC command to merge records from multiple component trace data sets into one data set. Each component trace data set may represent one NFS server or one NFS client in the NFS multi-server or z/OS sysplex implementations. See *z/OS MVS IPCS Commands* for additional information.

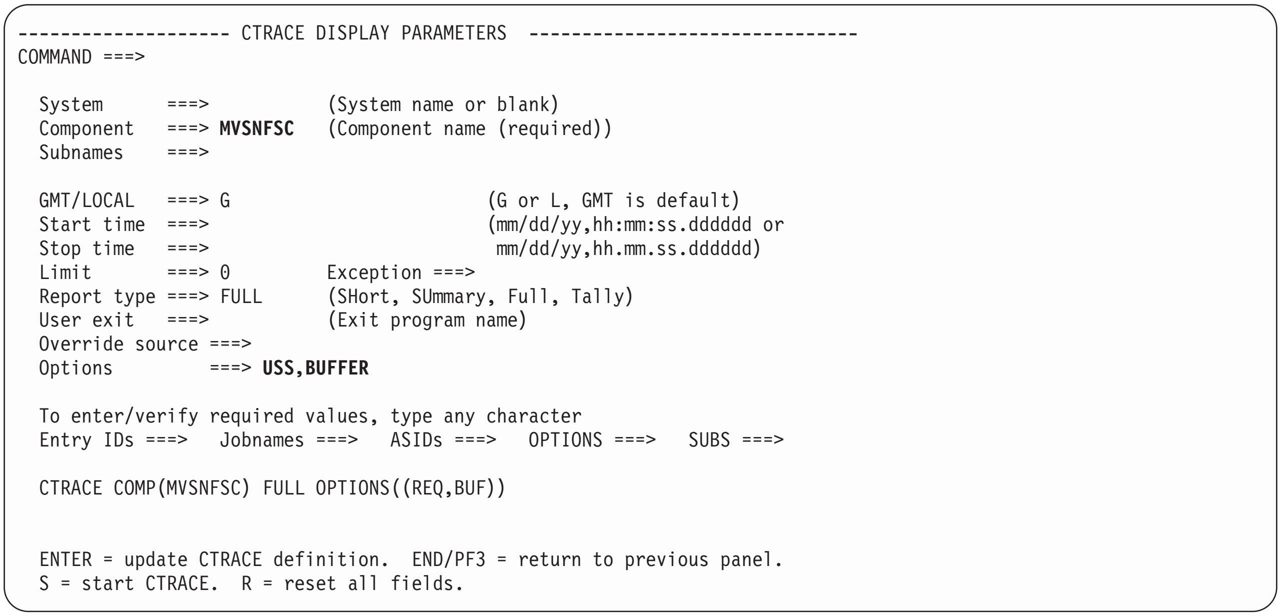
Filtering NFS ctrace records in IPCS

The z/OS NFS Ctrace function allows you to restrict the trace records to be processed by IPCS and displayed on the trace screen. Use the CTRACE DISPLAY PARAMETERS panel, shown in Figure 41 on page 308, to do so.

Use the Component field to specify whether you want trace records for the NFS server or NFS client. Specify the name of the server or client as it appears in your start procedure. The example shown in Figure 41 on page 308 uses MVSNFSC as the name of the NFS client.

Use the Options field to specify any of the following:

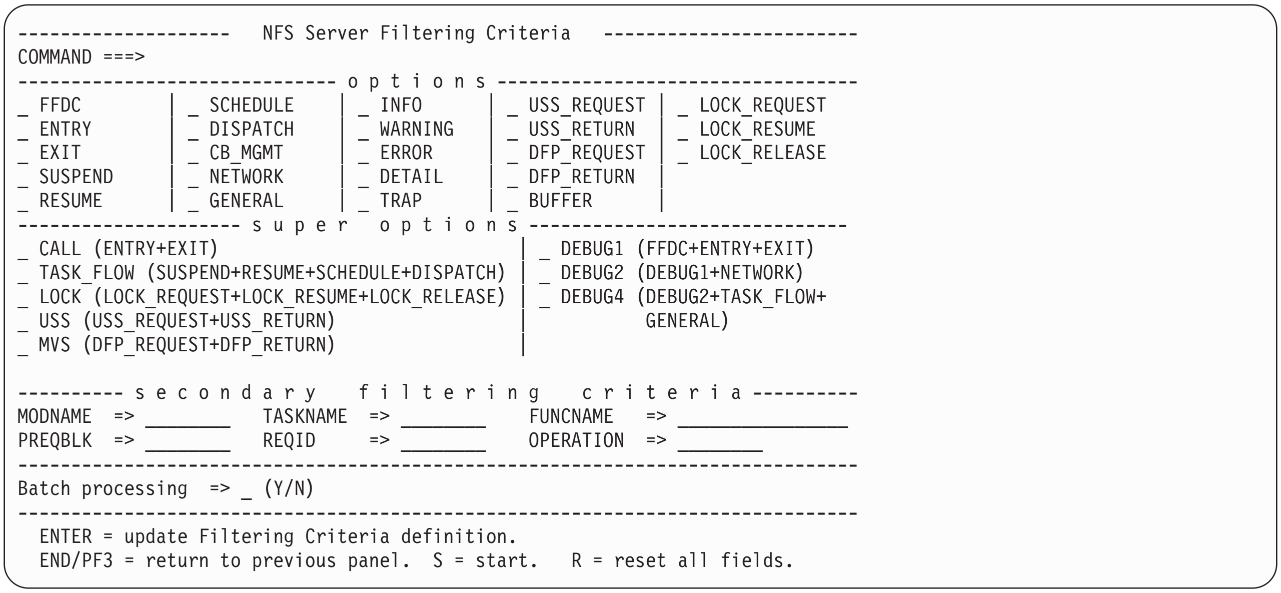
- Nothing, in which case all records will be processed and displayed.
- A specific list of record types to be processed and displayed. In this case, you can use shorthand record specifications (for example, CALL, TASK_FLOW), as well as the full-length record type names. All other record types will be filtered out and not displayed. Any invalid option values are ignored. If no valid options are specified, no trace records are displayed. Figure 41 on page 308 shows an example of selecting only the BUFFER and USS (that is, USS_REQUEST and USS_RETURN) record types.
- The value "PANEL" to display a filtering criteria panel, which provides another method of specifying options. If the Component field specifies the name of the NFS server, then the NFS Server Filtering Criteria panel (shown in Figure 42 on page 308) is displayed. If the Component field specifies the name of the NFS client, then the NFS Client Filtering Criteria panel (shown in Figure 43 on page 309) is displayed.



```
-------------------- CTRACE DISPLAY PARAMETERS  ------------------------------
COMMAND ===>

  System      ===>           (System name or blank)
  Component   ===> MVSNFSC   (Component name (required))
  Subnames    ===>

  GMT/LOCAL   ===> G                        (G or L, GMT is default)
  Start time  ===>                          (mm/dd/yy,hh:mm:ss.dddddd or
  Stop time   ===>                           mm/dd/yy,hh.mm.ss.dddddd)
  Limit       ===> 0         Exception ===>
  Report type ===> FULL      (SHort, SUmmary, Full, Tally)
  User exit   ===>           (Exit program name)
  Override source ===>
  Options         ===> USS,BUFFER

  To enter/verify required values, type any character
  Entry IDs ===>   Jobnames ===>   ASIDs ===>   OPTIONS ===>   SUBS ===>

  CTRACE COMP(MVSNFSC) FULL OPTIONS((REQ,BUF))


  ENTER = update CTRACE definition.  END/PF3 = return to previous panel.
  S = start CTRACE.  R = reset all fields.
```

Figure 41. CTRACE Display Parameters panel

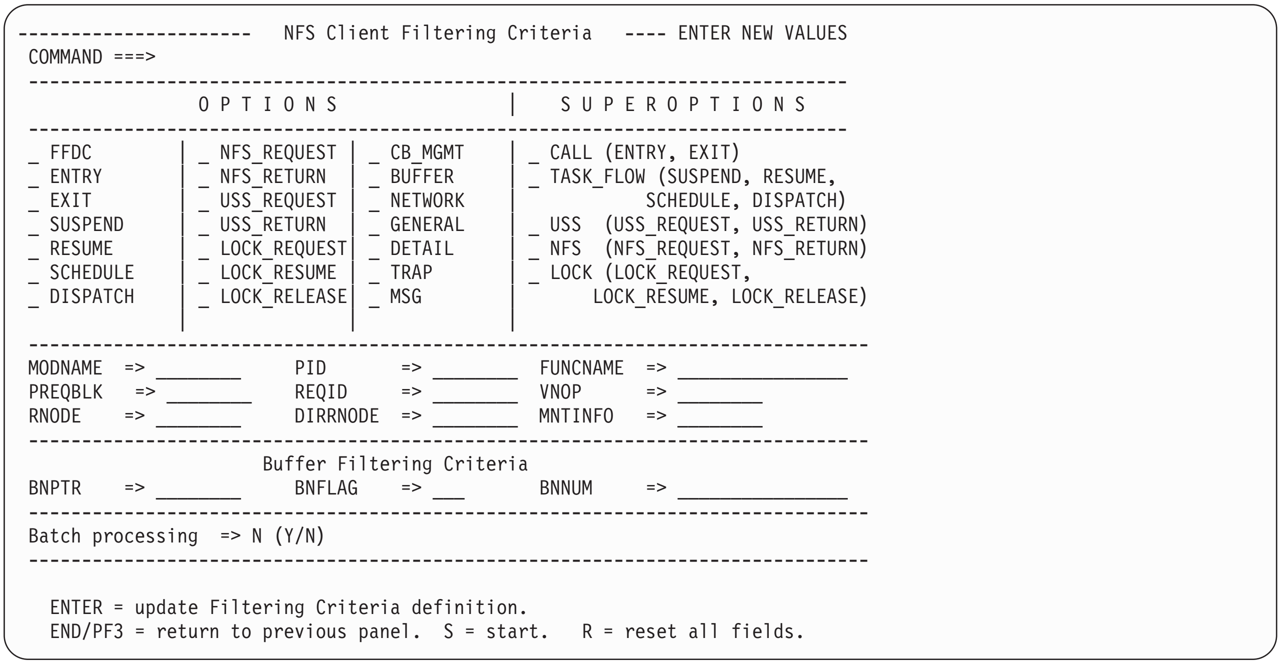
If you specify the NFS server name in the Components field and PANEL in the Options field, then the NFS Server Filtering Criteria panel is displayed:



```
--------------------   NFS Server Filtering Criteria   -----------------------
COMMAND ===>
------------------------------ o p t i o n s ---------------------------------
_ FFDC        | _ SCHEDULE    | _ INFO      | _ USS_REQUEST | _ LOCK_REQUEST
_ ENTRY       | _ DISPATCH    | _ WARNING   | _ USS_RETURN  | _ LOCK_RESUME
_ EXIT        | _ CB_MGMT     | _ ERROR     | _ DFP_REQUEST | _ LOCK_RELEASE
_ SUSPEND     | _ NETWORK     | _ DETAIL    | _ DFP_RETURN  |
_ RESUME      | _ GENERAL     | _ TRAP      | _ BUFFER      |
-------------------- s u p e r   o p t i o n s ------------------------------
_ CALL (ENTRY+EXIT)                          | _ DEBUG1 (FFDC+ENTRY+EXIT)
_ TASK_FLOW (SUSPEND+RESUME+SCHEDULE+DISPATCH) | _ DEBUG2 (DEBUG1+NETWORK)
_ LOCK (LOCK_REQUEST+LOCK_RESUME+LOCK_RELEASE) | _ DEBUG4 (DEBUG2+TASK_FLOW+
_ USS (USS_REQUEST+USS_RETURN)               |          GENERAL)
_ MVS (DFP_REQUEST+DFP_RETURN)               |

---------- s e c o n d a r y   f i l t e r i n g   c r i t e r i a ----------
MODNAME  => ________   TASKNAME  => ________    FUNCNAME   => ________________
PREQBLK  => ________   REQID     => ________    OPERATION  => ________
------------------------------------------------------------------------------
Batch processing  => _ (Y/N)
------------------------------------------------------------------------------
  ENTER = update Filtering Criteria definition.
  END/PF3 = return to previous panel.  S = start.   R = reset all fields.
```

Figure 42. NFS Server Filtering Criteria

If you specify the NFS client name in the Components field and PANEL in the Options field, then the NFS Client Filtering Criteria panel is displayed:



```
----------------------  NFS Client Filtering Criteria   ---- ENTER NEW VALUES
 COMMAND ===>
 -----------------------------------------------------------------------------
                O P T I O N S              |   S U P E R O P T I O N S
 -----------------------------------------------------------------------------
 _ FFDC        | _ NFS_REQUEST | _ CB_MGMT    | _ CALL (ENTRY, EXIT)
 _ ENTRY       | _ NFS_RETURN  | _ BUFFER     | _ TASK_FLOW (SUSPEND, RESUME,
 _ EXIT        | _ USS_REQUEST | _ NETWORK    |          SCHEDULE, DISPATCH)
 _ SUSPEND     | _ USS_RETURN  | _ GENERAL    | _ USS  (USS_REQUEST, USS_RETURN)
 _ RESUME      | _ LOCK_REQUEST| _ DETAIL     | _ NFS  (NFS_REQUEST, NFS_RETURN)
 _ SCHEDULE    | _ LOCK_RESUME | _ TRAP       | _ LOCK (LOCK_REQUEST,
 _ DISPATCH    | _ LOCK_RELEASE| _ MSG        |     LOCK_RESUME, LOCK_RELEASE)
               |               |              |
 -------------------------------------------------------------------------------
 MODNAME  => ________    PID      => ________  FUNCNAME  => ________________
 PREQBLK   => ________   REQID    => ________  VNOP      => ________
 RNODE    => ________    DIRRNODE => ________  MNTINFO   => ________
 -------------------------------------------------------------------------------
                      Buffer Filtering Criteria
 BNPTR    => ________    BNFLAG   => ___       BNNUM     => ________________
 -------------------------------------------------------------------------------
 Batch processing  => N (Y/N)
 -------------------------------------------------------------------------------

   ENTER = update Filtering Criteria definition.
   END/PF3 = return to previous panel.  S = start.   R = reset all fields.
```

Figure 43. NFS Client Filtering Criteria

Any combination of record types can be selected on the Options and Superoptions sections of the Filtering Criteria menus by putting an X or other character before the desired option. Within those record types, further selection filtering can be done based on the following criteria (on either Filtering Criteria panel):

FUNCNAME

1 to 16 character name of a specific z/OS NFS server or client function for which trace records are to be displayed. This will limit the display to only trace records issued from within the selected function.

Note: Because function names are mixed case, but this parameter can not be specified in mixed case in all instances, the filter function will match the names regardless of case. For example, FUNC1 will match func1, Func1, FUNC1, and so on.

If the specified name ends in an asterisk, all function names starting with the specified string will be considered a match and will be selected. For example, FUN* will match FUNC1, function2, Funny, and so on. If the asterisk is the only character specified for FUNCNAME, all function names will be considered a match. The asterisk can be specified only on the right end of the string or as the only character in the string. No other wildcard variations are supported.

MODNAME

1 to 8 character name of the z/OS NFS server or client module for which trace records are to be displayed. This will limit the display to only trace records issued from within the selected load module.

PREQBLK

1 to 8 hexadecimal digit number representing the address of a specific request block for which trace records are to be displayed. Note that request blocks are reused.

REQID
1 to 8 hexadecimal digit number representing a specific request for which trace records are to be displayed.

The following fields apply only to the NFS Server Filtering Criteria panel (Figure 42 on page 308):

OPERATION
Four-byte, 8-character, hexadecimal representation of the NFS V2 or V3 request, or V4 operation, for which trace records are to be displayed. For example, 0xFFFFFF60.

TASKNAME
1 to 8 character name of the z/OS NFS server task for which trace records are to be displayed. This will limit the display to only trace records issued from within the selected task.

The following fields apply only to the NFS Client Filtering Criteria panel (Figure 43 on page 309):

DIRRNODE
1 to 8 hexadecimal digit number representing the address of a specific parent directory rnode associated with the trace records to be displayed.

MNTINFO
1 to 8 hexadecimal digit number representing the address of a specific mount info block associated with the trace records to be displayed.

PID
1 to 8 hexadecimal digit number representing the process id of the z/OS NFS client process for which trace records are to be displayed.

RNODE
1 to 8 hexadecimal digit number representing the address of a specific rnode associated with the trace records to be displayed.

VNOP
1 to 8 character name of the vnode/vfs operation (for example, vn_rdwr) for which trace records are to be displayed. Only a single vnode/vfs operation name can be specified.

Buffer record types
For BUFFER record types on the client filtering panel, several additional selection criteria are available. These criteria are ignored if no buffer records are selected.

BNNUM
1 to 16 hexadecimal digit number representing a specific buffer associated with the BUFFER trace records that are to be displayed.

BNFLAG
3 bit flag field associated with the BUFFER trace records that are to be displayed. This value is treated as a mask for identifying buffers whose selected bits are on.

BNPTR
1 to 8 hexadecimal digit number representing the address of a specific 8K buffer associated with the BUFFER trace records that are to be displayed.

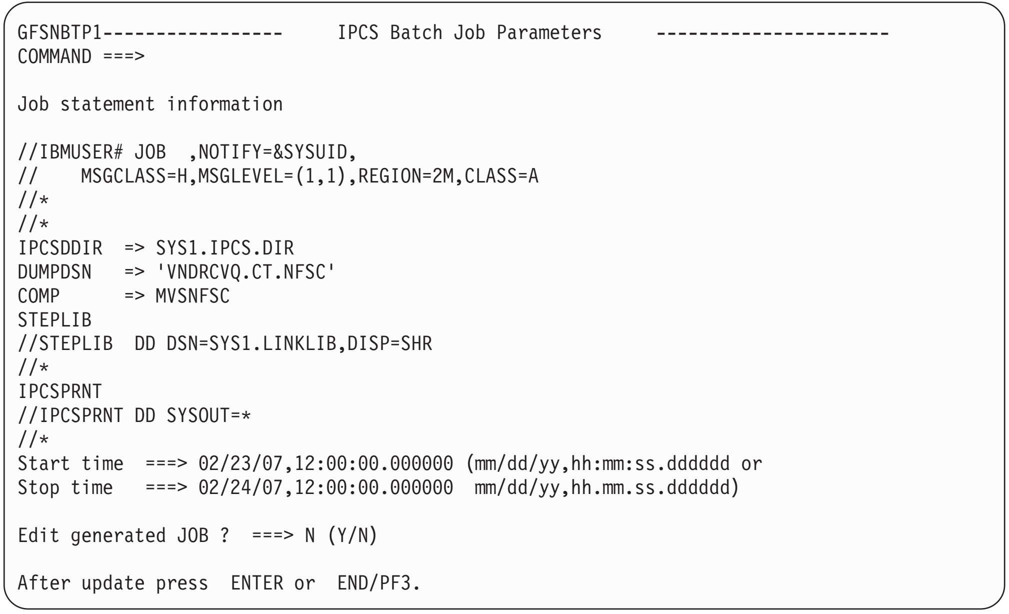
Notes:

1. If multiple criteria are specified, records matching any of the record type options and all of the other specified criteria are considered to be a match and are selected.
2. Any of these secondary options can be specified on the IPCS Ctrace Display Parameters panel OPTIONS list if one of the Filtering Criteria panels is not being used.

Processing Traces in Batch Mode: You can process traces in batch mode by running IPCS under the TSO/E terminal monitor program (TMP) in a batch job. Since it can take a long time to process all of the trace records in a large trace file, use of batch mode for filtering the records and producing a readable output file can be helpful by not tying up the debugger's TSO session the entire time. This section discusses only the batch mode features specifically applicable to NFS traces. For general information on the use of batch mode, see *z/OS MVS IPCS User's Guide*, SA22-7596.

The trace analysis can be submitted from the NFS Filtering Criteria panel by specifying "Y" in the "Batch Processing" field on the menu.

When batch processing is selected from the NFS Filtering Criteria panel, that panel will be followed by an IPCS Batch Job Parameters panel, with contents similar to this example:



```
GFSNBTP1-----------------     IPCS Batch Job Parameters     ---------------------
COMMAND ===>

Job statement information

//IBMUSER# JOB  ,NOTIFY=&SYSUID,
//    MSGCLASS=H,MSGLEVEL=(1,1),REGION=2M,CLASS=A
//*
//*
IPCSDDIR  => SYS1.IPCS.DIR
DUMPDSN   => 'VNDRCVQ.CT.NFSC'
COMP      => MVSNFSC
STEPLIB
//STEPLIB  DD DSN=SYS1.LINKLIB,DISP=SHR
//*
IPCSPRNT
//IPCSPRNT DD SYSOUT=*
//*
Start time  ===> 02/23/07,12:00:00.000000 (mm/dd/yy,hh:mm:ss.dddddd or
Stop time   ===> 02/24/07,12:00:00.000000  mm/dd/yy,hh.mm.ss.dddddd)

Edit generated JOB ?  ===> N (Y/N)

After update press  ENTER or  END/PF3.
```

The information from this panel is used together with a skeleton file for building the JCL to be submitted for the batch job. If desired, you can edit the JCL before it is submitted. This is done by specifying "Y" on the "Edit generated JOB ?" line. The default for this option is N. Once the JCL has been edited, you can submit the job by issuing the SUBMIT command on the editor command line.

Setting up a dump data set for abnormal ends

Normally, the Network File System ESTAE issues a SVC dump when failure occurs. However, if ESTAE is not able to do this, z/OS takes over and issues the appropriate dump you coded in your DD statement. This is an example of setting up a DUMP data set:

```
//SYSMDUMP  DD   DISP=SHR,DSN=MVSNFS.SYSMDUMP
```

File attributes of this DUMP data set should be set up like this:

```
Organization      PS
Record Format     FB
Record Length     4160
Block Size        4160
```

Searching the IBM database for APARs and PTFs

If your installation has access to the interactive online database program, IBMLink/ServiceLink, you can use IBMLink/ServiceLink to perform these tasks:

1. Search and browse for an existing APAR that is similar to your problem. Use the full set of keywords that is developed from the diagnostic procedures. Use only the keywords that are described in this book. Make sure that keywords are spelled exactly as they are described in this book.
2. If an APAR exists, search to see if a program temporary fix (PTF) is available.
3. If a PTF is available, order the PTF.
4. If an APAR does not exist, you can create an Electronic Technical Response (ETR) problem report to receive assistance from a z/OS NFS service representative. See "Contacting the IBM Support Center" for the type of information you will need.

Contacting the IBM Support Center

If your installation does not have access to IBMLink/ServiceLink, z/OS NFS also provides telephone support. Within the U.S.A. and Puerto Rico, call the following number and request assistance for the z/OS NFS feature by specifying the program number 5695DF121 and release level keyword.

```
IBM Support Center
1-800-237-5511
Monday through Friday, 8 a.m.-5 p.m. (excluding national holidays)
```

Outside of the U.S.A. and Puerto Rico, contact your local IBM representative.

When contacting IBM, be prepared to supply the following information:

- Your customer number
- Release level
- Current® service level (from the APAR list)
- Keyword set or sets used to search in IBMLink/Service

You will be asked to describe the Network File System server and client machine environment. The IBM support representative might request the following relevant information.

- A minimum set of input commands on the client machine or z/OS operator console that reproduces the error.
- A copy of the minimum output from the client machine or z/OS operator console necessary to illustrate the failure.
- A copy of the z/OS NFS server log data set, the z/OS NFS client log data set, or both log data sets created by the input commands provided to recreate the error.
- Storage dump (if for an abnormal end).
- Linkedit map (if for abnormal end).
- Other supporting material, such as trace file printout from a network analyzer.

- For DOC SC26-7417 (*z/OS Network File System Guide and Reference*) failures, include the revision number and page(s) containing the error, and a description of the problem it caused.
- A copy of the attributes data set.
- A copy of the exports data set.
- A copy of the Network File System startup procedure.

Submitting Documentation on Tape: If the IBM service representative requests you submit documentation on tape, please write it to a standard label tape and include a hard copy of the DCB information for each data set along with the JCL used to create the tape.

Diagnostic aids

A description of first failure data capture, including SVC dump characteristics, dump contents, and errors and messages, is provided with the NFS as a major diagnostic aid.

First failure data capture

Network File System RAS characteristics are improved by the capture of diagnostic service data. Error records are written to SYS1.LOGREC and dumps are requested to SYS1.DUMPnn (these are in addition to the existing server trace). Component-specific information is provided in the SYS1.LOGREC entry and in the dump for the generation of RETAIN® search symptom strings.

Symptom data

Table 57 lists the component-specific symptom data placed in the System Diagnostic Work Area (SDWA).

Table 57. NFS symptom data

SDWA Field	Meaning	RETAIN Key
SDWAMODN	Active load module name	RIDS/
SDWACSCT	Active CSECT name	RIDS/
SDWAMDAT	Active CSECT assembly date	VALU/C
SDWAMVRS	Active CSECT service level	VALU/C
SDWAREXN	Recovery routine module name	RIDS/
SDWARRL	Recovery routine label name	FLDS/
SDWACID	Component identifier	PIDS/
SDWACIDB	Base component identifier	PIDS/
SDWASC	Active server function name	RIDS/

SVC dump

The Network File System SVC dumps have these characteristics:

Dump title: The dump title contains the component name, component identifier, release level, abend code, reason code, and the name of the ESTAE module requesting the dump. If available, the name of the failing module and the offset within the module are included.

Dump content: Table 58 on page 314 shows the dump options and areas of storage that are included in the dump request:

Table 58. Dump content and storage areas

Dump Options	Storage Areas
SUMDUMP	Suspend summary dump
RGN	Server private area storage; programs and subpools
TRT	z/OS trace table
GRSQ	GRS ENQ control blocks
IO	I/O data areas
ALLPSA	All Prefixed Storage Areas
DFA	Data Facility Area
DFVT	Data Facility Vector Table
NFSSVT	Network File System Vector Table

Eye-catchers: Each CSECT within each server load module is identified by the CSECT name, the compile date, and the FMID or APAR level. Each function within a CSECT is identified by its variable length name.

Dump suppression

z/OS Dump Analysis and Elimination (DAE) is supported by providing sufficient information for DAE to uniquely identify the dump and by setting the VRADAE key in the Variable Recording Area (VRA) of the SDWA.

Data capture suppression

SYS1.LOGREC entries and SVC dumps are not requested for these abend codes:

X'0F3'
X'806'
X'A03'
X'x13'
X'x22' (except X'122')
X'x37'
X'x3E'

Invocation

A server ESTAE instance is entered whenever any server task ends abnormally. It is the ESTAE’s responsibility to ensure that adequate and correct diagnostics are captured.

Errors and messages

Table 59 on page 315 illustrates the diagnostic errors and messages GFSA470I and GFSA471I.

Message GFSA470I is written if the SVC dump request fails. The message contains the error reason code from the SDUMP service.

Secondary ESTAE routines detect failures during the execution of the primary server ESTAE. If ESTAE processing is unable to complete, message GFSA471I is issued, and the server task is allowed to stop. The message contains the last abend code detected by the secondary ESTAE routines.

Table 59. Diagnostic errors and messages

Message	Explanation
GFSA470I	NETWORK FILE SYSTEM SVC DUMP REQUEST FAILED. REASON = hh
GFSA471I	NETWORK FILE SYSTEM ESTAE EXIT UNABLE TO COMPLETE PROCESSING. ABEND = X'xxx'

Debug trace data capture

The z/OS Network File System uses trace facilities to record debug trace diagnostic information when a problem requires additional diagnostic information beyond diagnostic messages. Assuming that debug trace diagnostics were not activated at the time of the original failure, the problem must be recreated a second time and the NFS debug trace facilities turned on to capture the diagnostic information. The server provides debug trace diagnostic information from two separate trace facilities. The z/OS Network File System client provides debug trace diagnostic information from one trace facility.

The z/OS NFS server records z/OS NFS debug trace diagnostic information to the z/OS component trace buffer or to the server log data sets. z/OS NFS error and informational messages are also recorded to the z/OS component trace buffer or to the server log data sets. Only ERROR, WARN and INFO go into the log data set, depending on the server startup job setting. This setting may be ERROR, WARN (which includes ERROR), or INFO (which includes all three and is the default). All others are NOT written to the log, but all may be written in the component trace buffer. When recording debug trace diagnostic information, IBM recommends that you set the component trace parameter according to performance tastes; "ALL" will cause the server to run more slowly and the defaults (ERROR + WARN + INFO + FFDC) will run faster. To record z/OS NFS server debug trace diagnostic information in the z/OS component trace buffer, see “Using z/OS component tracing” on page 297. To record z/OS NFS server debug trace diagnostic information in the log data sets, see “z/OS NFS server debug trace capture” on page 525.

The z/OS NFS client records z/OS NFS debug trace diagnostic information to the z/OS component trace buffer. z/OS NFS client error, informational, and warning messages are also recorded to the client log data sets. To record z/OS NFS client debug trace diagnostic information in the z/OS component trace buffer, see “Using z/OS component tracing” on page 297. To record z/OS NFS client error, informational, and warning messages in the log data sets, see “z/OS NFS client debug trace capture” on page 526.

NFS client hang problem analysis

NFS client hang situations can arise from many different causes. Detailed analysis of the situation is required to determine the source. Some of the possible causes could be:

- Slow server or underlying file system response
- Server failure
- Network failure
- Socket hang.

When a client hang occurs, the general UNIX netstat (or TSO netstat or z/OS shell onetstat) command can be used to determine whether this could be a socket hang problem. If the netstat command is issued on the failing client system and it shows that a very large number of TCP sockets are in CLOSEWAIT state with the destination IP address of the z/OS NFS server, it indicates that sockets are probably hung (z/OS NFS server may not accept new TCPIP connections). In this case, it is very probable that the diagnostic trace data recorded by the z/OS NFS Server is already lost by the time that the hang is recognized at the client. The only thing that can be done to resolve the immediate situation is to restart the z/OS NFS server.

Once the z/OS NFS server is restarted, it is recommended that the **MODIFY sockhang** command is used to help capture the necessary diagnostic data before it is lost the next time that this situation occurs. This command tells the server to monitor the server's sockets for potential hang conditions and to create a dump when a hang is suspected so that the diagnostic trace data can be captured before it is lost. For details on the **MODIFY sockhang** command, see "Entering operands of the modify command for diagnosis" on page 233.

For other possible causes of a client hang, standard problem analysis techniques should be used.

Environmental checklist

This environmental checklist covers useful information that is recommended prior to the initializing the NFS server.

Dispatching Priority

Ensure that z/OS NFS has lower dispatching priority than TCP/IP. Also ensure that TCP/IP has a lower dispatching priority than VTAM®.

TCP/IP

Ensure that the MTU (Packet Size) is set to the lowest MTU when in a heterogeneous network. For example, if the network is comprised of:

```
Ethernet 802.3 (MTU=1492)
Ethernet Version 2 IEEE (MTU=1500)
token ring (MTU=2000)
FDDI (MTU=4000)
CTC (MTU=65527)
CLAW (MTU=4096)
```

In this example, the lowest MTU is set to 1492 to reduce packet fragmentation. The MTU setting is defined in the TCP/IP profile.

If there are changes to network addressing and the z/OS NFS server is running in NODHCP mode, the following sequence must be performed to correctly process the changes within the NFS server:

1. Unmount all client mounts
2. Clear the mount handle database (MHDB)
3. Stop the z/OS NFS server
4. Make the necessary changes in the network
5. Restart the z/OS NFS server.

NFS

Verify that NFS is fully initialized.

- RPCBIND or Portmapper is up. The z/OS Portmapper does not support IPv6. Therefore, when using IPv6 addresses, the z/OS server host must be configured with RPCBIND, not the Portmapper. RPCBIND supports both IPv6 and IPv4. As of z/OS V1R8, Portmapper should only be used for IPv4 only systems. Otherwise, RPCBIND should be used.

Note: If a hostname is defined with a primary IPv6 address, and a secondary IPv4 address, the z/OS NFS Client will use the IPv6 address for accessing the host. The z/OS NFS Client will only select the IPv4 address if instructed to do so with the "rpcbind(n)" attribute. Because the z/OS Portmapper does not support IPv6, the z/OS server host must be configured with RPCBIND, not the Portmapper.

- NFS has obtained port 2049 for NFS program such as *rcpinfo -p <hostname>*.
- NFS has z/OS UNIX SEGMENT with UID=0 defined. The NFS server also needs OPERATIONS RACF authority.

For more information, see "Configuring the z/OS NFS server" on page 171 and "Configuring the z/OS NFS client" on page 167.

Useful Utilities

The following utilities are available on client machines to help diagnose simple network connection problems:

- *rcpinfo -p <hostname>* to determine if RPCBIND or Portmapper (port 111), NFS (port 2049) are initialized with the appropriate port.
- *ping* to confirm that there is a live TCP/IP connection between client/server machines.
- *traceroute <hostname>* to determine how packets are being routed from client to server.
- *iptrace* (AIX) or *snoop* (Solaris) are useful packet tracing utilities used to debug inbound and outbound packets between client and server. For example, during a mount request, using *iptrace* or *snoop* will show whether the client transmitted the mount request, the server has received the request, or the server is still processing the request.

Chapter 19. Network File System messages

This topic lists messages from the NFS server, the NFS client, and the client operating system.

Table 60 lists the contents of this topic.

Table 60. Messages - client operating system, NFS server, and NFS client

Section	Page
Server messages	319
Client messages	381
Client/Server Shared Messages	402
Messages from the client platform (AIX)	412

Server messages

This is a listing of the messages generated by the NFS server. Each message description gives an explanation and recommended actions where applicable. The system substitutes data for any part of a message shown here in *italics*.

Messages appear in the NFS server log data set in the same format as this example: **19:25:14 GFSA348I (I) GFSAMAIN ANMAI 02 NFS_INIT: z/OS NETWORK FILE SYSTEM SERVER (HDZ11VS) STARTED.**

Table 61 on page 319 shows the message format for the NFS server log data set:

Table 61. Message format for the NFS server log data set

19:25:14	The time stamp (*hours*:*minutes*:*seconds*)
GFSA	The component identifier for the NFS server
348	A unique message number
I	The message type: **A** Action; the user must perform a specific action. **E** Eventual action; the user must perform an action when time is available. **I** Informational; no user action is required.
(I)	The message level: E (error), W (attention), or I (informational). The system programmer can use the message level to determine which type of messages are shown by specifying log=error, log=warn, or log=info.
GFSAMAIN ANMAI 02 NFS_INIT:	Programming support information
z/OS NETWORK FILE SYSTEM SERVER (HDZ11VS) STARTED.	The message text

© Copyright IBM Corp. 1991, 2010

The messages are listed in numerical order (the time stamp, message level, and programming support information are not shown).

1. A value of *h_digits* is a hexadecimal number, and *d_digits* is a decimal number. A value of *text* or *dsname* is variable text (such as a data set name).
2. Messages GFSA300I through GFSA319I are intended for IBM support personnel when they are performing diagnosis. They do not indicate a problem with NFS but do provide statistics on NFS processing. As such, an extensive number of GFSA300I through GFSA319I messages may be issued.
3. For messages written to the console, the name of the start procedure is substituted for *procname*.

GFSA300I - GFSA319I

Explanation: These messages are intended for IBM support personnel when they are performing diagnosis. They do not indicate a problem with NFS but do provide statistics on NFS processing. As such, an extensive number of GFSA300I through GFSA319I messages may be issued.

System action: NFS continues processing.

Operator response: None.

System programmer response: Turn debugging off.

GFSA320I (*procname*) NETWORK FILE SYSTEM SERVER INITIALIZATION FAILED: *text*

Explanation: In the message text:

procname The name of the start procedure.

text The value of *text* can be any of these messages:

- VIRTUAL STORAGE IS UNAVAILABLE
- MAPPING SIDE FILE NOT FOUND
- MAPPING SIDE FILE HAS INVALID SYNTAX OR FORMAT
- ERROR OPENING/READING MAPPING SIDE FILE
- SIDE FILE SPECIFIED BUT MAPPING IS DIS-ALLOWED BY INSTALLATION
- TASK IS NOT APF AUTHORIZED
- DFP LEVEL MUST BE DFSMS 1.2 OR HIGHER
- SERVER ALREADY STARTED

System action: The NFS server startup ends.

Operator response: Notify the system programmer.

System programmer response: The system programmer response follows:

- If it is a virtual storage problem, increase the region size.
- If it is an APF-authorization problem, APF authorize all libraries in the STEPLIB DD statement.
- If the mapping side file is not found, make sure that the name specified in the attribute data set is correct and the file exists.
- If the mapping side file has invalid syntax, check the part GFSAPMAP in SYS1.NFSSAMP library for mapping side file rules. Refer to the server log data sets for message GFSA432I, which identifies the record number containing the syntax error. If GFSA432I is not issued, check that the last two qualifiers of the side file data set name are NFS.MAPPING.
- If an error occurs during the opening of the side file, check to make sure that the side file is not migrated and that it is readable.
- If **sfmax=0**, then the side file cannot be specified in the attribute data set.
- If you are using a down-level release of DFSMSdfp, restart the NFS server after installing DFSMSdfp 1.2 or a later release.
- If the same start procedure is used with the same TCP/IP stack name, use a different name for either the start procedure or the TCP/IP stack name.

GFSA321I (*procname*) NETWORK FILE SYSTEM SERVER INITIALIZATION FAILED: z/OS UNIX ADDRESS SPACE HAS NOT BEEN STARTED.

Explanation: The Network File System server was not able to establish successful communications with the UNIX address space.

In the message text:

procname The name of the start procedure.

System action: The NFS server startup ends.

Operator response: Before starting the NFS server, both the UNIX and the TCP/IP address spaces must have been successfully started.

GFSA322I (*procname*) z/OS UNIX V_REG FAILED: RV=1, RC=*h_digit1*, RSN=*h_digit2*.

Explanation: The NFS server failed to register.

In the message text:

procname The name of the start procedure.

System action: The NFS server ends.

Operator response: Contact the system programmer.

System programmer response: The values of *h_digit1* and *h_digit2* are the return code and reason code from the z/OS UNIX V_REG callable service. See *z/OS UNIX System Services File System Interface Reference* for more information about return code and reason codes.

GFSA323I (*procname***) z/OS UNIX FID=***fid1* **for path** *pathname* **is different from MHDB FID=***fid2***; unable to restore mount handle for client** *clienthostname***.**

Explanation: The internal FileID *fid1* returned by z/OS UNIX for path name pathname is different from the fileID *fid2* indicated in the MHDB record for the same path name. This can be due to a filesystem presented by z/OS UNIX having reordered its inode numbers when moving from one z/OS release level to another, or unmounting a filesystem and mounting a different filesystem to the same z/OS UNIX pathname while the NFS server was not running.

In the message text:

procname The name of the start procedure.

fid1 The file hande in another z/OS mount handle data base.

pathname The pathname of the mount object.

fid2 The file hande in the current z/OS mount handle data base.

clienthostname The client host name, or IP address, that mounted the object.

System action: NFS continues processing.

Operator response: Contact the system programmer.

System programmer response: Any clients mounted to the affected mount points must have those mount points remounted.

GFSA324I (*procname***) MVS FHDB records from FMID** *fmid* **are not tolerated in this release; mount point** *pathname* **for client** *clienthostname* **must be remounted.**

Explanation: The Filehandle Database from FMID *fmid* contains mount records for MVS data sets that cannot be used in the current release. FHDB records that cannot be used will be listed to the console with one or more GFSA324I messages. If multiple GFSA342I messages are issued to the console, message GFSA907I may be issued to indicate that additional GFSA324I messages are in the NFS log, if the logging level permits.

In the message text:

procname The name of the start procedure.

fmid The FMID of the server that generated the FHDB records being processed.

pathname The pathname of the mount object.

clienthostname The client host name, or IP address, that mounted the object.

System action: NFS continues processing.

Operator response: Contact the system programmer.

System programmer response: Any clients mounted to the affected mount points must have those mount points remounted.

GFSA326E NETWORK FILE SYSTEM SERVER INITIALIZATION FAILED: CANNOT BIND *text1* **PORT** *d_digit* **FOR PROGRAM** *text2* **return code=***h_digit1* **reason code=***h_digit2* **.**

Explanation: The NFS server was not able to bind port *d_digi*t of port type *text1* for program name *text2*, and failed with return code *h_digit1* and reason code *h_digit2*. For definitions of the return code and reason code values, see *z/OS UNIX System Services Messages and Codes*.

System action: The NFS server ends.

Operator response: Notify your system programmer.

System programmer response: The possible failing reasons can be:

- The port d_digitis not available. Program *text2* is either specified in /etc/services with a duplicate port *text1*, or is reserved in the range of the NFS server defined in TCPIP.PROFILE but port *text1* does not match program *text2* in /etc/services.
- The port *d_digit* is in use. Issue '/bin/netstat -P *d_digit*' on the z/OS UNIX shell to find out the status of port *d_digit*. Correct the situation and retry.

GFSA328I (*procname***) NO SWAP REQUEST FOR NETWORK FILE SYSTEM SERVER FAILED.**

Explanation: The z/OS NO SWAP request for NFS failed.

In the message text:

procname The name of the start procedure.

System action: NFS server ends.

Operator response: Contact the system programmer.

System programmer response: Try to restart the NFS server. If the failure appears to be an NFS error, contact the IBM Support Center. Have available a symptom string and a copy of the z/OS console log.

GFSA329I (*procname*) SERVER SHUTDOWN IN PROGRESS.

Explanation: Shutdown procedures have started.

In the message text:

procname The name of the start procedure.

System action: The shutdown of the NFS server continues.

GFSA330I (*procname*) SERVER SHUTDOWN COMPLETE.

Explanation: The NFS server and its associated subtasks ends.

In the message text:

procname The name of the start procedure.

System action: NFS ends.

GFSA331E (*procname*) RECALL FAILED FOR MIGRATED DATA SET *dsname*.

Explanation: DFSMShsm was unable to recall a data set, *dsname*, because the data movement program, DFSMSdss, detected during a restore that the migrated data set had internal errors.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

Operator response: Notify the system programmer for recovery actions.

System programmer response: The data set *dsname* had an internal error when it was migrated and cannot be recalled. See the DFSMShsm message ARC0075E for appropriate recovery actions.

GFSA332I (*procname*) ERRlogdata( ) - invalid length *digit1* from *text1* *text2*.

Explanation: The ERRlog function is called with a data length*digit1* greater than the allowed maximum of 2048 bytes. The call was from function*text2* in module *text1*.

System action: The trace record is skipped and NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Provide this information to IBM Support.

GFSA333I (*procname*) z/OS NETWORK FILE SYSTEM SERVER *fmid* IS NOT STARTED BY A START COMMAND.

Explanation: The system programmer tried to start the NFS server with a command other than **start**.

In the message text:

procname The name of the start procedure.

fmid The FMID of the NFS server; for example, null or (HDZ11VS).

System action: NFS stops.

System programmer response: Start the NFS server by issuing a **start** command.

GFSA334E (*procname*) MOUNT HANDLE DATABASE CANNOT BE READ.

Explanation: The NFS server attempted to read the mount handle database but could not do so. This might occur because the mount handle database could not be opened or because it contained a record with an incorrect length. If migration coexistence APAR OA11612 is installed, pre-HDZ11TS mount handle databases are not supported.

In the message text:

procname The name of the start procedure.

System action: The NFS Server ends.

Operator response: Contact the system programmer.

System programmer response: If the mount handle database could not be opened, check that the record length of the mount handle database is 1600 for HDZ11TS and 2000 for HDZ11US and later releases. If the record length of the mount handle database is correct, then rebuild the mount handle database. If migration coexistence APAR OA11612 is installed and a pre-HDZ11TS mount handle database is being used, then rebuild the mount handle database.

GFSA335E (*procname*) MOUNT HANDLE DATA SET CANNOT BE WRITTEN, EXPECTING LEN *d_digits1* REAL LEN *d_digits2*, VSAM R15(DEC) *d_digits3* REASON CODE(DEC) *d_digits4* LAST OP(DEC) *d_digits5*.

Explanation: The NFS server attempted to write a mount record to the mount handle database but could not do so. The length of the mount record is *d_digits1* bytes, but only *d_digits2* bytes were written. The failing information in writing the virtual storage access method (VSAM) key-sequenced data set (KSDS) mount handle database is the decimal return code *d_digits3*, the decimal error code or reason code *d_digits4*, and the code for the last operation *d_digits5*.

In the message text:

procname The name of the start procedure.

System action: System processing continues but in a degraded mode. Additional mounts or unmounts might fail in writing the record to the mount handle data set.

Operator response: Contact the system programmer.

System programmer response: The mount handle data sets have probably become unusable and need to be either cleared or restored to some previous level, and then the server has to be restarted. The information for the mount point was not saved in the mount handle data set. Clients might have to unmount and mount these mounted directories when the server is restarted.

GFSA335I MOUNT HANDLE DATA SET CANNOT BE WRITTEN, EXPECTING LEN *d_digits1* **REAL LEN** *d_digits2***, VSAM R15(DEC)** *d_digits3* **REASON CODE(DEC)** *d_digits4* **LAST OP(DEC)** *d_digits5***.**

Explanation: The NFS server tried to write a mount record to the mount handle database but could not do so. The length of the mount record is *d_digits1* bytes, but only *d_digits2* bytes were written. The failing information in writing the VSAM KSDS mount handle database is the decimal return code *d_digits3*, the decimal error code or reason code *d_digits4*, and the code for the last operation *d_digits5*.

System action: System processing continues but in a degraded mode. Additional mounts or unmounts might fail in writing the record to the mount handle data set.

Operator response: Contact the system programmer.

System programmer response: The mount handle data sets have probably become unusable and need to be either cleared or restored to some previous level, and then the server has to be restarted. Clients might have to unmount and mount any previously mounted directories.

GFSA336E (*procname***) MOUNT HANDLE DATA SET CANNOT BE OPENED, VSAM R15(DEC)** *d_digits1* **REASON CODE(DEC)** *d_digits2* **LAST OP(DEC)** *d_digits3***.**

Explanation: During a resource timeout, the NFS server tried to open the mount handle data set for writing but could not do so. The failing information in **fopen** for the VSAM KSDS mount handle data set is the decimal return code *d_digits1*, the decimal error code or reason code *d_digits2*, and the code for the last operation *d_digits3*.

In the message text:

procname The name of the start procedure.

System action: NFS stops.

Operator response: Contact the system programmer.

System programmer response: The mount handle data sets probably were created with incorrect attributes.

GFSA337E (*procname***) Network File System Server CTRACE initialization failed because macro** *text1* **had return code=***digit1***, reason code=***digit2***. Diagnostic information is being recorded in error log data sets only.**

Explanation: The Network File System server was not able to obtain services needed for NFS component tracing. This message is received during the Network File System server initialization or resulting from a TRACE CT,ON,COMP=MVSNFS command.

In the message text:

procname
The name of the start procedure.

Text1 Name of system service call which failed.

Digit1 Return code returned from the service.

Digit2 Reason code returned from the service.

System action: The Network File System server continues. Debug trace messages, if any, will be recorded to the error log data sets only.

Operator response: Notify the system programmer.

System programmer response: Return and reason codes for system services such as DSPSERV, ALESERV and CTRACE are documented in *z/OS MVS Programming: Assembler Services Reference ABE-HSP* and *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*. Restart the NFS server to enable NFS component tracing after correcting the error or contact IBM support.

GFSA339I (*procname***) Network File System** *text1* **RESTRICTION FOR DATA SET** *text2***; ACCESS BLOCKED TO** *text3* **DATA SETS.**

Explanation: NFS encountered a data set (*text2*) which is a type of data set that is not supported at the NFS release level.

In the message text:

procname
The name of the start procedure.

Text1 NFS release level.

Text2 Name of the unsupported data set.

Text3 Type of data set.

System action: NFS processing terminates.

Operator response: Notify the system operator.

System programmer response: Remove the data set and retry NFS.

GFSA340I (*procname*) Mvslogin is replacing userid *text1* with userid *text2* for client UID: *d_digits*, IP: *text3*

Explanation: During MVSLOGIN, the existing MVS userid, *text1*, having RACF authorization has been replaced with the new MVS userid, *text2*, having RACF authorization for the client with UNIX UID *d_digits* on the host system with IP address *text3*.

In the message text:

procname
The name of the start procedure.

Text1 The existing MVS userid.

Text2 The new MVS userid.

d_digits
UNIX user identification (UID).

Text3 IP address of the host system.

System action: NFS processing continues with the new MVS userid authority.

Operator response: None.

System programmer response: None.

GFSA344I (*procname*) MOUNT HANDLE DATABASE CANNOT BE READ.

Explanation: The NFS server attempted to read the mount handle database but could not do so. This might occur because the mount handle database contained a record with an incorrect length or incorrect contents.

In the message text:

procname The name of the start procedure.

System action: System processing continues but in a degraded mode. Some or all the directories have not been remounted.

Operator response: Contact the system programmer.

System programmer response: If the mount handle database contains a record with an incorrect length or incorrect contents, then rebuild the mount handle data sets.

GFSA345E VSAM DATA SET IS NOT REUSABLE.

Explanation: At least one VSAM data set (FHDBASE, FHDBASE2, LDBASE, or LDBASE2 DD statements in the z/OS NFS Server job) is not defined with the REUSE option.

System action: NFS server processing stops.

Operator response: Contact the system programmer.

System programmer response: Recreate the appropriate VSAM data sets using IDCAMS with the REUSE option, and restart the NFS server.

GFSA346I *time_stamp*.

Explanation: Displays the current time stamp. This message is issued when the NFSLOG switches.

GFSA347I ERROR RETURNED TO CLIENT: RC = *d_digits text*.

Explanation: The error code *d_digits* was returned to the client.

In the message text:

text The value of *text* is the meaning of the error code.

System action: NFS processing continues.

GFSA348I (*procname*) z/OS NETWORK FILE SYSTEM SERVER *fmid* STARTED.

Explanation: The NFS is initialized and ready to accept **modify** commands from the operator console.

In the message text:

procname The name of the start procedure.

fmid The FMID of the NFS server; for example, null or (HDZ11VS).

System action: NFS continues processing.

GFSA349I UNEXPECTED ERROR DETECTED: *d_digits text*.

Explanation: The NFS server encountered a condition that indicates continued processing might produce undesirable results.

In the message text:

text The value of *text* is additional debugging information for programming support personnel.

System action: NFS either shuts down or stops the request and continues processing, depending on where the error was detected.

System programmer response: Contact your programming support personnel.

GFSA352E Lock Data Set *dsname* for the DDNAME=*ddname* failed op *opname* with rc=*rtncode*

Explanation: An attempt to read, write, or modify the lock data set has failed. This messages identifies the data set, the operation, and the failure code.

In the message text:

dsname
The name of the failing LDB data set.

ddname
The ddname of the failing data set.

opname
The name of the operation that failed.

rtncode
The failure code from the operation.

System action: This lock data set will not be used. NFS continues, using the other lock data set. If both lock data sets have failed, lock information will not be recorded and reclaim permissions on the next NFS restart may be inaccurate.

Operator response: Notify the system programmer.

System programmer response: The lock data set has probably been corrupted. Plan to delete the data set and reallocate it.

GFSA360I *text***.**

Explanation: This message displays memory management statistics, *text*.

System action: NFS continues processing.

GFSA361I NETWORK FILE SYSTEM IS SHORT ON STORAGE.

Explanation: This message displays on the operator console when a shortage of virtual storage is detected.

System action: NFS processing continues. The storage constraint might be relieved when some storage is freed up later on.

Operator response: If this message is displayed repeatedly within a short period of time, stop or cancel NFS and notify the system programmer.

System programmer response: Take either or both of the following actions before restarting the NFS server.

- Increase the region size for the step or started task.
- Decrease the value specified for the **bufhigh** attribute of the attributes data set.

GFSA362I (*procname***) REGION SIZE WILL NOT ACCOMMODATE BUFHIGH AND LOGICAL CACHE SPECIFICATIONS.**

Explanation: The values of **bufhigh** and **logicalcache** specified in the attributes data set are incorrect.

In the message text:

procname The name of the start procedure.

System action: NFS ends.

System programmer response: Either increase the region size of the step, or reduce the total value of the **bufhigh** and **logicalcache** attribute values.

GFSA363I (*procname***) NETWORK FILE SYSTEM SERVER IS SHORT OF BUFFERS.**

Explanation: NFS has no memory blocks in the buffer area limited by the **bufhigh** attribute for data buffers.

In the message text:

procname The name of the start procedure.

System action: None

Operator response: Notify the system programmer.

System programmer response: Increase the **bufhigh** value in the NFS attribute profile.

GFSA364I (*procname***) Not enough storage below 16Mb for requested number of Legacy Tasks.**

Explanation: The size of available storage is not enough for the requested number of legacy Tasks.

In the message text:

procname The name of the start procedure.

System action: The NFS server is shut down.

Operator response: Notify the system programmer.

System programmer response: Reduce the number of legacy tasks in the site attribute data set, using the **nfstasks** attribute.

GFSA365I Routine *text1***( ) for** *text2* **failed return code=** *digit1* **errno=***digit2* **errno2=***digit3***.**

Explanation: Function call *text1* failed for data set *text2* with return code *digit1*. The *digit2* and *x_digit3* values specify the errno and errno2 error codes. For explanations of these codes, see *z/OS UNIX System Services Messages and Codes*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA366I Routine *text1***( ) for data set** *text2* **failed, return code=** *digit1***.**

Explanation: Function call *text1* failed for data set *text2* with return code *digit1*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA367I No correct label, NFS.MAPPING and NFS.MAPPING.MAPPED

Explanation: The current mapping side file does not contain labels: NFS.MAPPING or NFS.MAPPING.MAPPED. See "File extension mapping" on page 40 in *z/OS Network File System Guide and Reference* for details on the use of these labels in the mapping side file.

System action: The current mount request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Check and correct the current mapping side file.

GFSA368I Catalog locate for *text1* failed because file is migrated.

Explanation: The *text1* data set is not available because it is migrated.

System action: The current request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Recall the migrated data set. Then retry the request.

GFSA369I Catalog locate for *text1* failed because file does not exist in catalog.

Explanation: The *text1* data set is not available because it is migrated.

System action: The current request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Either create the requested data set, or change the request to specify a different data set.

GFSA370I Catalog locate for *text1* failed with return code = *digit1*.

Explanation: The *text1* data set is not available. The Catalog Locate function (SVC 26) failed with return code *digit1*. See the Catalog Services documentation for details on the return code.

System action: The current request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Either correct the data set access problem, or change the request to specify a different data set.

GFSA371I The *text1* Mapping Side File name is missing an entry for string *text2*.

Explanation: The mapping side file named *text1* does not have an entry for *text2*.

System action: The current mount request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Check and correct the mapping side file and retry the mount request.

GFSA372I Invalid *text1* dataset, LRECL = *digit1* and BLKSIZE = *digit2*, expected LRECL = 256 and BLKSIZE = multiple of 256.

Explanation: The *text1* data set has an invalid format. The Logical Record Length and/or Block Size are not the required values. The LRECL must be 256 and the BLKSIZE must be a multiple of 256.

System action: The current request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Create an data set with the required format and retry the request.

GFSA375I Operation *text1* failed expected length=*digit1* real length=*digit2* VSAM R15(dec)=*digit3* Reason code(dec)=*digit4* RBA(dec)=*digit5* Last op(dec)=*digit6* errno=*digit7* errno2=*digit8* .

Explanation: During shutdown, the NFS Server encountered an error trying to perform operation *text1* on the Mount Handle Data Base. The cited VSAM error codes were received.

System action: The failing Mount Handle Data Base record is ignored. NFS Shutdown processing continues.

Operator response: Contact the system programmer.

System programmer response: Analyze the cited VSAM error codes and take the appropriate corrective action. If necessary, contact IBM Support.

GFSA376I Operation *text1* failed VSAM R15(dec)=*digit1* Reason code(dec)=*digit2* RBA(dec)=*digit3* Last op(dec)=*digit4* .

Explanation: . During UNMOUNT request processing, the NFS Server encountered an error trying to perform operation *text1* on the Mount Handle Data Base. The cited VSAM error codes were received

System action: The Mount Handle Data Base record is not removed. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Analyze the cited

VSAM error codes and take the appropriate corrective action. If necessary, contact IBM Support.

GFSA377I **Operation** *text1* **failed expected length=***digit1* **real length=***digit2* **VSAM R15(dec)=***digit3* **Reason code(dec)=***digit4* **RBA(dec)=***digit5* **Last op(dec)=***digit6* **.**

Explanation: During timeout processing, the NFS Server encounered an error trying to perform operation *text1* on the Mount Handle Data Base. The cited VSAM error codes were received.

System action: The Mount Handle Data Base record is ignored and NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Analyze the cited VSAM error codes and take the appropriate corrective action. If necessary, contact IBM Support.

GFSA378I **Routine** *text1 text1***failed errno=***digit1* **errno2=***digit2* **.**

Explanation: System Function call *text1* failed for data set *text2* with errno=*digit1*, errno2=*digit2*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Analyze the cited VSAM error codes and take the appropriate corrective action. If necessary, contact IBM Support.

GFSA379I **MAPPING SIDE FILE NOT FOUND .**

Explanation: The Mapping Side File could not be read. The preceding GFSA368I, GFSA369I, or GFSA370I message should provide more details on the reason for the failure.

System action: The current mount request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Take the specified action for the associated message.

GFSA380I **MAPPING SIDE FILE HAS INVALID SYNTAX OR FORMAT .**

Explanation: The current Mapping Side file contains a syntax or format error. See preceding messages for more details.

System action: The current mount request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Check the Mapping Side File for any errors and correct them.

GFSA381I **ERROR OPENING/READING MAPPING SIDE FILE.**

Explanation: An error was encountered attempting to open/read the Mapping Side File.

System action: The current mount request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Check the Mapping Side File for any data set errors and correct them. If necessary, contact IBM Support.

GFSA382I **STORAGE LIMIT REACHED LOADING MAPPING SIDE FILE.**

Explanation: The NFS server ran out of memory attempting to read the mapping side file.

System action: The current mount request will fail. Otherwise, NFS processing attempts to continue. It may not be possible to continue due to the out-of-memory situation.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA383I **(***mvsnfs***) Network File System Server data buffer utilization has reached** *yyy***% of** *xxxxxx* **MB.**

Explanation: Buffer utilization has reached the alerting threshold of bufhigh z/OS NFS Server processing attribute. The message prints current data buffer usage in percent of the limit from z/OS NFS Server bufhigh attribute (*yyy* value) and a value of the limit in Mbytes (*xxxxxx* value).

mvsnfs the Network File System Server start up procedure name

yyy the percent of MVS data buffers used in reference to the limit specified by *xxxxxx*.

xxxxxx the maximum MVS internal data buffer pool size (value of attribute bufhigh)

System action: None.

Operator response: None.

System programmer response: Inform the application administrators on the remote NFS client to possibly decrease the incoming workload or change z/OS NFS Server Bufhigh or Percentsteal attributes.

GFSA384I **(***mvsnfs***) Resource shortage for Network File System Server data buffers has been relieved.**

Explanation: Data buffer utilization decreased from a high value to the value below the alerting threshold of the Bufhigh z/OS NFS Server processing attribute

mvsnfs the Network File System Server start up procedure name

System action: None.

Operator response: None.

System programmer response: None.

GFSA385I (*mvsnfs*) Current data buffer utilization is *xxxx* Mb.

Explanation: As a response to the **MODIFY** *nfsserver*, **BUFFERUSAGE** command, this message displays the current data buffer storage allocation.

mvsnfs the Network File System Server start up procedure name

xxxx the current MVS data buffer storage allocation in megabytes.

System action: None.

Operator response: None.

System programmer response: None.

GFSA386E (*mvsnfs*) Command error: *ERROR DESCRIPTION*.

Explanation: There is an error in the command. It can be a command syntax error or an invalid value specified.

ERROR DESCRIPTION can be one of the following:

- "wrong syntax"
- "the first argument is out of range"
- "the second argument is out of range"
- "invalid or omitted value"

mvsnfs the Network File System Server start up procedure name

System action: None.

Operator response: None.

System programmer response: Use correct command syntax and valid values.

GFSA387I (*mvsnfs*) The current attribute settings for bufhigh are (*XX* Mb, *YY* %).

Explanation: A response to the successful completion of the **MODIFY nfsserver, bufhigh** and **MODIFY nfsserver, bufhigh=(*xx,yy*)** commands. This message displays actual z/OS NFS Server Bufhigh attribute values.

mvsnfs the Network File System Server start up procedure name

XX the current Bufhigh data buffer pool size in megabytes.

YY the Bufhigh alert percentage.

System action: None.

Operator response: None.

System programmer response: None.

GFSA400I INVALID RECFM SPECIFICATION (*text*).

Explanation: The value of *text* is the incorrect record format that was specified in the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA401I (*procname*) CANNOT OPEN THE ATTRIBUTE DATA SET.

Explanation: The server was unable to open the attributes data set defined in the JCL for DDNAME NFSATTR. The DD statement might be missing or the data set name might be incorrect.

In the message text:

procname The name of the start procedure.

System action: NFS stops.

Operator response: Notify the system programmer.

System programmer response: Correct the JCL for DDNAME NFSATTR.

GFSA402I (*procname*) READ FAILED FOR THE ATTRIBUTES DATA SET.

Explanation: An error occurred while NFS was processing the attribute data set. This message follows other messages that describe the error in greater detail. The attributes data set is defined in the JCL for DDNAME NFSATTR.

In the message text:

procname The name of the start procedure.

System action: NFS stops.

Operator response: Notify the system programmer.

System programmer response: Correct the attributes data set.

Note: When the NFS attribute data set is created, the *num off* option in ISPF should also be used. The sequence number is not allowed in the NFS attributes data set.

GFSA403I (*procname*) PARSE FAILED IN LINE *d_digits text*.

Explanation: The parsing of line number *d_digits* in the attribute data set failed.

In the message text:

procname
The name of the start procedure.

text
The value of *text* is the actual line from the attributes data set that contains the failure.

This message follows other messages that describe the error in greater detail.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA404I UNEXPECTED END OF STRING ON END OF PARSE IN LINE *d_digits*.

Explanation: A comma is missing between attributes on line number *d_digits* of the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA405I PARSE FAILED FOR ATTRIBUTE FIELD - ILLEGAL KEYWORD IN LINE *d_digits*.

Explanation: The keyword specified in line number *d_digits* of the attribute data set is not a valid attribute keyword.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA406I MISSING LEFT PARENTHESIS IN LINE *d_digits*.

Explanation: An attribute specified on line number *d_digits* of the attributes data set is missing a left parenthesis.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA407I MISSING RIGHT PARENTHESIS IN LINE *d_digits*.

Explanation: An attribute specified on line number *d_digits* of the attributes data set is missing a right parenthesis.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA408I PARSE FAILED ON NUMBER FIELD IN LINE *d_digits*.

Explanation: An attribute with a negative number was specified on line number *d_digits* of the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA409I INVALID DSORG SPECIFICATION IN LINE *d_digits*.

Explanation: The data set organization specified in the **dsorg** attribute on line number *d_digits* of the attributes data set is incorrect or is not supported by NFS.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA411I INVALID VOL SPECIFICATION IN LINE *d_digits*.

Explanation: The volume specified in the **vol** (volume) attribute on line number *d_digits* of the attributes data set is incorrect.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA412I INVALID UNIT SPECIFICATION IN LINE *d_digits*.

Explanation: The unit specified in the unit attribute on line number *d_digits* of the attributes data set is incorrect.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA413I THE ATTRIBUTE VALUE *d_digits1* IS NOT IN THE RANGE OF *d_digits2* TO *d_digits3*.

Explanation: The value of *d_digits1*, specified in one of the attributes of the attributes data set, must be between the minimum value, *d_digits2*, and the maximum value, *d_digits3*, for this attribute. Message GFSA403I follows this message.

System action: NFS stops.

System programmer response: See message GFSA403I to determine the attribute in error, and then correct the attributes data set.

GFSA414I THE ATTRIBUTE VALUE *d_digits1* EXCEEDS THE MAXIMUM TIMEOUT VALUE OF *d_digits2*.

Explanation: In the message text:

d_digits1 The value of *d_digits1*, specified in one of the attributes of the attributes data set, must be less than or equal to *d_digits2*.

d_digits2 The value of *d_digits2* is the maximum value allowed for the attribute.

Message GFSA403I follows this message.

System action: NFS stops.

System programmer response: See message GFSA403I to determine the attribute in error, and then correct the attributes data set.

GFSA415I THE ATTRIBUTE TIME OUT VALUE *d_digits1* IS LESS THAN THE MINIMUM TIME OUT VALUE *d_digits2*.

Explanation: The value of *d_digits1*, specified in the **attrtimeout** attribute in the attributes data set, must be greater than or equal to the value *d_digits2*, which is specified in the **mintimeout** attribute of the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA416I THE READ TIME OUT VALUE *d_digits1* IS LESS THAN THE MINIMUM TIME OUT VALUE *d_digits2*.

Explanation: The value of *d_digits1*, specified in the **readtimeout** attribute of the attributes data set, must be greater than or equal to the value *d_digits2*, which was specified in the **mintimeout** attribute of the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA417I THE WRITE TIME OUT VALUE *d_digits1* IS LESS THAN THE MINIMUM TIME OUT VALUE *d_digits2*.

Explanation: The value of *d_digits1*, specified in the **writetimeout** attribute of the attributes data set, must be greater than or equal to the value *d_digits2*, which was specified in the **mintimeout** attribute of the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA418I THE ATTRIBUTE TIME OUT VALUE *d_digits1* IS GREATER THAN THE MAXIMUM TIME OUT VALUE *d_digits2*.

Explanation: The value of *d_digits1*, specified in the **attrtimeout** attribute of the attributes data set, must be less than or equal to the value *d_digits2*, which was specified in the **maxtimeout** attribute of the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA419I THE READ TIME OUT VALUE *d_digits1* IS GREATER THAN THE MAXIMUM TIME OUT VALUE *d_digits2*.

Explanation: The value of *d_digits1*, specified in the **readtimeout** attribute of the attributes data set, must be less than or equal to the value *d_digits2*, which was specified in the **maxtimeout** attribute of the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA420I THE WRITE TIME OUT VALUE *d_digits1* IS GREATER THAN THE MAXIMUM TIME OUT VALUE *d_digits2*.

Explanation: The value of *d_digits1*, specified in the **writetimeout** attribute of the attributes data set, must be less than or equal to the value *d_digits2*, which was specified in the **maxtimeout** attribute of the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA421I THE NOATTRTIMEOUT ATTRIBUTE WAS SPECIFIED BUT THE MAXTIMEOUT VALUE WAS SET TO *d_digits*.

Explanation: The **noattrtimeout** attribute means that the data set is not to be deallocated following a **lookup** or **getattr** operation. The **maxtimeout** attribute specifies the maximum timeout value allowed for any of the timeout attributes. The value of *d_digits* was specified as the **maxtimeout** value, in seconds, that the data set is to remain allocated. These attributes are in conflict. The **noattrtimeout** and **maxtimeout** attributes are specified in the attributes data set.

System action: NFS stops.

System programmer response: If you want to use the **noattrtimeout** attribute, specify the **nomaxtimeout** attribute in the attributes data set. Correct the attributes data set.

GFSA422I THE NOREADTIMEOUT ATTRIBUTE WAS SPECIFIED BUT THE MAXTIMEOUT VALUE WAS SET TO *d_digits*.

Explanation: The **noreadtimeout** attribute means that the data set is not to be deallocated following a read operation. The **maxtimeout** attribute specifies the maximum timeout value allowed for any of the timeout attributes. The value of *d_digits* was specified as the value of the **maxtimeout** attribute, in seconds, that the data set is to remain allocated. These attributes are in conflict. The **noreadtimeout** and **maxtimeout** attributes are specified in the attributes data set.

System action: NFS stops.

System programmer response: If you want to use the **noreadtimeout** attribute, specify the **nomaxtimeout** attribute in the attributes data set. Correct the attributes data set.

GFSA423I THE NOWRITETIMEOUT ATTRIBUTE WAS SPECIFIED BUT THE MAXTIMEOUT VALUE WAS SET TO *d_digits*.

Explanation: The **nowritetimeout** attribute means that the data set is not to be deallocated following a write operation. The **maxtimeout** attribute specifies the maximum timeout value allowed for any of the timeout attributes. The value of *d_digits* was specified as the value of the **maxtimeout** attribute, in seconds, that the data set is to remain allocated. These attributes are in conflict. The **nowritetimeout** and **maxtimeout** attributes are specified in the attributes data set.

System action: NFS stops.

System programmer response: If you want to use the **nowritetimeout** attribute, specify the **nomaxtimeout** attribute in the attributes data set. Correct the attributes data set.

GFSA424I MINIMUM TIME OUT VALUE, *d_digits1*, IS GREATER THAN THE MAXIMUM TIME OUT VALUE, *d_digits2*.

Explanation: The value of *d_digits1*, specified in the **mintimeout** attribute of the attributes data set is greater than the value *d_digits2*, specified in the **maxtimeout** attribute of the attributes data set.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA425I INVALID SPECIFICATION RECFM(*text*).

Explanation: One of the characters in the value of *text* specified in the **recfm** data set creation attribute of the attributes data set is incorrect.

System action: The NFS stops.

System programmer response: Correct the attributes data set.

GFSA426I INVALID RECFM(*text*) - MUST SPECIFY U, F, OR V.

Explanation: One of the characters in the value of *text* specified in the **recfm** data set creation attribute of the attributes data set must define whether the record is fixed length (F), variable length (V), or undefined (U).

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA429I INVALID DSNTYPE SPECIFICATION IN LINE *d_digits*.

Explanation: The data set type specified in the **dsntype** attribute on line number *d_digits* of the attributes data set is incorrect.

System action: NFS stops.

System programmer response: Correct the attributes data set.

GFSA430I INVALID *SMS_keyword* SPECIFICATION IN LINE *d_digits*.

Explanation: The SMS keyword *SMS_keyword* is syntactically incorrect on line number *d_digits*. See system-managed storage documentation for DATACLAS, MGMTCLAS, and STORCLAS naming conventions.

System action: NFS startup ends if the keyword was specified as a site attribute. If the incorrect SMS keyword was specified by a client as a mount parameter or in a command, the line number is set to zero and an I/O error is returned to the client.

System programmer response: Correct the site attributes file, if applicable.

GFSA431I INVALID OPTION SPECIFICATION IN LINE *d_digits*.

Explanation: The option specified in the value on line number *d_digits_* of the attributes data set is incorrect.

System action: The current NFS request fails. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Correct the site attributes file, if applicable.

GFSA432I Incorrect text in MAPPING SIDE FILE entry *digit1*

Explanation: An syntax error was detected processing record *digit1* in the current mapping side file.

System action: The current mount request will fail. Otherwise, NFS processing continues. If the syntax error was detected during z/OS NFS Server initialization, z/OS NFS Server initialization will fail.

Operator response: Contact the system programmer.

System programmer response: Correct any errors in the record and then reissue the mount request or restart the z/OS NFS Server.

GFSA433 Pathname parse error *text1*

Explanation: During startup processing, the NFS server detected an invalid z/OS UNIX File System name (*text1*) in the Mount Handle Data Base. The file system either no longer exists, or was modified locally while the NFS Server was down.

System action: The specified Mount Handle Data Base entry will be skipped and NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: If the file system no longer exists or was modified while the NFS server was down, ignore this message. If no changes were made and there are no other explanations for this error, keep the existing z/OS NFS server traces and contact IBM Support.

GFSA434I (*procname*) NFSTASKS(*value*) is set to the default value, NFSTASKS(*default_values*).

Explanation: In the message text:

procname	The name of the start procedure.
value	Value specified on NFSTASKS keyword
default_values	Current default values for NFSTASKS keyword

Only one value was specified for the NFSTASKS keyword, but a minimum of two values is required. NFS has set NFSTASKS to its current set of default values. See the description of the nfstasks site attribute in Table 24 on page 134 for a listing of the current defaults.

System action: NFS processing continues.

GFSA435E For *attrname* Site Attribute, Sum of N_SUBTASKS (*n_value*) and *parmname* (mo_value) must be less than or equal to *max_value*.

Explanation:

attrname	MINTASKS or NFSTASKS
n_value	value specified for N_SUBTASKS
parmname	M_SUBTASKS or O_SUBTASKS
mo_value	value specified for M_SUBTASKS or O_SUBTASKS
max_value	Maximum allowed sum

The sum of the value specified for N_SUBTASKS *n_value* plus the value specified for M_SUBTASKS or O_SUBTASKS *mo_value* is greater than the allowed amount *max_value*. See the descriptions of the **mintasks** or **nfstasks** site attributes and their *n*, *m*, and *o* values in Table 24 on page 134 for more information.

System action: NFS processing stops.

System programmer response: Correct the values.

GFSA436I INVALID SIDEFILE SPECIFICATION IN LINE *linenum*

Explanation: A data set name was not specified in the **sidefile** attribute.

System action: NFS server startup ends.

System programmer response: Correct the problem and make the necessary changes in the attributes data set.

GFSA437I INVALID PUBLIC SPECIFICATION IN LINE *d_digits*

Explanation: Parsing of the **public** keyword resulted in an error for one of the following reasons:

- **public** keyword specification is syntactically incorrect
- No public path names have been specified
- The path name specified is not valid
- The HFS public path name does not match the HFS prefix

System action: NFS server startup ends.

Operator response: Correct the problem and make the necessary changes in the attributes data set.

GFSA438I EXPORT SPANNING PATHNAMES NOT SUPPORTED

Explanation: The export-spanning path names support for a multicomponent **lookup** request is not supported.

System action: The request fails. NFS processing continues.

User response: Construct a different path name in which the path is not spanned.

GFSA439I z/OS UNIX PUBLIC PATHNAME SPECIFIED BUT z/OS UNIX IS NOT ENABLED

Explanation: A z/OS UNIX public path name was specified for the **public** keyword but **nohfs** was also specified, which disables z/OS UNIX processing.

System action: NFS server startup ends.

Operator response: Correct the problem and make the necessary changes in the installation table.

GFSA440I INVALID SECURITY SPECIFICATION IN *d_digits*

Explanation: Parsing of the **security** keyword resulted in an error for one of the following reasons.

- Missing first parameter
- First parameter not valid

System action: NFS server startup ends.

Operator response: Correct the problem and make the necessary changes in the attributes data set.

GFSA441E (*procname***) Invalid leasetime value. It must be smaller than logout value.**

Explanation: The specified Network File System lock lease time value is larger than the logout timeout value. The leasetime value must be smaller that the logout timeout value.

System action: The Network File System server initialization fails and the server terminates.

Operator response: None.

System programmer response: Check and correct the relationship between the leasetime and logout values specified in the Site Attribute File.

GFSA442I SIDE FILE SPECIFIED BUT MAPPING IS DIS-ALLOWED BY INSTALLATION.

Explanation: A Mapping Side File was specified on a Mount command but mapping is disallowed by the NFS server site attribute settings.

System action: The current mount request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Either change the NFS site attribute settings or remove the Mapping Side File specification from the Mount request.

GFSA443I Missing parameter in MODULE INFO entry.

Explanation: During startup processing, the NFS server detected incorrect internal module statistics.

System action: NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA444E Invalid IMPPREFIX specification in line *d_digit***.**

Explanation: The IMPPREFIX site attribute specified on line *d_digit* in the site attribute file contains an invalid value.

System action: The Network File System server initialization fails and the server terminates.

Operator response: Contact the system programmer.

System programmer response: Check and correct the IMPPREFIX value specified in the Site Attribute File and then restart the NFS server.

GFSA445E Site Attribute IMPPREFIX(NONE) is set. Required path prefix is missing in the path name: *text***.**

Explanation: The IMPPREFIX site attribute indicates that all path names must be specified with an explicit hfs or mvs prefix, but this path name was specified without a prefix. The path type (z/OS UNIX or MVS) cannot be determined. In the message text:

- *text* - The path name string that is in error.

System action: The request fails. NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Change the IMPPREFIX value specified in the Site Attribute file, or add an explicit prefix to the path name.

GFSA446E Invalid path name: *text***.**

Explanation: The specified path name is not valid. Possible causes for this error include the following:

- The name does not start with a prefix and the path name contains invalid characters for the path type determined from the implicit path type heuristic specified in the IMPPREFIX site attribute.

- The name was determined to be an MVS data set name, either due to the presence of an MVS prefix or based on the implicit path type heuristic specified in the IMPPREFIX site attribute, but the name is too long for a valid MVS data set name.

In the message text:

- *text* - The path name string that is in error.

System action: The request fails. NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Either correct the name, change the IMPPREFIX value specified in the Site Attribute file, or add an explicit prefix to the path name.

GFSA450I CREATED TASK(*h_digits*) - *text1* - *text2*.

Explanation: The NFS is creating the number of tasks requested in the **nfstasks** attribute of the attributes data set. This message displays for each task created.

In the message text:

h_digits
The value of *h_digits* is the TCB address.

text1
The value of *text1* is the task name.

text2
The value of *text2* is the module name.

System action: NFS processing continues.

GFSA451I DELETING TASK(*h_digits*) - *text*.

Explanation: The NFS is deleting a task. This is in response to the **stop** operand of the **modify** command. This message is displayed for each task deleted.

In the message text:

h_digits
The value of *h_digits* is the TCB address.

text
The value of *text* is the module name.

System action: NFS shutdown continues.

GFSA452I SUBTASK TERMINATED: *h_digits*.

Explanation: NFS is stopping a task. This is in response to the **stop** operand of the **modify** command. This message is displayed for each ended task.

In the message text:

h_digits
The value of *h_digits* is the task control block (TCB) address.

System action: NFS shutdown continues.

GFSA453E (*procname*) NETWORK FILE SYSTEM SERVER LOST TASK *tasktype*. NOW AVAILABLE: *nn1* LEGACY TASKS, *mm1* HFS TASKS, *oo1* LONG SERVICE TASKS. LEGACY TASKS SHUTDOWN LIMIT IS < *nn2*. HFS TASKS SHUTDOWN LIMIT IS <*mm2*. LONG SERVICE TASKS SHUTDOWN LIMIT IS < *oo2*.

Explanation: This message means that one *tasktype* task was detached due to an ABEND 878 . If the count of the available tasks of one type is less than the shutdown limit for the task, the NFS server restarts the suspended tasks.

In the message text:

procname
The name of the start procedure.

tasktype
This value indicates the type of tasks: `“legacy”`, `“HFS”`, or `“Long service”` tasks.

nn1
This value indicates the number of available legacy tasks (for z/OS conventional MVS data sets).

mm1
This value indicates the number of HFS tasks.

oo1
This value indicates the number of long service tasks.

nn2
This value indicates the limit of legacy tasks (for z/OS conventional MVS data sets).

mm2
This value indicates the limit of HFS tasks.

oo2
This value indicates the limit of long service tasks.

System action: The task ends. The NFS server restarts the suspended tasks.

User response: None

GFSA454I (*procname*) *taskname* TASK HAS BEEN RECOVERED AFTER 878 ABEND. ALL TASKS WILL BE STARTED NOW

Explanation: The NFS task had an ABEND 878 (80A) and then was recovered. The NFS server is ready to restart task processing.

In the message text:

procname
The name of the start procedure.

taskname
Specifies the name of the NFS task that had the ABEND 878 (80A).

System action: NFS server restarts task processing.

User response: None

GFSA455E (*procname*)Connection failed from TCPIP, shutting down. Errno(*digit1*) errno2(*digit2*).

Explanation: The Network File System was unable to connect to TCPIP The attempted connection returned error code *digit1* and reason code *digit2*.

System action: The Network File System server initialization fails and the server terminates.

Operator response: None.

System programmer response: Check the return and reason codes. If these indicate a system configuration error, correct it. Otherwise, submit this error to IBM Support.

GFSA456E (*procname*) Socket creation failed, errno(*digit1*) errno2(*digit2*).

Explanation: The Network File System was unable to create a communication socket. The attempted connection returned error code *digit1* and reason code *digit2*.

System action: The Network File System server initialization fails and the server terminates.

Operator response: None.

System programmer response: Check the return and reason codes. If these indicate a system configuration error, correct it. Otherwise, submit this error to IBM Support.

GFSA459E (*procname*) ioctl: cannot set socket to blocking/non-blocking mode, errno(*digit1*) errno2(*digit2*).

Explanation: The Network File System was unable to set the required socket blocking mode. The z/OS UNIX ioctl function returned error code *digit1* and reason code *digit2*.See *z/OS XL C/C++ Run-Time Library Reference* for details on the ioctl function.).

System action: The request require this mode fails.

Operator response: None.

System programmer response: Check the return and reason codes. If these indicate a system configuration error, correct it. Otherwise, submit this error to IBM Support.

GFSA460E (*procname*) send/sendto request failed, errno(*digit1*) errno2(*digit2*).

Explanation: The Network File System server was unable to send a request to unregister functions in the portmapper before registering this new instance of the server. z/OS UNIX returned error code *digit1* and reason code *digit2*.

System action: The Network File System server initialization fails and the server terminates.

Operator response: None.

System programmer response: Check the return and reason codes. If these indicate a system configuration error, correct it. Otherwise, submit this error to IBM Support.

GFSA461E (*procname*) Min *d_digits* subtask threshold is reached.

Explanation: The number of NFS Server subtasks has reached the minimum value.

In the message text:

procname The name of the start procedure.

System action: NFS stops.

User response: None

Operator response: Contact the system programmer.

System programmer response: Check mintasks and nfstasks in the NFS server attribute data set, and reduce this number if possible.

GFSA462E (*procname*) MVS delegation disabled. Return code=0x*h_digit1*.

Explanation: An error occured when the NFS Server was registering ENF 51 to Listen For System Events. Return code, *h_digit*, is documented for the ENFREQ in *z/OS MVS Authorized Assembler Services Reference*. The capability of delegation for MVS data set is disabled.

System action: The Network File System server continues processing

User response: None

Operator response: Contact the system programmer.

System programmer response: Correct the problem if it is possible; otherwise, keep the existing z/OS NFS server traces and contact IBM Support Center.

GFSA463E (*procname*) MVS delegation disabled due to short of storage.

Explanation: There is not enough memory to support NFS version 4 file management delegation. The capability of delegation for MVS data set is disabled.

System action: The Network File System server continues processing

User response: None

Operator response: None.

System programmer response: None.

GFSA464I (*procname*) MVS delegation was disabled.

Explanation: NFS version 4 file management delegation for MVS data sets is disabled by the Modify operator subcommand, V4DELG=OFF.

System action: The Network File System server continues processing

User response: None

Operator response: None.

System programmer response: None.

GFSA465I (*procname*) MVS delegation was enabled.

Explanation: NFS v4 MVS data set delegation has been enabled by the operator subcommand, V4DELG=ON.

System action: The Network File System server continues processing

User response: None

Operator response: None.

System programmer response: None.

GFSA469I (*procname*) *taskname*(*tcbaddr*) ABEND=*abendcode*,REASON=*reasoncode*

Explanation: A subtask (thread or TCB) of the z/OS NFS server abnormally terminates when the Recovery Termination Manager (RTM) does not allow the ESTAEX exit to retry.

In the message text:

procname
The name of the Network File System Server startup procedure.

taskname
Eight-character name of the z/OS NFS server task that terminated.

tcbaddr Address of the TCB corresponding to the task that terminated.

abendcode
Abend code indicating the type of abend.

reasoncode
Reason code (if any) associated with this abend.

System action: NFS Server processing terminates.

User response: None

Operator response: Contact the system programmer..

System programmer response: If the NFS server was not terminated as part of a deliberate operator action, then collect the dumps and the NFS Server Joblog datasets prior to contacting the IBM Support Center. The z/OS NFS Server ESTAEX exit does not internally generate a dump for the system abend code of x22, x37, x3E, 806, A03.

For these Abend codes the SLIP command is required to capture the necessary dump.

GFSA470I (*procname*) NETWORK FILE SYSTEM SERVER SVC DUMP REQUEST FAILED. REASON=*reason_code*.

Explanation: A request to write a z/OS SVC dump failed.

In the message text:

procname The name of the start procedure.

reason_code The value of *reason_code* is a hexadecimal number indicating the reason that z/OS was unable to write the dump. See the description of the **sdump** macro in *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU* for the meaning of the reason code.

System action: ABEND processing continues.

Operator response: If the dump could not be written due to an operational procedure and a dump is necessary to diagnose the failure, correct the procedure.

System programmer response: If the failure appears to be due to an NFS error, contact the IBM Support Center. Have available a symptom string and a copy of the z/OS console log.

GFSA471I (*procname*) NETWORK FILE SYSTEM SERVER ESTAE EXIT UNABLE TO COMPLETE PROCESSING. ABEND=*abend_code*.

Explanation: The NFS ESTAE exit routine that ended abnormally and is unable to complete processing.

In the message text:

procname The name of the start procedure.

abend_code The *abend_code* value is set to the last abend code encountered by a secondary instance of the ESTAE exit routine.

System action: ESTAE processing is stopped. The NFS address space might end also.

Operator response: Restart the NFS address space. Notify network users of the failure.

System programmer response: Contact the IBM Support Center. Have available a symptom string, the related SYS1.LOGREC entries, any related SDUMPs, and the NFS error trace log.

GFSA472I (*procname*) 878 ABEND HANDLING FOR TASK *taskname* IN PROCESS. ALL TASKS STOPPED. PLEASE WAIT FOR RECOVERY.

Explanation: The NFS task had an ABEND 878 (80A), and the recovery procedure is in process.

In the message text:

procname
The name of the start procedure.

taskname
Specifies the name of the NFS task that had the ABEND 878 (80A).

System action: The NFS server is in recovery processing and all normal processing is suspended.

User response: None

GFSA473E (*procname*) No connection yet between z/OS UNIX and TCPIP. Waiting for connection. errno(*rtncode*) errno2(*rsncode*).

Explanation: The Network File System was unable to connect to TCP/IP. The attempted connection returned error code *rtncode* and reason code *rsncode*.

In the message text:

procname
The name of the start procedure.

System action: NFS processing waits for TCP/IP connection to be established.

User response: None

Operator response: Contact the system programmer.

System programmer response: Check the error and reason codes. If these indicate a TCP/IP system configuration error, correct it. Otherwise, submit this error to IBM TCP/IP Support.

GFSA474E (*procname*) No connection yet from z/OS Portmapper. Waiting for connection. errno(*errcode*) errno2(*rsncode*).

Explanation: The Network File System was unable to connect to the TCP/IP Portmapper/RPCBIND. The attempted connection returned error code *errcode* and reason code *rsncode*.

System action: NFS processing waits for the TCP/IP Portmapper/RPCBIND connection to be established.

Programmer response: None

Operator response: Contact the system programmer.

System programmer response: Check the error and reason codes. If these indicate a TCP/IP Portmapper/RPCBIND system configuration error, correct it. Otherwise, submit this error to IBM TCP/IP Support.

GFSA475E (*procname*) cannot connect host(*ipaddr*) *hostname* UNIX & TCPIP. port(*port1 port2*) errno(*errcode*) errno2(*rsncode*).

Explanation: The Network File System was unable to connect to client host. *ipaddr* is the client host IP address. *hostname* is the client hostname if hostname exists. The attempted connection from port, *port1*, to client host port, *port2*, returned error code *errcode* and reason code *rsncode*.

System action: The request continues without the ability to callback to the client host. NFS processing continues.

Programmer response: None

Operator response: Contact the system programmer.

System programmer response: Check the error and reason codes. If these indicate a TCP/IP system configuration error, correct it. Otherwise, submit this error to IBM TCP/IP Support.

GFSA476E (*procname*) MVS System Call *service failed*, return code *retc*.

Explanation: Calling MVS System service service failed with return code *retc*.

The service can be:

IEAVAPE
allocate Pause Element

IEAVPSE
PAUSE

IEAVRLS
RELEASE a PAUSE element

The possible return code are documented in *z/OS MVS Programming: Authorized Assembler Services Reference*.

System action: The Network File System server terminates.

User response: None

Operator response: Contact the system programmer.

System programmer response: Collect failure documentation and contact IBM Support Center.

GFSA480I REQUEST FAILED BECAUSE OF ERRORS ENCOUNTERED IN THE CONVERSION SERVICES, RC=*d_digits1*, RSN=*d_digits2*.

Explanation: The request received an error condition from the conversion services while data was being translated.

System action: The request failed. NFS processing continues.

User response: See *z/OS Unicode Services User's Guide* *and Reference* for information about the error encountered, and correct the problem if possible.

GFSA481I REQUEST FAILED BECAUSE TEXT CONVERSION RESULTED IN THE LENGTHS OF INPUT AND OUTPUT STRINGS BEING DIFFERENT.

Explanation: The request failed because the translation of the text string resulted in a different length. Translations that involve length changes (such as SBCS to MBCS) are not currently supported.

System action: The request failed. NFS processing continues.

User response: Make the necessary changes so that text string translations do not result in different lengths.

GFSA482I REQUEST FAILED BECAUSE TRANSLATION OF TEXT IS NOT POSSIBLE.

Explanation: An operation is being requested for a file that is tagged with a valid coded character set identifier (CCSID), but the user has not specified a **cln_ccsid** value to be used for translation, and the **xlat()** keyword is also specified.

System action: The request failed. NFS processing continues.

User response: Either specify a **cln_ccsid** value for translation or remove the specification **xlat()** keyword.

GFSA483I WRITE REQUEST FAILED BECAUSE BINARY DATA CANNOT BE WRITTEN TO A FILE WITH PURE TEXT DATA.

Explanation: A write request has the **binary** option specified explicitly, and the file being written to has pure text data.

System action: The request failed. NFS processing continues.

User response: Either remove the **binary** option or change the file tag that is associated with the file to show that the file has mixed data.

GFSA483E TAG OPTION IS SPECIFIED BUT CONVERSION SERVICES IS NOT ACTIVE.

Explanation: Conversion Services must be active if TAG is specified in the site attribute data set or mount command.

System action: Network File System Server startup ends if TAG is specified in the site attribute data set, or mount point is ignored during MHDB processing, or mount command is rejected.

Operator response: Notify the system programmer.

System programmer response: Either change from TAG to NOTAG in site attribute data set (or mount command) or activate Conversion Services.

GFSA484E TECHNIQUE VALUE OF CONVSERV OPTION IS INVALID.

Explanation: The technique specified for CONVSERV(technique) in the attributes data set or mount command is incorrect. It can consist of up to five characters corresponding to the five available techniques: R, E, C, L and M.

System action: Network File System Server startup ends if CONVSERV(technique) is specified in the attributes data set, or the mount command is rejected.

Operator response: Notify the system programmer.

System programmer response: Update the values for CONVSERV(technique).

GFSA485I (*procname*) CONVERSION SERVICES ARE NOT ACTIVE DURING NETWORK FILE SYSTEM SERVER STARTUP.

Explanation: During startup, the NFS Server detected that Conversion Services were not available.

In the message text:

procname The name of the start procedure.

System action: NFS server processing continues without the availability of Conversion Services.

Operator response: Notify the system programmer.

System programmer response: If data conversion is required, Conversion Services must be available to NFS. See *z/OS Unicode Services User's Guide and Reference*for more information.

GFSA486I (*procname*) CONVERSION SERVICES ARE ACTIVE AND WILL BE USED BY NETWORK FILE SYSTEM SERVER.

Explanation: During RPC MOUNT processing, the NFS Server detected that Conversion Services were available.

In the message text:

procname The name of the start procedure.

System action: NFS server processing continues.

GFSA487E (*procname*) During startup, Network File System Server encountered problems with Conversion Services: Service returned RC =*rtncode*, RSN = *rsncode*.

Explanation: During Network File System server startup, it encountered problems with the Conversion

Services. The conversion service failed.

In the message text:

procname The name of the start procedure.

rtncode The return code from conversion services.

rsncode The reason code from conversion services.

System action: NFS server processing continues.

System programmer response: The conversion services problem must be resolved before the conversion services can be used by the NFS server.

GFSA501I REQUEST HEADER ALLOCATION FAILED.

Explanation: An operation to allocate virtual memory for a request header was tried but was unsuccessful.

System action: The request is stopped. NFS processing continues.

System programmer response: Increase the size of the step region.

GFSA502I REQUEST HEADER DATA BLOCK ALLOCATION FAILED.

Explanation: An operation to allocate virtual memory for a request header data block was tried but was unsuccessful.

System action: The request is stopped. NFS processing continues.

System programmer response: Increase the size of the step region.

GFSA509I UNSUPPORTED *option* **OPTION** *(value)* **has been ignored.**

Explanation: With installed toleration APAR OA14907, the NFS server, during parsing of an HDZ118N exports data set, ignores options which are unsupported by the current release of the NFS server.

In the message text:

option The string “exports” or “checklist”.

value The value of the unsupported option.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the HDZ118N exports data set.

GFSA510I EXPORTS: CAN'T READ /etc/netgroup FILE. NETGROUP *name* **HAS BEEN IGNORED.**

Explanation: The netgroup name *name* in the exports data set has been ignored because NFS cannot read the local netgroup file. See messages GFSA878I and GFSA879I for more information.

In the message text:

name The netgroup name.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Check for validity of the specified netgroup file. Correct the name contents of the local netgroup file.

GFSA511I EXPORTS: FILE /etc/netgroup CONTAINS EMPTY NETGROUP (*text***).**

Explanation: The netgroup name *text* in the exports data set has no data in the local netgroup file.

In the message text:

text The netgroup name.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the local netgroup file.

GFSA512I EXPORTS: CIRCULAR GROUP NAME REFERENCE DETECTED IN /etc/netgroup FILE DURING NETGROUP *text1* **PARSING. NETGROUP** *text2* **HAS BEEN IGNORED.**

Explanation: The netgroup named *text2* in the exports data set has been ignored because NFS detected a circular group name reference when it parsed the nested netgroup named *text1*. NFS checks for cycles which can happen if, for example, a netgroup named @name1 includes a netgroup named @name2 which in turn includes a netgroup named @name1.

In the message text:

text1 The name of the netgroup that is the cause of the cycle.

text2 The starting netgroup name.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the local netgroup file.

GFSA513I EXPORTS: SYNTAX ERROR IN /etc/netgroup FILE IN NETGROUP (*text1*) . NETGROUP (*text2*) HAS BEEN IGNORED.

Explanation: The netgroup named *text2* in the exports data set has been ignored because NFS detected a syntax error when parsing the nested netgroup named *text1*. NFS checks syntax for the starting group and all nested subgroups.

In the message text:

text1 The name of the netgroup with incorrect syntax.

text2 The starting netgroup name.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the local netgroup file.

GFSA514I EXPORTS: NETGROUP (*text*) PARSING ALREADY FAILED, NETGROUP HAS BEEN IGNORED.

Explanation: The netgroup named *text* in the exports data set has been previously parsed unsuccessfully. See messages GFSA512I and GFSA513I for more information.

In the message text:

text The name of the netgroup.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the local netgroup file.

GFSA515I EXPORTS: NETGROUP (*text*) NOT FOUND IN /etc/netgroup FILE AND HAS BEEN IGNORED.

Explanation: The netgroup named *text* in the exports data set has been ignored because NFS did not find a netgroup with this name in the local netgroup file.

In the message text:

text The name of the netgroup.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the local netgroup file.

GFSA516I EXPORTS: NETWORK FILE SYSTEM SERVER IS IPV4-NODE, IPV6 *text1 (text2)* HAS BEEN IGNORED.

Explanation: The network template or IP address specified in *text2* has IPv6 format whereas the NFS Server runs as an IPv4 node. The network template or IP address is ignored.

In the message text:

text1 The item that is in IPv6 format: *network template* or *address*.

text2 The value of the network template or IP address.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the exports data set.

GFSA517I EXPORTS: DHCP IS *text1*, *text2 (text3)* HAS BEEN IGNORED.

Explanation: If the NFS server runs with NODHCP support (DHCP is OFF), it ignores any hostname with wild cards in the exports data set. If the NFS server runs with DHCP support (DHCP is ON) it ignores any network templates or IP addresses in the exports data set.

In the message text:

text1 The DHCP value: *ON* or *OFF*.

text2 The type of item that has been ignored: *network template*, *hostname with wildcards*, or *IP address*.

text3 The value of the network template or hostname template (hostnames with wildcards), or IP address.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the exports data set.

| **GFSA518A CHECKLIST: WRONG *text1 (text2)*, CHECK LIST HAS NOT BEEN CREATED.**

Explanation: NFS detected a syntax error when parsing a directory suffix or full directory name. A full directory name is formed by concatenation of string1, which is before the directory suffix, with string2 which is a subdirectory in the directory suffix. For example: string1 <string2,nosaf>. String2 may be empty, for example: string1 <nosaf>.

In the message text:

text1
The type of item where the syntax error was found:

- *directory suffix*
- *directory name*
- *name* – in this case, the name or its parent directory or subdirectory are already specified in the exports data set.

text2
The value of the directory suffix or full directory name.

System action: The Network File System server initialization fails and the server terminates.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the exports data set.

GFSA520I Operation *text1* **for** *text2* **type** *digit1* **failed** *digit2***.**

Explanation: The NFS server detected an error *digit2* on *text1* operation for DDname *text2*, open mode *digit1*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA521I Operation *text1* **for** *text2* **failed** *digit1***.**

Explanation: The NFS server detected an error *digit1* on *text1* operation for DDname *text2*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA522I Operation *text1* **for** *text2* **failed** *digit1* **reason** *digit2***.**

Explanation: The NFS server detected an error *digit1*, reason code *digit2* on *text1* operation for data set name *text2*, open mode *digit1*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA523I Operation *text1* **for** *text2* **type** *digit1* **return code***digit2* **reason code***digit3***.**

Explanation: The NFS server detected an error *digit2*, reason code *digit3* on *text1* operation for DDname *text2*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA524I VSIO *text1* **failed** *digit1* **for branch** *digit2***.**

Explanation: The NFS server detected an error *digit1* on I/O phase *digit2* for DDname *text1*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA525I VSIO *text1* **failed** *digit1***/***digit2* **for branch** *digit3* **VSAM supplied codes:** *digit4***,***digit5***.**

Explanation: The NFS server detected an error *digit1*, return code *digit2* on I/O phase *digit3* for DDname *text1* for a VSAM data set.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA526I VSIO *text1* **busy** *digit1* **for branch** *digit2* **.**

Explanation: The NFS server detected an error *digit1* on I/O phase *digit2* for DDname *text1*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA527I VSIO for *text1* **VSAM message:** *text2***.**

Explanation: The NFS server detected an error *text2* for DD name *text1*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA550I (*procname***) IP***text* **is active.**

Explanation: The Network File System server found the IP level (*text*) of V6 or V4, with the TCPIP Stack active. The server will use that stack for all its communications.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA554I Routine *text1* **() failed, errno(***digit1***) errno2(***digit2***).**

Explanation: Function call *text1*() failed with an errno value of *digit1* and an errno2 value of *digit2*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA555I Connection failed between z/OS UNIX and TCPIP errno(*digit1***) errno2(***digit2***), shutting down.**

Explanation: Connection between the NFS Server and TCPIP failed with an errno value of *digit1* and an errno2 value of *digit2*. For definitions of the errno and errno2 values, see*z/OS UNIX System Services Messages and Codes*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Check the errno and errno2 value definitions in *z/OS UNIX System Services Messages and Codes* for any possible corrective action that can be taken and proceed accordingly. Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA556I getaddrinfo() failed, error(*digit1***):** *text1***.**

Explanation: Call to function getaddrinfo() failed with an errno value of *digit1*, as described in *text1*.

System action: The NFS server processing continues.

Operator response: Contact the system programmer.

System programmer response: Check the errno value definition in *z/OS UNIX System Services Messages and Codes* for any possible corrective action that can be taken and proceed accordingly. Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA557I Routine *text1* **() failed, errno(***digit1***) -** *text2***.**

Explanation: Call to function *text1* failed with an errno value of *digit1* due to *text2*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Check the errno value definition in *z/OS UNIX System Services Messages and Codes* for any possible corrective action that can be taken and proceed accordingly. Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA558I (*procname***) UNABLE TO CREATE IPC QUEUE.**

Explanation: An operation to allocate virtual memory for an interprocess communication (IPC) queue was tried but was unsuccessful.

In the message text:

procname The name of the start procedure.

System action: NFS stops.

Operator response: Notify your system programmer.

System programmer response: Increase the size of the step region.

GFSA559I (*procname***) CANNOT CREATE UDP SERVICE.**

Explanation: User datagram protocol (UDP) service transport could not be created, or you started TCP/IP before UNIX initialization was complete.

In the message text:

procname The name of the start procedure.

System action: NFS stops.

Operator response: Check your z/OS TCP/IP setup or notify your system programmer. Before starting TCP/IP, make sure that UNIX initialization is complete, and that the TCP/IP UNIX connection is established.

System programmer response: Check your z/OS TCP/IP setup. Also check the UNIX BPXPRMxx parmlib member, specifically MAXFILEPROC, MAXSOCKETS, INADDRANYPORT, and INADDRANYCOUNT. INADDRANYPORT and INADDRANYCOUNT must be specified, but their range cannot include 2049, for the server to initialize. See *z/OS MVS Initialization and Tuning Guide* for more details.

GFSA563I Operation *text1* for socket(*digit1*) failed, errno(*digit2*) errno2(*digit3*).

Explanation: The operation identified by *text1* failed with an errno value of *digit1* and an errno2 value of *digit3*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Check the errno and errno2 value definitions in *z/OS UNIX System Services Messages and Codes* for any possible corrective action that can be taken and proceed accordingly. Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA564I Socket(*digit1*) invalid fragment length. Expected size (*digit2*), real size(*digit3*).

Explanation: A request was received for the socket(*digit1*). The length for the fragment should be *digit2* but (*digit3*) bytes of data were received

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA565I The registration (xid=*digit1*) to the PORTMAPPER failed.

Explanation: The NFS Server PORTMAPPER registration request with xid=*digit1* failed to register NFS information in the PORTMAPPER.

System action: The NFS Server will shut down.

Operator response: Contact the system programmer.

System programmer response: Check the NFS server job log and NFS Log data set for an explanation. If none is found, keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA566I (*procname*) CANNOT ACCEPT NEW TCP CLIENT CONNECTION — MAXIMUM NUMBER OF SOCKETS HAS REACHED

Explanation: When an NFS TCP client attempts to connect to the NFS server, the server cannot accept the connection because the maximum number of sockets is reached.

In the message text:

procname The name of the start procedure.

System action: The connection request fails. NFS processing continues.

Operator response: Increase the value of **maxsockets** for the AF_INET domain in the BPXPRM*xx* parmlib member so that client TCP connections can be accepted by the NFS server. See *z/OS UNIX System Services Planning* for more information.

GFSA567I Client host changed IP address with NODHCP.

Explanation: The NFS Server detected that an NFS Client's host IP address has been changed, but the NFS Server was started with NODHCP mode.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Either change to DHCP mode or correct the NFS Client's IP address.

GFSA595I Time Zone Delta is *digit1* hours, *digit2* minutes.

Explanation: During startup the z/OS NFS Server set its Time Zone as *digit1* hours *digit2* minutes. The NFS Server prints to the log the MVS system time zone value that was specified in the SYS1.PARMLIB(CLOCKxx) member at system IPL. server creates.

System action: NFS processing continues.

Operator response: None.

System programmer response: None.

GFSA596I BUFHIGH is set up : *digit1*.

Explanation: During startup NFS Server sets the BUFHIGH value as *digit1*.

System action: NFS processing continues.

Operator response: None.

System programmer response: None.

GFSA598I Network File System is SHORT of BUFFERS. BUFHIGH is too small for current workload.

Explanation: The NFS server detected that the BUFHIGH Site Attribute value is too small for the current workload.

System action: NFS processing continues. However, server performance may be somewhat degraded because of not being able to keep file data cached.

Operator response: Contact the system programmer.

System programmer response: Increase the BUFHIGH value in the NFS Server Site Attribute file and restart the Server.

GFSA660I Incorrect build info event was received.

Explanation: The NFS server detected an error in the load module statistics during startup.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Check that the latest NFS Server maintenance install completed successfully. Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA665I ***text1*** **failed.**

Explanation: XDR type *text1* decoding of a request, or XDR type *text1* encoding of a response, failed due to invalid data.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA666I ***text1*** **encoding error.**

Explanation: XDR type *text1* encoding of a response failed due to invalid data.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA667I Program ***digit1*** **Version(*digit2*) Procedure(*digit3*) not supported.**

Explanation: The NFS server received a request for Program *digit1*, Version *digit2*, Procedure *digit3*. This combination is not supported by NFS. The request is rejected.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Investigate and correct the source of this request. If necessary, keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA668I ***text1*** **Version(*digit1*) Procedure(*digit2*) not supported.**

Explanation: The NFS server received a request for the *text1* Protocol, Version *digit1*, Procedure *digit2*. This combination is not supported by NFS. The request is rejected.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Investigate and correct the source of this request. If necessary, keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA669I XDR operation ***text1*** **(*digit1*) failed.**

Explanation: XDR decoding of request *text1* (operation : *digit1*), or XDR encoding of response request *text1* (operation : *digit1*), failed due to invalid data.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA670I Tag=*digit1* xdr_status=*digit2* or minor=*digit3* or numops=*digit4* failed.

Explanation: XDR decoding of NFS version 4 request tag, minor version or the number of operations in the compound request failed due to invalid data. The tag pointer is *digit1*, XDR status value is *digit2*, minor version is *digit3* and number of operations is*digit4*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA671I Minor version ***digit1*** **not zero.**

Explanation: The NFS server received an NFS version 4 protocol compound request for a minor version other than version 0. The z/OS NFS Server supports only minor version 0 at this time.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Investigate and correct the source of the request. If necessary, keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA672I Illegal operation code *digit1*.

Explanation: The NFS server received an NFS version 4 protocol compound request containing an invalid operation code: *digit1*. This operation code is not supported by the protocol.

System action: As defined in the NFS version 4 protocol, the z/OS NFS server will attempt to process all of the operations before to one in error. At that point the compound request will fail and no further operations in the request will be processed. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Investigate and correct the source of the request. If necessary, keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA673I *text1* (*digit1*) staled GLOBAL(*digit2*) STATEID(*digit3*).

Explanation: The NFS server received an NFS version 4 protocol received an NFS version 4 protocol request. XDR decoding of the request *text1*(operation : *digit1*) has detected that the request is attempting to use a stale Stateid (the base value is *digit3*). The current NFS Server Stateid base value is *digit2*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: None. No action should be required. According to the NFS version 4 protocol, the NFS client is supposed to automatically recover from this error situation.

GFSA674I Not supported operation code(*digit1*) type(*digit2*).

Explanation: The NFS server received an NFS version 4 protocol compound request containing an invalid operation code: *digit1*, type: *digit1*. This operation code is not supported.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA675I Bad file handle length *digit1*.

Explanation: NFS Server received an NFS version 4 protocol compound request containing a PUTFH operation with an invalid file handle length: *digit1*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA677I *text1* decoding error.

Explanation: The NFS server detected an XDR error decoding *text1* for a received request.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA678I Character conversion error.

Explanation: The NFS server detected a character conversion error while processing a request.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA679I *text1* (*digit1*) Version(*digit2*) not equal to 1.

Explanation: The NFS server received a request for protocol *text1* (Program number = *digit1*), version *digit2*. The z/OS NFS Server only supports version 1 of this protocol.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Investigate and correct the source of the request. If necessary, keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA680I *text1* **decoding error, errno(***digit1***) errno2(***x_digit2***).**

Explanation: The NFS server detected an error attempting to decode a *text1*. It received the following error codes: errno=*digit1*, errno2=*x_digit2*. See the *z/OS UNIX System Services Messages and Codes* for definitions of the errno/errno2 values.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA681I **XDR of attribute** *text1* **failed, status** *digit1***.**

Explanation: While processing an NFS version 4 protocol response, XDR encoding/decoding of Attribute number *text1* (Owner or Owner Group) failed. The XDR error status is *digit1*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA682I **XDR of attribute mask failed, status(***digit1***) or length(***digit2***) wrong.**

Explanation: While processing an NFS version 4 protocol response, XDR encoding of the Supported Attribute mask failed. Either the XDR received an error status (*digit1*) or the mask length (*digit2*) is invalid (length < 1 or length > 2).

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA683I **XDR length=***digit1* **supported bitmap_attr=***digit2* **failed.**

Explanation: While processing an NFS version 4 protocol response, XDR encoding of the Supported Attribute mask experienced an error. XDR of either the mask length (*digit1*) or the mask (*digit2*) failed.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA684I **XDR setclientid() failed xdr_status=***digit1* **v4_status=***digit2***.**

Explanation: While processing an NFS version 4 protocol SetClientId operation response, XDR encoding of the response experienced an error. The XDR status is *digit1* and the NFS version 4 status is *digit2*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA685I **Client requested invalid READ attribute in bitmap_attr=***digit1***.**

Explanation: The NFS server received an NFS version 4 protocol request. The request attempted to read unsupported attributes. The request attribute bitmap is *digit1*.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA686I **Mount(***digit1***) Version(***digit2***) not 1 or 3.**

Explanation: The NFS server received a Mount Protocol (program number *digit1*) Version (*digit2*) request. Only versions 1 and 3 of the Mount protocol are supported by the z/OS NFS server.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Investigate and correct the source of the request. If necessary, keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA687I *text1* **Version(***digit1***) not supported.**

Explanation: The NFS server received a *text1* Protocol Version (*digit1*) request. This Protocol/Version combination is not supported by the z/OS NFS Server.

System action: The current NFS request will fail.

Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Investigate and correct the source of the request. If necessary, keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA727E NETWORK FILE SYSTEM SERVER KERBEROS DLL LOAD FAILED: *d_digits*

Explanation: Dynamic Linked Library loading of the Kerberos runtime environment has failed during the NFS server startup. This message is created when NFS is configured to require Kerberos (such that system authentication (authsys) is not allowed).

In the message text:

d_digits
The value of errno after the call to the krb5_dll_load API.

System action: NFS server processing ends.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information about this failure.

GFSA728E (*procname*) SAF APPLICATION USER MAPPING FAILED WITH SAF RETURN CODE *d_digits1*, RACF RETURN CODE *d_digits2*, and RACF REASON CODE *d_digits3*.

Explanation: NFS encountered a problem in determining the RACF user ID associated with the NFS Client's Kerberos Principal.

In the message text:

d_digits1
The value of *d_digits1* is the SAF return code.

d_digits2
The value of *d_digits2* is the RACF return code.

d_digits3
The value of *d_digits3* is the RACF reason code.

System action: The NFS Client's request is failed with an RPC reply status of MSG_ACCEPTED and an accepted status of SYSTEM_ERR.

Operator response: Ensure that the NFS client principal is defined to RACF. Refer to R_usermap information in *z/OS Security Server RACF Callable Services* for a full explanation of the SAF Return code, RACF return code and RACF reason code.

System programmer response: Refer to R_usermap information in *z/OS Security Server RACF Callable Services* for a full explanation of the SAF Return code, RACF return code and RACF reason code.

GFSA729E (*procname*) USERNAME *text1* NOT FOUND IN SAF DATABASE

Explanation: NFS encountered a problem in determining the z/OS UNIX uid/gid for this RACF user.

In the message text:

procname
The name of the start procedure.

text1
The RACF User ID for which the z/OS UNIX uid/gid could not be obtained.

System action: The NFS Client's request is failed with an RPC reply status of MSG_ACCEPTED and an accepted status of SYSTEM_ERR.

Operator response: Ensure that the NFS Client Principal is correctly defined.

GFSA730I (*procname*) NETWORK FILE SYSTEM SERVER KERBEROS INITIALIZATION SUCCESSFUL

Explanation: The NFS server's Kerberos initialization was successful.

In the message text:

procname
The name of the start procedure.

System action: NFS server processing continues.

GFSA731I NETWORK FILE SYSTEM SERVER COULD NOT LOAD KERBEROS DLL: *d_digits*

Explanation: Dynamic Linked Library loading of the Kerberos runtime environment could not be completed during the NFS server startup. This message is issued when NFS is configured to also allow system authentication (AUTH_SYS).

If only AUTH_SYS authentication is specified as a site attribute (sys) then this message is issued to the NFS server log only (and not to the console). If AUTH_SYS authentication (sys) and any kerberos flavors (KRB5 or KRB5I or KRB5P) are specified as site attributes then this message is issued to the console as well as the NFS server log.

In the message text:

d_digits
The value of the errno after the call to the krb5_dll_load API.

System action: NFS processing continues such that only requests with system authentication (sys) will be supported.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information.

GFSA732E NETWORK FILE SYSTEM SERVER KERBEROS CONTEXT CREATION FAILED: *d_digits*

Explanation: The creation of a Kerberos Context and its initialization with default values obtained from the Kerberos configuration file during the server startup has failed.

In the message text:

d_digits The value of the Kerberos error code after the call to the API krb5_init_context().

System action: The NFS server is brought down.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information about this failure.

GFSA733I NETWORK FILE SYSTEM SERVER COULD NOT CREATE KERBEROS CONTEXT: *d_digits*

Explanation: The creation of a Kerberos Context and its initialization with default values obtained from the Kerberos configuration file during the server startup could not be completed.

If only AUTH_SYS authentication is specified as a site attribute (sys) then this message is issued to the NFS server log only (and not to the console). If AUTH_SYS authentication (sys) and any kerberos flavors (KRB5 or KRB5I or KRB5P) are specified as site attributes then this message is issued to the console as well as the NFS server log.

In the message text:

d_digits The value of the Kerberos error code after the call to the API krb5_init_context().

System action: NFS processing continues such that only requests with system authentication (sys) will be supported.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information.

GFSA734E NETWORK FILE SYSTEM SERVER CREDENTIAL CACHE MANAGEMENT FAILED IN ROUTINE *procName***(), KERBEROS RETURN CODE(***krbRc***) GSS MAJOR STATUS(***majorStat***) GSS MINOR STATUS(***minorStat***)**

Explanation: There was a failure in the NFS server's Kerberos credential cache management. This message is issued when NFS is configured to require Kerberos (that is, system authentication (AUTH_SYS) is not allowed).

In the message text:

procName The name of the failing API. For details, refer to *z/OS Integrated Security Services Network Authentication Service Programming*.

krbRc The error code returned by the Kerberos Security Mechanism. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

majorStat The major status returned by the Generic Security Services. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

minorStat The minor status returned by the Generic Security Services. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

System action: NFS server processing ends.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information about this failure.

GFSA735I NETWORK FILE SYSTEM SERVER COULD NOT SETUP CREDENTIAL CACHE MANAGEMENT IN ROUTINE *procName***(), KERBEROS RETURN CODE(***krbRc***) GSS MAJOR STATUS(***majorStat***) GSS MINOR STATUS(***minorStat***)**

Explanation: There was a failure in the NFS server's Kerberos credential cache management. This message is issued when NFS is configured to also allow system authentication (AUTH_SYS).

If only AUTH_SYS authentication is specified as a site attribute (sys) then this message is issued to the NFS server log only (and not to the console). If AUTH_SYS authentication (sys) and any kerberos flavors (KRB5 or KRB5I or KRB5P) are specified as site attributes then this message is issued to the console as well as the NFS server log.

In the message text:

procName The name of the failing API. For details, refer to *z/OS Integrated Security Services Network Authentication Service Programming*.

krbRc The error code returned by the

Kerberos Security Mechanism. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

majorStat
The major status returned by the Generic Security Services. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

minorStat
The minor status returned by the Generic Security Services. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

System action: NFS server processing continues such that only requests with system authentication(sys) will be supported.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information about this failure.

GFSA736E NETWORK FILE SYSTEM SERVER KERBEROS TICKET ACQUISITION FAILED IN ROUTINE *procName***(), KERBEROS RETURN CODE(***krbRc***)**

Explanation: There was a failure in the NFS server's acquisition of Kerberos Ticket. This message is issued when NFS is configured to require Kerberos (that is, system authentication (AUTH_SYS) is not allowed).

In the message text:

procName
The name of the failing API. For details, refer to *z/OS Integrated Security Services Network Authentication Service Programming*.

krbRc
The error code returned by the Kerberos Security Mechanism. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

System action: NFS server processing ends.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information about this failure.

GFSA737I NETWORK FILE SYSTEM SERVER COULD NOT GET KERBEROS TICKET IN ROUTINE *procName***(), KERBEROS RETURN CODE(***krbRc***)**

Explanation: There was a failure in the NFS server's acquisition of Kerberos Ticket. This message is issued when NFS is configured to also allow system authentication (AUTH_SYS).

If only AUTH_SYS authentication is specified as a site attribute (sys) then this message is issued to the NFS server log only (and not to the console). If AUTH_SYS authentication (sys) and any kerberos flavors (KRB5 or KRB5I or KRB5P) are specified as site attributes then this message is issued to the console as well as the NFS server log.

In the message text:

procName
The name of the failing API. For details, refer to *z/OS Integrated Security Services Network Authentication Service Programming*.

krbRc
The error code returned by the Kerberos Security Mechanism. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

System action: NFS server processing continues such that only requests with system authentication (sys) will be supported.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information about this failure.

GFSA738E NETWORK FILE SYSTEM SERVER GSS CREDENTIAL ACQUISITION FAILED IN ROUTINE *procName***(), GSS MAJOR STATUS(***majorStat***) GSS MINOR STATUS(***minorStat***)**

Explanation: There was a failure in the NFS server's acquisition of GSS credentials. This message is issued when NFS is configured to require Kerberos (that is, system authentication (AUTH_SYS) is not allowed).

In the message text:

procName
The name of the failing API. For details, refer to *z/OS Integrated Security Services Network Authentication Service Programming*.

majorStat
The major status returned by the Generic Security Services. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

minorStat
The minor status returned by the Generic Security Services. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

System action: NFS server processing ends.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information about this failure.

GFSA739I NETWORK FILE SYSTEM SERVER COULD NOT ACQUIRE GSS CREDENTIALS IN ROUTINE *procName*(), GSS MAJOR STATUS(*majorStat*) GSS MINOR STATUS(*minorStat*)

Explanation: There was a failure in the NFS server's acquisition of GSS credentials. This message is issued when NFS is configured to also allow system authentication (AUTH_SYS).

If only AUTH_SYS authentication is specified as a site attribute (sys) then this message is issued to the NFS server log only (and not to the console). If AUTH_SYS authentication (sys) and any kerberos flavors (KRB5 or KRB5I or KRB5P) are specified as site attributes then this message is issued to the console as well as the NFS server log.

In the message text:

procName
The name of the failing API. For details, refer to *z/OS Integrated Security Services Network Authentication Service Programming*.

majorStat
The major status returned by the Generic Security Services. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

minorStat
The minor status returned by the Generic Security Services. For details, refer to *z/OS Integrated Security Services Network Authentication Service Administration*.

System action: NFS server processing continues such that only requests with system authentication (sys) will be supported.

Operator response: Contact the system programmer.

System programmer response: Refer to *z/OS Integrated Security Services Network Authentication Service Programming* for more information about this failure.

GFSA740W (*procname*) Routine *rtn*() could not retrieve the stack names, errno(*errno*) errno2(*errno2*).

Explanation: This message is issued when w_ioctl fails to retrieve the stack names. In the message text:

rtn
Name of the failing routine.

errno
errno set by the failing routine.

errno2
errno2 set by the failing routine.

System action: NFS processing continues with the limitation that only the TCP/IP stack of the local host will be processing the RPCSEC requests.

Programmer response: None

Operator response: None

System programmer response: Ensure that the TCP/IP stacks are configured correctly.

GFSA741W (*procname*) Configured TCP Stacks(*d_digit1*) is more than the number allowed(*d_digit2*).

Explanation: This message is issued when the maximum number of TCP/IP stacks configured on a single NFS server is greater than the maximum allowed. In the message text:

d_digit1
Number of TCP/IP stacks that are currently configured.

d_digit2
Maximum number of stacks allowed on a single NFS server.

System action: NFS processing continues with the limitation that only the number of stacks represented by *d_digit2* will be supported.

Programmer response: None

Operator response: Contact the system programmer.

System programmer response: Ensure that only up to the *d_digit2* TCP/IP stacks are configured to the NFS server.

GFSA742W (*procname*) Could not get *hostname* for *stackname*, return value(*h_digit1*), return code(*h_digit2*), reason code(*h_digit3*).

Explanation: The message is issued when the *hostname* can not be retrieved for a TCP/IP stack. In the message text:

stackname
The name of TCP/IP stack for which the *hostname* could not be retrieved.

h_digit1
The returned value returned by BPX1PCT.

h_digit2
The return code returned by BPX1PCT.

h_digit3
The reason code returned by BPX1PCT.

For details on rv,rc,rsnc, refer to *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803.

System action: NFS processing continues with the limitation that RPCSEC_GSS requests will not be processed for the failing TCP/IP stack identified by the *stackname* in this message.

Programmer response: None

Operator response: None

System programmer response: Check the TCPIP/ stack configuration for the stack identified by the *stackname*.

GFSA743W (*procname*) Could not get GSS credentials for host *hostname*: Routine *rtn()* returned Major Status(*h_digit1*) Minor Status(*h_digit2*).

Explanation: This message is issued when the NFS server's attempt to acquire the GSS credential failed. In the message text:

hostName The name of the host for which GSS credentials could not be acquired.

rtn The name of the GSS API that failed.

h_digit1 The major status returned by the failing GSS API.

h_digit2 The minor status returned by the failling GSS API.

Note: For details on the failing GSS API, Major Status and Minor Status, refer to *z/OS Integrated Security Services Network Authentication Service Programming*, SC24-5927.

System action: NFS processing continues with the limitation that RPCSEC_GSS requests will not be processed for the failing host identifed by the hostName in this message.

Programmer response: None

Operator response: None

System programmer response: Check the GSS/Kerberos configurations.

GFSA744E(*procname*) Routine *text* () could not retrieve the TCP stack names, Return Value(*dec_value*) Return Code(*HEX_value1*) Reason Code(*HEX_value2*).

Explanation: This message specifies that the z/OS NFS Server cannot determine TCP stack names at start up. If only some stack names are found, the z/OS NFS Server start up will proceed, but only with the resolved stacks.

In the message text:

procname The startup procedure name of z/OS NFS Server.

text The name of the z/OS USS function (BPX1PCT) that returned the error.

dec_value The decimal Return Value from the function.

HEX_value1 The hex value of Return Code from the function.

HEX_value2 The hex value of Reason Code from the function.

System action: The z/OS NFS Server shuts down.

Programmer response: None

Operator response: Contact the system programmer.

System programmer response: Correct the TCP stacks configuration and restart the z/OS NFS Server.

GFSA750I (*procname*) SMF PROCESSING ACTIVE FOR USER LOGOUT.

Explanation: SMF processing is active for user logout records. This message displays in response to the **status** operand of the **modify** command.

In the message text:

procname The name of the start procedure.

System action: NFS server processing continues.

GFSA751I (*procname*) SMF PROCESSING SUSPENDED FOR USER LOGOUT.

Explanation: The system-managed facility (SMF) processing is suspended for user logout records. This message displays in response to the **status** operand of the **modify** command or after a nonzero return code from SMF. See message GFSA754I for the SMF return code.

In the message text:

procname The name of the start procedure.

System action: NFS server processing continues without generating any user or file SMF records.

Operator response: Resume SMF recording by entering a **modify** command that specifies **smf=on**.

GFSA752I (*procname*) SMF PROCESSING ACTIVE FOR FILE TIMEOUT.

Explanation: The system-managed facility (SMF) processing is active for file timeout records. This message displays in response to the **status** operand on the **modify** command.

In the message text:

procname The name of the start procedure.

System action: NFS server processing continues.

GFSA753I (*procname*) SMF PROCESSING SUSPENDED FOR FILE TIMEOUT.

Explanation: The system-managed facility (SMF) processing is suspended for file timeout records. This message displays in response to the **status** operand of the **modify** command or after a nonzero return code from SMF. See message GFSA754I for the SMF return code.

In the message text:

procname The name of the start procedure.

System action: NFS server processing continues without generating any user or file SMF records.

Operator response: Resume SMF recording by entering the **modify** command that specifies **smf=on**.

GFSA754I (*procname***) UNEXPECTED RETURN CODE** *d_digits* **RECEIVED FROM SMF WHILE WRITING RECORD TYPE 42 SUBTYPE [7|8].**

Explanation: The NFS server received a nonzero return code, *d_digits*, while processing a file timeout (subtype 7) or user logout (subtype 8) record.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues. No more SMF records of the same type and subtype are generated until the NFS address space is restarted or the **smf=on** operand of the **modify** command is entered.

Operator response: Notify your system programmer.

System programmer response: See *z/OS MVS System Management Facilities (SMF)* for information about the return code. Correct the problem, and have the operator enter a **modify** command that specifies **smf=on**.

GFSA770I z/OS UNIX REGISTRATION SUCCESSFUL.

Explanation: A connection with z/OS UNIX was established.

System action: NFS processing continues.

GFSA771I (*procname***) z/OS UNIX MOUNTS SUSPENDED.**

Explanation: z/OS UNIX mount processing is suspended by the **freeze=onhfs** operand of the **modify** command. This message displays in response to either the **freeze=onhfs** operand or the **status** operand of the **modify** command. Any additional z/OS UNIX mount requests from the network are ignored. The existing mounts are unaffected.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA772I (*procname***) z/OS UNIX MOUNTS RESUMED.**

Explanation: Mount requests to the z/OS UNIXl file system have been enabled. This message displays in response to a console **modify** command.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA776I z/OS UNIX CANNOT RESOLVE PATH NAME *text***.**

Explanation: There was a failure to resolve the path name with UNIX when initializing from the mount handle data sets. The z/OS UINX file system was removed or renamed. If the user attempts to access a file object under this mount point, the NFS error response NFSERR_STALE is returned.

System action: NFS processing continues.

GFSA777I z/OS UNIX SERVICE REQUESTER DOES NOT HAVE SECURITY PRIVILEGE.

Explanation: The client user must be defined to Resource Access Control Facility (RACF) as a user of z/OS UNIX to access z/OS UNIX file objects.

System action: The request is stopped. NFS processing continues.

System programmer response: Check the System Authorization Facility (SAF) security product user profiles.

GFSA782I (*procname***) NO ACTIVE z/OS UNIX MOUNT POINTS.**

Explanation: This message is in response to the **list=mounts** operand of the **modify** command and shows that no z/OS UNIX clients are connected to NFS.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA784I *text1* **RPC z/OS UNIX ERROR VNODE_OP** *text2* **RC:** *number1***:** *text3* **RSN:** *number2 number3***.**

Explanation: NFS encountered a problem on an interface call to z/OS UNIX. The error was encountered during processing of a *text1* remote procedure call (RPC).

In the message text:

text2 The value of *text2* data is the function called when failure occurred.

number1 The value of *number1* is the return code.

text3 The value of *text3* text is the English description of the return code.

number2 number3
The combination of *number2 number3* represents the reason code as returned by z/OS UNIX.

System action: NFS processing continues.

System programmer response: See z/OS UNIX documentation for a full explanation of z/OS UNIX reason codes (for example, *z/OS UNIX System Services Messages and Codes*).

GFSA786I MULTI-COMPONENT LOOKUP REQUEST FOR PATHNAME *text* **CANNOT BE RESOLVED.**

Explanation: The multi-component **lookup** request for the path name failed. Possible reasons follow:

- Symbolic links cannot be embedded in a multi-component path name because it is not supported at this time. See "Accessing symbolic links on z/OS NFS version 4" on page 54 for details on NFS version protocol use.
- Either the specified path name specified is not supported, or access is not allowed.
- A public path is not set up on this server.

System action: The request fails. NFS processing continues.

User response: Construct a different **lookup** request with a valid path name.

GFSA787I z/OS UNIX PATHNAME SPECIFIED, BUT NOHFS SPECIFIED IN THE ATTRIBUTE DATA SET.

Explanation: A z/OS UNIX path name was specified but **nohfs** was also specified in the attributes data set, which disables z/OS UNIX processing.

System action: Network File System Server request fails.

User response: Either change the attributes data set to activate z/OS UNIX processing and restart the NFS server, or change the path name.

GFSA788I z/OS UNIX OPERATION *unixop* **FAILED WITH RETURN CODE** *number***, NFS client IP address** *IPv6addr* **hostname** *hostname***.**

Explanation: z/OS UNIX operation *unixop* failed with return code *rc* for NFS client *IPv6addr* and hostname *hostname*.

In the message text:

unixop The z/OS UNIX operation that failed.

number The return code.

IPv6addr
The NFS client address in the form of 0:FFFF:a.b.c.d

hostname
The hostname and domain name of the NFS client host.

System action: NFS processing continues.

System programmer response: See z/OS UNIX documentation (for example, *z/OS UNIX System Services Messages and Codes*) for a full explanation of z/OS UNIX reason codes.

GFSA790I (*procname***) NETWORK FILE SYSTEM SERVER holds delegation=** *access***.**

Explanation: This message displays the fact that the NFS server holds access type delegation for the file requested in the LISTLOCKS operator command. This does not necessarily mean that the NFS v4 delegation is currently granted to a remote NFS Client.

access indicates the delegation access type:

- R - Read
- RW - Read/Write

System action: The Network File System server continues processing.

User response: None

Operator response: None.

System programmer response: None.

GFSA791I (*procname***) Owner (***serverid clientid userid processid***) offset=***hex1* **len=***hex2* **lockacc=***mode* **status=***status*

Explanation: This message displays a byte range lock owner for the file requested in the LISTLOCKS operator command. Message GFSA791I will be returned for each lock. The lock owner is identified by *serverid, clientid, userid,* and *processid*. The lock is for offset *hex1* and length *hex2*. The lock access *mode* (share/exclusive) and the current lock *status* (waiting/granted) are also displayed.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA792I **(***procname***) Owner (***serverid clientid userid processid***) access=***access* **deny=***deny* **delegation=** *delg_access*

Explanation: This message displays a lock share holder for the file requested in the LISTLOCKS operator command. Message GFSA792I will be returned for each share holder. The share owner is identified by *serverid, clientid, userid,* and *processid*. The lock access mode *access*, deny mode *deny*, and delegation mode *delg_access* are displayed.

delg_access
indicates the delegation access type:

- R - Read
- RW - Read/Write

System action: The Network File System server continues processing.

Operator response: None.

System programmer response: None.

GFSA793I **(***procname***)** *text* **does not have Locks.**

Explanation: This message is returned from the LISTLOCKS operator command for the specified file *text*, if the file does not have any locks or shares.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA794I **(***procname***) LISTLOCK for** *text* **found** *count* **shares/locks. Command was completed successfully.**

Explanation: This message indicates that processing of the LISTLOCKS operator command for the specified file *text* has been completed. *count* is the number of shares and/or locks that were found and were listed with messages GFSA791I (for byte range locks) or GFSA792I (for shares).

System action: The NFS server continues.

Operator response: None.

System programmer response: None.

GFSA796I **(***procname***)** *cmdstr***: completed successfully.**

Explanation: The modify command has completed without error and there are no other messages associated with the command.

- *procname* - Network File System Server start up procedure name
- *cmdstr* - the text of the command issued

System action: None.

Operator response: None.

System programmer response: None.

GFSA797I **(***procname***) Command** *cmdstr* **is obsolete. Instead, use the TRACE CT operator command to capture the desired diagnostics.**

Explanation: The a modify command has been issued to change the NFS server logging to a level that is no longer supported.

- *procname* - Network File System Server start up procedure name
- *cmdstr* - the text of the command issued

System action: None.

Operator response: None.

System programmer response: Issue the appropriate Component Trace commands to capture debug trace information.

GFSA801I **MOUNT FAILED:** *text*

Explanation: The value of *text* can be any of these messages.

- `FILE MAPPING ENABLED BUT NO SIDE FILE SPECIFIED`
- `STORAGE LIMIT REACHED LOADING MAPPING SIDE FILE`
- `MAPPING SIDE FILE NOT FOUND`
- `ERROR OPENING/READING MAPPING SIDE FILE`
- `MAPPING SIDE FILE HAS INVALID SYNTAX OR FORMAT`
- `SIDE FILE SPECIFIED BUT MAPPING IS DIS-ALLOWED BY INSTALLATION`

User response: Take one of these actions depending on the value of *text*:

- Specify a side file if **fileextmap** is ON.
- Fix the problem with mapping the side file.
- Ask the system administrator to change the **sfmax** value, and then reissue the **mount** command.

GFSA802E **REMOUNT FAILED — PHYSICAL FILE SYSTEM CHANGED. PATH: pathname PREV: datasetname CURRENT: datasetname**

Explanation: During a restart of the NFS server, the rebuild of the mount point recorded in the mount handle database failed. The failure occurred because the physical file system for the mount point was changed by a TSO UNMOUNT command because the mount point was originally mounted.

System action: NFS server processing continues.

GFSA803I Mounting to a remote file system not allowed.

Explanation: The z/OS NFS server received a request to access a cross-system mount point. Crossing into local and remote file systems is not supported by the z/OS NFS Server for NFS protocol versions 2 and 3, and remote file systems for the NFS protocol version 4.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Investigate and correct the source of the request. If necessary, keep the existing z/OS NFS server traces and contact IBM Support.

GFSA804I Mounting on a file not allowed.

Explanation: The NFS server received a Mount request to mount on a file. File Systems can only be mounted on directories, not on files.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Investigate and correct the source of the request. If necessary, keep the existing z/OS NFS server traces and contact IBM Support.

GFSA805E *(procname) client_host_or_ip_address* **attempts to mount to** *pathname* **for file system** *file_system_name_new***. However, there are** *number* **client(s) already mounted to this path with file system** *file_system_name***.**

Explanation: An attempt to nfs mount to *pathname* for a file system *file_system_name_new* from a particular client *client_host_or_ip_address* failed because at least one mount point remained on the server side for that *pathname* mount path but for a different file system *file_system_name*. Usually this happens when nested file systems underneath the mount path have changed (attached or detached) during server processing.

System action: NFS continues processing.

Operator response: Perform the following recovery actions:

- Check all existing mount points on the server side with the following:

  ```
  F procname,LIST=MOUNTS
  ```
- Find all of the clients using the same mount path */pathname*
- Prevent any client from mounting to z/OS UNIX file systems with the F *procname*,FREEZE=ONHFS operator command
- Inform the system administrator about the file system change and about the possibility of performing unmount on clients for */pathname* on behalf of new mount from *client_host_or_ip_address*
- After clients unmount the mount-points, verify again that there are no other clients accessing the */pathname* with the F *procname*,LIST=MOUNTS operator command
- If no other client is accessing the */pathname* but the Server still shows them in F *procname*,LIST=MOUNTS report, use F *procname*,UNMOUNT='/hfs/ *pathname'* to remove any information on the server side about obsolete mount points
- Enable clients to mount to z/OS UNIX file systems with F *procname*,FREEZE=OFFHFS

System programmer response: Contact clients to inform them about the file system change situation during new mount processing and a need to unmount the mount-points on all clients.

GFSA806I Mount removed from *text* **server instance, Reason:** *rsnstr***, Client:** *host***, Mount path:** *mpath***, Filesystem:** *fsname***, mountFH:** *fhandle*

Explanation: A mount for server text was dropped. At z/OS NFS server shutdown, the message is issued and the mount path is written into the MHDB along with the reason string *rsnstr*. At next z/OS NFS startup, the reason string in the MHDB is detected and the message issued. By contrast, at resource timeout the message is issued, but no MHDB record will be written in the swapped MHDB dataset. In the message text:

text "current" or "prior". The text "prior" will only be seen at z/OS NFS server startup from MHDB diagnostic records. All other instances of this message will see "current".

rsnstr The reason for the mount to be dropped is one of the following:

- "Mount Block current count is 0"
- "No MVS File Block found"
- "No Mount Usage Block found"
- "NFS version 4 non-recorded mount"
- "Mount Usage Block marked for deletion"
- "Mount Usage Block has usage count of 0"
- "Mount timed out"
- "Mount has no hostname"
- "Mount record has no hostname"
- "Mount MVS File Block locate reasoncode=*fbrc*"

  where *fbrc* is an internal reason code

host the hostname of the mount being dropped

mpath the pathname of the mount being dropped

fsname the USS filesystem name for the mount pathname, or "MVS" for an MVS mount *fhandle* hex representation of the mount filehandle, so the mount keys can be correlated to dependent filehandles getting dropped errors

Note: If "No Mount Usage Block found" is the reason string *rsnstr*, then the host and *fhandle* values can be ignored.

System action: NFS continues processing.

Operator response: None, unless an NFS client mount is lost. In that case, collect NFS Server log data sets and notify the system programmer.

System programmer response: None, unless an NFS client mount is lost. In that case, collect NFS Server log data sets and contact the IBM Support Center.

GFSA811I CANNOT FLUSH PARTIAL RECORDS FOR DATA SET *text1***(***text2***): FB** *h_digits***.**

Explanation: There was not enough memory to allocate the storage required to flush partial records to the data set at data set close time. The partial record was discarded.

System action: The data set is closed. NFS processing continues.

System programmer response: Increase the size of the step region. The data set might be incomplete.

GFSA812I FLUSH FAILED: RC *d_digits1* **OFFSET** *d_digits2* **WAS DROPPED FOR DATA SET** *text1***(***text2***).**

Explanation: When NFS attempted to flush cached data at data set close time, the error *d_digits1* was detected. The data at offset *d_digits2* was discarded. This error message follows more specific error messages.

System action: The data set is closed. NFS processing continues.

System programmer response: See the message preceding this message to determine the correct action. The data set might be incomplete. See Table 66 on page 417 for a description of the return code *d_digits1*.

GFSA813I REMOVE/RENAME FAILED: RC *h_digits* **DSN** *text1***(***text2***).**

Explanation: The error *h_digits* was detected during an attempt to remove or rename member *text2* of PDS *text1*.

System action: The current NFS request fails. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: The value of *h_digits* is the return code from the z/OS DFSMS STOW macro. See *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of STOW return codes.

GFSA814I RENAME FAILED: RC *h_digits* **DSN** *text1* **OLDMEM** *text2* **NEWMEM** *text3***.**

Explanation: The error *h_digits* was detected during an attempt to rename member *text2* to *text3* in partitioned data set (PDS) *text1*.

System action: NFS processing continues.

System programmer response: The value of *h_digits* is the return code from the z/OS DFSMS STOW macro. See *z/OS DFSMS Macro Instructions for Data Sets* for the explanation of STOW return codes.

GFSA815I RENAME FAILED: RC *h_digits1* **OLDDSN** *text1* **VOL** *text2* **UNIT** *h_digits2* **NEWDSN** *text3***.**

Explanation: The error *h_digits1* was detected during an attempt to rename the non-VSAM data set *text1* to *text3*. The value of *text2* is disk volume serial number. The value of *h_digits2* is the disk device type code.

System action: NFS processing continues.

System programmer response: The value of *h_digits1* is a composite of one or more error return codes that the system encountered when trying to rename the data set. You can decode the hexadecimal digits using the following list.

Renaming a data set requires three steps:

1. Uncatalog the old data set name,
2. Rename the data set in the disk VTOC, and
3. Catalog the new data set name.

Should an error occur in either of the last two steps, the prior step or steps are undone to preserve the old data set name.

Find the step that failed by matching the value in byte zero of the return code with one of the values under the heading **Byte 0** in the following list. Byte three contains the return code from the first failing DFP service (uncatalog/catalog or DADSM rename). If more errors occur during an attempt to recatalog or rename the data set back to the old name, the return codes are placed in bytes one and two respectively.

Byte 0	**Meaning/other bytes**
00	Error uncataloging old data set name. Byte 3: Uncatalog return code.
01	Error renaming the data set. Byte 1: Recatalog return code for old data set name. Byte 3: DADSM rename return code.
02	Error cataloging new data set name. Byte 1: Recatalog return code for old data set name. Byte 2: DADSM rename return code for old data set

name. Byte 3: Catalog return code for new data set name.

Catalog and DADSM rename return codes are documented in *z/OS DFSMSdfp Advanced Services*.

Message IEC614I is written for DADSM rename errors and contains more diagnostic codes. These codes are documented in *z/OS DFSMSdfp Diagnosis*.

GFSA816I HOST NAME OF IP ADDRESS (*d_digits*) WAS NOT FOUND BY TCP/IP.

Explanation: The client host name of IP address *d_digits* is not defined in either the TCP/IP domain name server or the TCP/IP site table. See *TCP/IP: Performance Tuning Guide*.

System action: NFS processing continues. The dotted IP address is used as the host name.

System programmer response: Insert this client host's entry into either the TCP/IP domain name server or the TCP/IP site table.

GFSA817I *text1* REQUEST NOT VALID ON ALIAS NAME *text2*.

Explanation: The value of *text1* can be remove or rename. The value of *text2* is an MVS access method services alias name of a file that also has a true name. Remove (**rm** or **rmdir**) and rename (**mv**) requests cannot be run using an alias name. The true file name is required.

System action: The request is stopped. An I/O error indication is returned to the client. NFS processing continues.

System programmer response: Inform the client user of this error.

User response: Provide the true name of the file in the request.

GFSA818I (*procname*) EXPORTS: NO VALID HOST NAMES IN *text* LIST.

Explanation: None of the client host names in the read/write or access list is defined to the network (see *text*).

In the message text:

procname The name of the start procedure.

System action: The NFS does not export the associated directory if the result is a null access list. If the result is a null read/write list, the directory is exported with read-only access.

System programmer response: Correct the host names in the exports data set or have the host names defined to the network.

GFSA819I DATA SET *text1* CREATION USING DATA CLASS = *text2*.

Explanation: Data set *text1* is being allocated using the attributes in data class *text2*.

System action: NFS processing continues.

GFSA820I CATALOG ERROR OCCURRED WHILE [RETRIEVING|UPDATING] CATALOG INFORMATION FOR *text*. RETURN CODE IS *d_digits1*, REASON CODE IS *cc-d_digits2*.

Explanation: The catalog management module IGG0CL*cc* returned the return code *d_digits1* and reason code *d_digits2* as the result of a catalog error or an exception condition. The value of *text* is the name of the data set against which the retrieve or update operation was performed.

System action: NFS processing continues.

System programmer response: See message IDC3009I in *z/OS MVS System Messages, Vol 6 (GOS-IEA)* for specific return code and reason code information.

GFSA821I ERROR OCCURRED WHILE UPDATING THE FORMAT 1 DSCB FOR *text1* ON *text2*. FUNCTION CODE IS *d_digits1*, RETURN CODE IS *d_digits2*, REASON CODE IS *d_digits3*.

Explanation: In the message text:

text1 The value of *text1* is the name of the data set.

text2 The value of *text2* is the serial number of the volume on which the data set resides.

The function code can be one of the following codes:

2 Deserializing the unit control block (UCB)

4 Deserializing the direct access storage device (DASD) volume

12 Searching for the UCB

16 Serializing the DASD volume

20 Reading the data set control block (DSCB)

24 Writing the DSCB

System action: NFS processing continues.

System programmer response: See the following manuals for specific return code and reason code information:

2 *z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*, macro UCBPIN

4 *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*, macro DEQ

12	*z/OS MVS Programming: Authorized Assembler Services Reference SET-WTO*, macro UCBLOOK
16	*z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU*, macro RESERVE
20	*z/OS DFSMSdfp Advanced Services*, return codes from CVAFDIR
24	*z/OS DFSMSdfp Advanced Services*, return codes from CVAFDIR

GFSA822I (*procname*) UNABLE TO PERFORM FILE MAPPING BECAUSE NO SIDE FILE SPECIFIED OR LOADED

Explanation: File mapping cannot be performed because a side file was not specified either as a default or at the mount point.

In the message text:

procname The name of the start procedure.

System action: The operation fails.

System programmer response: Specify a side file in the attributes data set.

User response: Specify a side file in the **mount** command.

GFSA823I (*procname*) PUBLIC PATH CANNOT BE ESTABLISHED.

Explanation: The public path name(s) specified in the **public** keyword cannot be established during server startup. This could be because the path is not exported or because it does not exist.

In the message text:

procname The name of the start procedure.

System action: NFS server startup ends.

Operator response: Make sure that the public path name exists and is exported if export list checking is enabled. Correct the problem and restart the server.

GFSA824W (*procname*) NETWORK FILE SYSTEM CLIENT *ipaddr* (*clientname*) DID NOT SEND COMPLETE RECORDS FOR OFFSET (*offset*) FOR DATA SET *dsname* WITHIN *seconds* SECONDS.

Explanation: During MVS z/OS conventional data set timeout for the data set *dsname* in TEXT mode, the NFS server detected incomplete records sent by the NFS client *ipaddr(clientname)* that were not completely processed in the required time (*seconds*) of the timeout value that was extended.

In the message text:

procname The name of the start procedure.

ipaddr The client IP address

clientname Client name

offset Offset into record

dsname Name of data set

seconds Time in seconds

System action: NFS server continues processing without closing the detected data set. NFS server will wait for an extended period for the NFS client to send the needed data to complete the records.

Operator response: Contact the end user.

User response: Check the network and end-user application on the client side. It is possible that the client terminated or an application is hung or has terminated. Restart the application or re-boot the client workstation to continue processing the data set.

GFSA825W (*procname*) DATA SET *dsname* CLOSED WITH PARTIAL RECORDS.

Explanation: During MVS z/OS conventional data set timeout or UNMOUNT or NFS server shutdown, the NFS server closed data set *dsname*, which contains partial records because the client did not send the necessary data packets to complete the partial record within the server extended timeout wait.

In the message text:

procname The name of the start procedure.

dsname Name of data set

System action: NFS server closes the data set and continues processing.

Operator response: Contact the end user.

User response: Check the data integrity of the data set. Check the network and end-user application on the client side. It is possible that the client terminated or an application is hung or has terminated. Restart the application or re-boot the client workstation to continue processing the data set if the NFS server has not shutdown yet.

GFSA827I Control block (*text1*) allocation failed.

Explanation: An operation to allocate virtual storage for control block *text1* was attempted but was unsuccessful.

In the message text:

text1 The name of the control block.

System action: The current NFS request fails. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Increase the size of the step region.

GFSA829I REQUEST *h_digits* **INVALID CREDENTIALS FLAVOR** *d_digits***.**

Explanation: An incorrect credentials type was received from the client. This is probably a client software error. NFS supports UNIX and nonauthentication styles.

In the message text:

h_digits The value of *h_digits* is the request block address.

System action: The request is stopped. NFS processing continues.

System programmer response: Inform the client user that the credentials used are not valid.

GFSA832I REQUEST *h_digits* **INVALID MEMBERNAME FOR** *text***.**

Explanation: In the message text:

text The value of *text* is the member name of a partitioned data set (PDS) that was specified as a file name by the NFS client user. The file name specified was incorrect or was not found in the PDS.

h_digits The value of *h_digits* is the request block address.

System action: The request is stopped. NFS processing continues.

System programmer response: Inform the client user of this error.

User response: Correct the error and resubmit the request.

GFSA833I REQUEST *h_digits* **PARSE FAILED FOR** *text***.**

Explanation: In the message text:

text The value of *text* is the member name of a partitioned data set (PDS) or a data set name that was specified as a file name by the NFS client user. The file name specified was incorrect, was not found in the PDS, or was an incorrect or nonexistent data set.

h_digits The value of *h_digits* is the request block address.

System action: The request is stopped. NFS processing continues.

System programmer response: Inform the client user of this error.

User response: Correct the error and resubmit the request.

GFSA834I Control block *text1* **insertion failed** *digit1***.**

Explanation: The NFS server can not insert *text1* control block (at location *digit1*) into the control block collection.

In the message text:

text1 The name of the control block.

digit1 The location of the control block.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA840I DYNAMIC ALLOCATION FAILED WITH RETURN CODE *h_digits* **FOR DATA SET** *text(dsname)***.**

Explanation: A dynamic file allocation error occurred.

In the message text:

h_digits The value of *h_digits* is the dynamic allocation return code.

dsname The value of *dsname* is the data set name.

text The value of *text* is the member name, if any.

System action: The request is stopped. NFS processing continues.

System programmer response: This message is preceded by either message GFSA853I or GFSA854I. See the programmer response for the message that precedes this message to determine the appropriate action.

GFSA841I READDIR: *text1* **(***digit1* **bytes) too small, at least** *digit2* **bytes needed.**

Explanation: The NFS server received a ReadDir request. The NFS Client specified buffer/dircount (*text1*) is too small (*digit1* bytes). It should be at least *digit2* bytes.

In the message text:

text The client-specified buffer/dircount.

digit1 The size of the client-specified buffer/dircount.

digit2 The required minimum size of the client-specified buffer/dircount.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA842I *dsname1* **UNSUPPORTED DSORG** *dsname2***.**

Explanation: NFS does not support data set organization of data set *dsname1*. The value of *dsname2* can be `ISAM` or `UNKNOWN`.

System action: The request is stopped. NFS processing continues.

System programmer response: Inform the client user of this error.

GFSA843I **CREATE FAILED FOR** *dsname***.**

Explanation: An error occurred while the NFS was trying to create the data set *dsname*. This message follows other messages that describe the error in greater detail.

System action: The request is stopped. NFS processing continues.

System programmer response: See the message(s) preceding this message to determine the appropriate response.

GFSA846I **Cannot locate Host Block from** *text1* **=** *digit1***.**

Explanation: NFS Server can not locate the Client Host control block from the *text1* control block (pointer to *text1* = *digit1*).

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA847I **IDCAMS ERROR:** *text***.**

Explanation: The value of *text* is an access method services error message.

System action: The request is stopped. NFS processing continues.

System programmer response: See *z/OS MVS System Messages* for more information about the access method services error message, or use LookAt. For a description of LookAt, see "How to look up message explanations with LookAt" on page xv.

GFSA848I **PDS** *text* **IS NOT EMPTY.**

Explanation: The NFS client user issued a **rmdir** (remove directory) AIX or UNIX command to remove a partitioned data set (PDS) that was not empty. The NFS version 2 protocol specification requires the directory (PDS) to be empty before it is removed. This is a NFS client user error.

In the message text:

text The value of *text* is the name of the PDS.

System action: The request is stopped. An error is returned to the client. NFS processing continues.

System programmer response: Inform the NFS client user of this error.

User response: Remove all files in the directory, and then resubmit the **rmdir** request.

GFSA849I **New and old files are not members (***text1***) of the same PDS (***text2***).**

Explanation: Rename is not allowed for a member of a partitioned data set (PDS) when the target name is not in the same PDS. This is a NFS client user error.

In the message text:

text1 The member names of the files.

text2 The name of the PDS.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Inform the NFS client user of this error.

User response: Check the file name used. Correct it and try again.

GFSA850I **READDIR: Server Cookie Verifier NOT SAME, Current Cookie Verifier:** *digit1* **Saved Cookie Verifier:** *digit2***.**

Explanation: NFS Server received a ReadDir continuation request. The target directory changed since the previous request was processed. Therefore, the Readdir continuation request can not be processed.

In the message text:

digit1 The current cookie verifier.

digit2 The saved cookie verifier.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: None. The client should automatically recover from this error. Keep the existing z/OS NFS Server traces and contact IBM Support if the client does not recover.

GFSA851I READDIR: Bad Cookie, Server Cookie Verifier: *digit1***, Client Cookie:** *digit2***.**

Explanation: NFS Server received a ReadDir continuation request. The Cookie verifier sent back to the Server by the Client does not match the Cookie Verifier saved by the Server.

In the message text:

digit1 The server cookie verifier.

digit2 The client cookie.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA853I DYNAMIC ALLOCATION: INPUT VALIDATION ROUTINE REJECTED ALLOCATION.

Explanation: Dynamic allocation failed when issued by the installation input validation routine. Installation configuration errors probably occurred.

System action: The request is stopped. NFS processing continues.

System programmer response: See the *z/OS MVS Installation Exits* for more information about the Input Validation routine.

GFSA854I DYNAMIC ALLOCATION: RC=*d_digits* **ERROR=***h_digits1* **INFO=***h_digits2* **:** *text***.**

Explanation: Dynamic allocation failed with return code *d_digits*, error reason code *h_digits1*, and information reason code *h_digits2*.

In the message text:

text The value of *text* is a description of the interrupted dynamic allocation request.

System action: The request is stopped. NFS processing continues.

System programmer response: See *z/OS MVS Programming: Authorized Assembler Services Guide* for more information about these codes.

GFSA858I OPEN FAILED RC *d_digits* **FOR DATA SET** *dsname1***(***dsname2***).**

Explanation: An error occurred during an attempt to open the data set.

In the message text:

dsname1 The value of *dsname1* is the data set name.

dsname2 The value of *dsname2* is the member name, if any.

d_digits The value of *d_digits* is the return code.

System action: The request is stopped. NFS processing continues.

System programmer response: If this error was not caused by an out-of-memory condition, contact your programming support personnel. See Table 66 on page 417 for a description of the return code *d_digits*.

GFSA859I READ FAILED RC *d_digits* **FOR DATA SET** *dsname1***(***dsname2***).**

Explanation: An error occurred during an attempt to read the data set.

In the message text:

dsname1 The value of *dsname1* is the data set name.

dsname2 The value of *dsname2* is the member name, if any.

d_digits The value of *d_digits* is the return code.

System action: The request is stopped. NFS processing continues.

System programmer response: If this error was not caused by an out-of-memory condition, contact your programming support personnel. See Table 66 on page 417 for a description of the return code *d_digits*.

GFSA860I WRITE FAILED RC *d_digits* **FOR DATA SET** *dsname1***(***dsname2***).**

Explanation: An error occurred during an attempt to write the data set.

In the message text:

dsname1 The value of *dsname1* is the data set name.

dsname2 The value of *dsname2* is the member name, if any.

d_digits The value of *d_digits* is the return code.

System action: The request is stopped. NFS processing continues.

System programmer response: If this error was not caused by an out-of-memory condition, contact your programming support personnel. See Table 66 on page 417 for a description of the return code *d_digits*.

GFSA862I CATALOG (*text*) COULD NOT BE LOCATED.

Explanation: The user catalog named *text* which contains the entry for an index, could not be located. The catalog does not exist or is not mounted. If it does not exist, the entry in the master catalog might be incorrect.

System action: NFS processing continues.

System programmer response: Investigate why the catalog could not be found and take corrective action.

GFSA863I READDIR ON ROOT IS NOT ALLOWED.

Explanation: The user attempted to list the contents of the master catalog.

System action: The request is stopped. NFS processing continues.

GFSA864I (*procname*) CANNOT OPEN THE EXPORTS DATA SET.

Explanation: The server was unable to open the exports data set defined in the job control language (JCL) for DDNAME EXPORTS. The DD statement might be missing or the data set name might be incorrect.

In the message text:

procname	The name of the start procedure.

System action: NFS stops.

Operator response: Notify your system programmer.

System programmer response: Correct the JCL for DDNAME EXPORTS.

GFSA865I (*procname*) EXPORTS: UNEXPECTED OPTION (*text*)-- SHUTDOWN SCHEDULED.

Explanation: The option information provided in the *text* data is incorrect. This error could occur as a result of unexpected blanks, incorrect syntax, or mutually exclusive options (for example, both **ro** and **rw**).

In the message text:

procname	The name of the start procedure.

System action: Checking of the exports data set continues, but the shutdown of NFS occurs at its completion.

System programmer response: Correct the exports data set and restart NFS.

GFSA866I (*procname*) EXPORTS: DIRECTORY *dsname* WAS NOT EXPORTED.

Explanation: An error was encountered that was severe enough to prevent the data set or index named *dsname* from being exported. This message follows a more specific error message that is issued to the console only, to the z/OS NFS Server log only, or to both the console and the z/OS NFS Server log depending on the severity of the error in the exports data set.

In the message text:

procname	The name of the start procedure.

System action: NFS processing continues or terminates depending on the severity of the error in the exports data set and the phase the z/OS NFS Server is processing.

System programmer response: Correct the exports data set.

GFSA867I (*procname*) EXPORTS: *dsname1* CANNOT BE EXPORTED BECAUSE *dsname2* ALREADY IS.

Explanation: The data set or index named in *dsname1* is a parent directory or a subdirectory of the data set or index named in *dsname2* which is already exported.

In the message text:

procname	The name of the start procedure.

System action: NFS processing continues.

System programmer response: Correct the exports data set.

GFSA868I (*procname*) EXPORTS: WRONG *text1* (*text2*) HAS BEEN IGNORED.

Explanation: The network template, hostname template, or IP address specified by *text2* was incorrect.

In the message text:

procname	The name of the start procedure.
text1	The type of item that has been ignored: *network template*, *hostname template*, or *IP address*.
text2	The value of the network template, hostname template, or IP address.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the exports data set.

GFSA869I (*procname*) EXPORTS: UNKNOWN HOST (*text*) HAS BEEN IGNORED.

Explanation: NFS cannot create the HOSTCACHE control block for the client host, specified by *text*. The reason is that the client host with hostname (*text*) is not defined to the network; NFS received a hostname client suffix syntax error.

In the message text:

procname The name of the start procedure.

text The host name or IP address.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the host name in the exports data set or have this host name defined to the network.

GFSA871I REQUEST *h_digits* HAS MISMATCHED UID: CRED = *d_digits1* ARGS = *d_digits2*.

Explanation: The value of *h_digits* is a block created for the logon or logout request. The value of *d_digits1* represents the credential user ID number. The value of *d_digits2* represents the client user ID number. The user ID numbers do not match, and this is considered a security failure.

System action: The client logon or logout request is stopped. NFS processing continues.

System programmer response: This is a NFS client application problem. If the NFS client application is offered by IBM, inform the IBM Support Center. If the NFS client application is offered by the programming support personnel, contact your programming support personnel.

GFSA872E No RACF support for *mvslogin* request of userid from client UID: *uid* on host: *hostname*

Explanation: The mvlogin request failed because RACF does not support a password change from the current password to new password phrase or from current password phrase to new password.

System action: The NFS server fails the request.

Operator response: None.

System programmer response: Retry the mvslogin command by subscribing to the RACF Password and Passphrase syntax rules as specified in the z/OS Security Server RACROUTE Macro Reference.

GFSA876I I/O ERROR ON DSN = *text1*(*text2*) SENSE *h_digits1* IOBCSW *h_digits2* *h_digits3*. ACCESS METHOD RC = *h_digits4* ACCESS METHOD RSN = *h_digits5*

Explanation: The physical I/O layer tried to check some previous operation in the data set and the check failed.

In the message text:

text1 The value of *text1* is the data set name.

text2 The value of *text2* is the member name, if any.

h_digits1 The value of *h_digits1* is the sense bytes 0 and 1 from the device.

h_digits2 The value of *h_digits2* is the first 3 bytes of the channel status word from the device.

h_digits3 The value of *h_digits3* is the last 4 bytes of the channel status word from the device.

h_digits4 The value of *h_digits4* is the access method return code.

h_digits5 The value of *h_digits5* is the access method reason code.

System action: The request is stopped. NFS processing continues.

System programmer response: See the appropriate device documentation for more information on the sense bytes and channel status word.

GFSA877I R0=*h_digits1* R1=*h_digits2*: *text* ACCESS METHOD RC = *h_digits3* ACCESS METHOD RSN = *h_digits4*

Explanation: A SYNAD error was detected during a physical I/O operation.

In the message text:

h_digits1 The value of *h_digits1* is the contents of register 0.

h_digits2 The value of *h_digits2* is the contents of register 1.

text The value of *text* is the message returned from the SYNAD analysis function macro.

h_digits3 The value of *h_digits3* is the access method return code.

h_digits4 The value of *h_digits4* is the access method reason code.

System action: The request is stopped. If the error detected is a B37, D37, or E37 abend, NFS restores the file size to the last known file size before the SYNAD error. NFS processing continues.

System programmer response: A data management message should have displayed on the console. See *z/OS MVS System Messages, Vol 1 (ABA-AOM)* through *z/OS MVS System Messages, Vol 9 (IGF-IWM)* for a description of the return code to determine the corrective action, or use LookAt. For a description of LookAt, see "How to look up message explanations with LookAt" on page xv.

GFSA878I (*procname*) EXPORTS: CANNOT OPEN THE NETGROUP FILE /etc/netgroup . Error=(*digit1*,*x_digit2*)

Explanation: NFS cannot open the local netgroup file. The *digit1* and *x_digit2* values specify the errno and errno2 error codes. See *z/OS UNIX System Services Messages and Codes* for details on these error codes.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Check for validity of the specified netgroup file.

GFSA879I (*procname*) EXPORTS: PROBLEMS ENCOUNTERED PARSING THE NETGROUP FILE /etc/netgroup. GROUP (*text1*) IS EXPERIENCING THE PROBLEM.

Explanation: NFS was not able to parse the local netgroup file. The problem occurred while parsing Group *text1*.

In the message text:

procname	The name of the start procedure.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Correct the contents of the netgroup file.

GFSA881I FUB: *fub1addr* UNABLE TO ACCESS FILE *dsname*(*member*) OWNED BY FUB *fub2addr*.

Explanation: A user tried to access a data set that is already locked by NFS for writing by another user. The data set has not been released yet.

In the message text:

fub1addr	is the address of the file usage block (FUB) attempting to access the file.
dsname	is the data set name.
member	is the member name, if any.
fub2addr	is the address of the file usage block (FUB) that currently has the file allocated.

System action: The request is stopped. A `Not Owner` error message is returned to the user. NFS processing continues.

System programmer response: If queried by the user, the `Not Owner` error message, as it relates to NFS, is described in "Messages from the client platform (AIX)" on page 412.

GFSA883I Record *digit1* size *digit2* is too short: minimum = *digit3* DSN *text1*(*text2*).

Explanation: The NFS server detected that a record in a data set is too short – less than the minimum required size.

In the message text:

digit1	The record of the data set that is too short.
text1	The data set name.
text2	The member name, if any.
digit2	The current size of the record.
digit3	The minimum required size of the record.

System action: The current NFS request will fail. Otherwise, NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS Server traces and contact IBM Support.

GFSA886I UNABLE TO WRITE RECNO *d_digits* TO *text1*(*text2*) DUE TO PREVIOUS ERROR.

Explanation: An error was previously detected which prevents writing to a data set.

System action: The request is stopped. NFS processing continues.

System programmer response: If the previous error cannot be determined, contact your programming support personnel.

GFSA895I REQUEST *h_digits* - FILE *text* NOT ALLOCATED.

Explanation: NFS did not have the data set *text* open during a request to close the file. The file name might have been specified incorrectly, or the timeout might have already occurred for this data set, causing the server to close the data set.

In the message text:

h_digits The request block address.

text The data set name.

System action: The system ends the request. Return code X'131' is passed back to the user. The system processing continues for NFS.

GFSA896I REQUEST *h_digits1* - FILE BLOCK *h_digits2* ASSOCIATED WITH FILE *text* NOT IN USE BY CREDENTIALS *h_digits3*.

Explanation: The request to close file *text* was received, but the file was not opened by the client. A file can be closed only by the same client that opened the file.

In the message text:

h_digits1 The value of *h_digits1* is the request block address.

h_digits2 The value of *h_digits2* is the file block address.

h_digits3 The value of *h_digits3* is the credentials block address.

System action: The request is stopped. Return code 132 is passed back to the client. NFS processing continues.

GFSA897I Record *d_digits1* size *d_digits2* is too long: maximum = *d_digits3* DSN = *text1*(*text2*).

Explanation: The NFS server detected that the record *d_digit1* of data set *text1*(*text2*) is too long (its size = *d_digits2*). Its size should be at most *d_digits3*.

System action: The current NFS request fails. Otherwise, NFS processing continues.

Programmer response:

1. Check whether z/OS NFS Server text processing mode is appropriate.
2. Check the target data set characteristics.
3. Check that there is no line in the remote source file being written to the target data set that is longer than the effective record length allowed for the data set type.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA898I EOL SEQUENCE MISMATCH FOR DATA SET *text1*(*text2*).

Explanation: The NFS server processed data in TEXT mode. The end-of-line terminator was not found in the same place as the previous end-of-line terminator for an offset that is being rewritten by the client. This is a NFS client error.

System action: The request is stopped. NFS processing continues.

User response: Make sure that the record you are writing is the same size as the record you are replacing.

GFSA899I BLANKSTRIP MODE: TRAILING BLANK(S) IN RECORD *d_digits* IS NOT ALLOWED DSN = *text1*(*text2*).

Explanation: Writing data in text mode with blank stripping enabled and blanks at the end of the line to a data set with fixed-length records is not allowed.

System action: The request is stopped. NFS processing continues.

GFSA900I (*procname*) MOUNT PROCESSING ACTIVE.

Explanation: Mounts can be issued from the network. This message displays in response to the **status** operand of the **modify** command.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA901I (*procname*) MOUNT PROCESSING SUSPENDED.

Explanation: Mount processing was suspended by the **freeze=on** operand of the **modify** command. This message displays in response to either the **freeze=on** operand or the **status** operand of the **modify** command. Any additional mount requests from the network are ignored. Existing mounts are unaffected.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA902I (*procname*) MOUNT PROCESSING RESUMED.

Explanation: Mount processing has been resumed by the **freeze=off** operand of the **modify** command. Any additional mount requests from the network are honored. Existing mounts are unaffected.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA903I (*procname*) TASK *h_digits1* TCB *h_digits2* PROGRAM = *text1* = *text2*.

Explanation: This is in response to the **status** operand of the **modify** command.

In the message text:

procname
The name of the start procedure.

h_digits1
The value of *h_digits1* is the task queue address.

h_digits2
The value of *h_digits3* is the task control block address.

text1
The value of *text1* is the program name.

text2
The value of *text2* is the specific task name.

System action: NFS processing continues.

GFSA904I (*procname*) z/OS UNIX MOUNT PROCESSING ACTIVE.

Explanation: Mounts can be issued from the network. This message displays in response to the **status** operand of the **modify** command.

In the message text:

procname
The name of the start procedure.

System action: NFS processing continues.

GFSA905I (*procname*) Current CTRACE buffer is flushed and switch made to the next buffer.

Explanation: In response to the FLUSHCTR operator command the Network File System Component Trace function flushed the remaining trace buffers to the component trace external writer and switched to the next available buffer.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA907I THERE IS MORE INFORMATION IN THE Network File System LOG.

Explanation: Due to the potential impact on the console, not all of the information associated with the previous message was reported on the console. The remaining lines will only be placed in the NFS Log. This message is displayed on the console to indicate that not all the information has been displayed on the console and more information is available in the NFS Log.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA908I (*procname*) z/OS UNIX PROCESSING DISABLED.

Explanation: z/OS UNIX file system processing is suspended. This message displays in response to the **status** operand of the **modify** command.

In the message text:

procname
The name of the start procedure.

System action: NFS processing continues.

GFSA909E (*procname*) UNMOUNT COMMAND FAILED: MOUNT POINT STILL IN USE

Explanation: This message is in response to the **unmount** operand of the **modify** command. The unmount processing fails because the file system is still in use. The user can retry the **unmount** command later after there is no reference to the file system.

In the message text:

procname
The name of the start procedure.

System action: The system ends the request. The system processing continues for NFS.

GFSA910I (*procname*) *path* ACTIVE = *d_digits* : *client_list*

Explanation: This message is in response to the **list=mounts** operand of the **modify** command and shows the mounted file path, the number of active users (*d_digits*) mounted to that path, and the host names (*client_list*) mounted to that path. The ACTIVE number might not be accurate if a client has crashed without unmounting the path.

Notes:

1. A maximum of approximately ten lines will be displayed on the console. However, all the hosts will be displayed in the NFS Log message.
2. Under the following circumstances the Active count may be larger than the list of host names:
   - If a host has multiple mounts to this mount point.
   - If a host mounts using NFS version 4 without the mvsmnt attribute
   - If a host mounts using NFS version 4 with the mvsmnt attribute but the NFS client does not reuse the mount file handle after the NFS server crashes and restarts, causing it to contribute more than 1 to the active count.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

Programmer response: None.

Operator response: None.

System programmer response: None.

GFSA911I (*procname***)** *path***(***member***) ACTIVE =** *d_digits* **:** *client_list*

Explanation: This message is in response to the **list=mounts** operand of the **modify** command and shows the mounted file space (path and member), the number of active users (*d_digits*) mounted to that path/member, and the host names (*client_list*) mounted to that path/member. The ACTIVE number might not be accurate if a client has crashed without unmounting the path/member.

This message is generated when the mount point is an MVS PDS or PDSE member.

Notes:

1. Only the first ten hosts will be displayed on the console message for a given mount point. However, all the hosts will be displayed in the NFS Log message.
2. Under the following circumstances, the Active count may be larger than the list of host names:
   - If a host has multiple mounts to this mount point.
   - If a host mounts using NFS version 4 without the mvsmnt attribute
   - If a host mounts using NFS version 4 with the mvsmnt attribute but the NFS client does not reuse the mount file handle after the NFS server crashes and restarts, causing it to contribute more than 1 to the active count.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

Programmer response: None.

Operator response: None.

System programmer response: None.

GFSA912I (*procname***)** *dsname***.**

Explanation: The data set name *dsname* appears in response to a **list=dsnames** operand of the **modify** command and shows a currently active data set.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA913I (*procname***)** *text1***(***text2***).**

Explanation: The PDS member appears in response to the **list=dsnames** operand of the **modify** command and shows a currently active data set.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA914I (*procname***)** *dsname* **DEALLOCATED.**

Explanation: The data set name *dsname* appears in response to a **release=dsname(member)** operand of the **modify** command after successful deallocation.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA915I (*procname***)** *dsname* **NOT ALLOCATED.**

Explanation: The data set name *dsname* appears in response to the **release=dsname(member)** operand of the **modify** command if the data set or member specified to be released was not found.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA916I (*procname***)** *dsname* **UNMOUNTED.**

Explanation: The data set name *dsname* appears in response to an **unmount** (data set or member) operand of the **modify** command after a successful unmount.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA917I (*procname***)** *dsname* **NOT MOUNTED.**

Explanation: The data set name *dsname* appears in response to the **unmount** (data set or member) operand of the **modify** command if the data set or member specified to be unmounted was not found in the current mount list.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA918I (*procname*) *dsname text1* IS NOT A VALID NAME.

Explanation: The *text1* specified in the **release**, the **listlock**, or the **unmount** operand of the **modify** command is not a valid name.

- For the **release** operand, only a valid MVS data set name is allowed.
- For the **listlock** operand, only a valid MVS data set name or a Unix filename is allowed.
- For the **unmount** operand, only a valid MVS data set name or a Unix directory is allowed.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

Operator response: Specify the command again with a valid data set name.

GFSA919I (*procname*) *text* IS NOT A VALID MEMBER NAME.

Explanation: The member name *text* specified in the **release**, the **listlock** or the **unmount** operand of the **modify** command is not a valid MVS member name.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

Operator response: Specify the command again with a valid member name.

GFSA920I (*procname*) NO ACTIVE MOUNT POINTS.

Explanation: This message is in response to the **list=mounts** operand of the **modify** command and shows that there are no clients connected to the NFS.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA921I (*procname*) NO ACTIVE DATA SETS.

Explanation: This message is in response to the **list=dsnames** or the **listlock** operand of the **modify** command and shows that there are no clients actively accessing data sets.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA922I (*procname*) VERIFY: (*dsname*) IS NOT A VSAM DATA SET.

Explanation: This message is in response to the **verify** operand of the **modify** command and shows that the data set name *dsname* specified is not a virtual storage access method (VSAM) data set.

In the message text:

procname The name of the start procedure.

dsname The value of *dsname* specifies the data set name.

System action: The system ends the request. The system processing continues for the NFS.

GFSA923I (*procname*) VERIFY SUCCESSFUL FOR (*dsname*).

Explanation: This message is in response to the **verify** operand of the **modify** command. It shows that the verify was successful for the VSAM data set.

In the message text:

procname The name of the start procedure.

dsname The value of *dsname* specifies the data set name.

System action: The system processing continues for NFS.

GFSA924I (*procname*) VERIFY FAILED WITH RC = *d_digits* FOR (*dsname*).

Explanation: This message is in response to the **verify** operand of the **modify** command. It shows that the **verify** operand failed with a return code for the data set. Message GFSA847I follows this message in the log data set.

In the message text:

procname
The name of the start procedure.

d_digits The value of *d_digits* is the return code.

dsname The value of *dsname* is the data set name.

System action: The system ends the request. The system processing continues for NFS.

GFSA925I (*procname*) ERROR WAS DETECTED IN THE EXPORTS FILE. EXPORT LIST NOT REBUILT.

Explanation: This is the response from the **exportfs** operand of the **modify** command, indicating that one or more errors were detected in the exports data set (for example, the exports data set cannot be opened).

In the message text:

procname The name of the start procedure.

System action: NFS processing continues. The existing exports list is not changed.

Operator response: Notify your system programmer.

System programmer response: Review previous console error messages for detailed information about the specific error in the exports data set.

GFSA927I (*procname*) MODIFY EXPORTS COMMAND IGNORED - THE EXPORTS FILE IS NOT BEING USED FOR SECURITY CHECKING.

Explanation: This is the reply from the **exportfs** operand of the **modify** command, indicating that the command was ignored because the site attribute for security requested that no exports file checking be done.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues with the security options unchanged.

GFSA929I MODIFY MAPFILE COMMAND IGNORED - MAPPED SIDE FILES ARE NOT BEING USED.

Explanation: This is the reply from the **mapfile** operand of the **modify** command, indicating that the command was ignored because mapped processing attribute was not specified in the site attribute file (NFSATTR) or any **mount** command.

System action: NFS processing continues with the side files data unchanged.

GFSA930I (*procname*) LOG DATA SET IS SWITCHED FROM *text1* TO *text2*.

Explanation: A “no space” or an I/O error condition is detected when writing to the log data set. NFS logging is now switched to the other log data set.

In the message text:

procname
The name of the start procedure.

text1 The value of *text1* is the data definition (DD) associated with the switched-from log data set.

text2 The value of *text2* is the DD associated with the switched-to log data set.

System action: NFS processing continues.

Operator response: If requested by the installation, back up the switched-from log data set at this point. The switched-from data set is reused when the switched-to log data set is also filled.

System programmer response: Consider allocating larger NFS server log data sets for future NFS usage. Note that the last data buffer is lost when the log is switched.

GFSA931I (*procname*) NETWORK FILE SYSTEM SERVER LOGGING IS TERMINATED.

Explanation: The NFS logging is ended. This can be caused by a “no space” condition of the log data set.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

System programmer response: Allocate a larger log data set for future NFS usage.

GFSA932I (*procname*) LOG DATA SET *text* IS FLUSHED.

Explanation: The data buffer of the active log data set is flushed to disk. If the log is written to standard error, *text* is STDERR. Otherwise, *text* is the associated data definition of the active log data set.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA933I (*procname*) LOG DATA SET IS RE-INITIALIZED.

Explanation: The log data set is reinitialized.

In the message text:

procname The name of the start procedure.

System action: NFS processing continues.

GFSA934I (*procname*) NFSLOG1 OR NFSLOG2 DD STATEMENTS NOT DEFINED.

Explanation: Either the NFSLOG1 and NFSLOG2 DD statements are not coded in the NFS startup cataloged procedure or the log data sets have incorrect data set characteristics.

In the message text:

procname The name of the start procedure.

System action: The server stops.

System programmer response: Code the NFSLOG1 and NFSLOG2 DD statements with attributes as described in “Using log data sets” on page 523 and allocate the associated log data sets.

GFSA935I (*procname*) SWITCHED-TO LOG IN ERROR. NETWORK FILE SYSTEM SERVER LOGGING CONTINUED ON DD:*text*.

Explanation: An operator issued a **switchlog** command, but an open error is detected in the new log data set. NFS logging is continued in the original log data set.

In the message text:

procname
The name of the start procedure.

text The value of *text* is the data definition associated with the original log data set.

System action: NFS processing continues.

Operator response: Fix the inactive log data set.

GFSA936I (*procname*) NETWORK FILE SYSTEM SERVER LOG *text* SET TO FORCELOG.

Explanation: The NFS server log was closed to force all log data to disk immediately.

In the message text:

procname The name of the start procedure.

System action: NFS continues.

GFSA937E MODIFY MAPFILE COMMAND – ERROR IN SIDEFILE DATA SET – – SIDEFILE IS NOT REFRESHED.

Explanation: This message is the response from the **mapfile** operand of the **modify** command, indicating that one or more errors were detected in the side files data set (for example, the side files data set cannot be opened).

System action: NFS processing continues. The existing side files data in storage is not changed.

Operator response: Notify your system programmer.

System programmer response: Review previous error messages for detailed information as to the specific error in the side files data set.

GFSA938I SIDE FILE WAS NOT LOADED BEFORE.

Explanation: This message is the response from the **mapfile** of the **modify** command, indicating that the command was ignored because the side file was not loaded.

System action: NFS processing continues with the side files data unchanged.

GFSA942I UNMNTXXX: MODIFY COMMAND IGNORED – MOUNT POINT *mount_point* IS IN USE BY USER *user_name*

Explanation: This message is the response from the **unmnt*xxx*** operand of the **modify** command, indicating that *mount_point* was ignored because the mount point is in use by another user, *user_name*.

System action: The NFS server processing continues.

Operator response: Reenter the same operand of the **modify** command and try again later.

GFSA943E (*procname*) EXPORTS: UNEXPECTED OPTION (*text*) WAS DETECTED IN THE EXPORTS FILE.

Explanation: The option information provided in *text* is incorrect. This error could occur as a result of unexpected blanks, incorrect syntax, or mutually exclusive options (for example, both **ro** and **rw**). This message might occur after performing the EXPORTFS command.

In the message text:

procname The name of the start procedure.

System action: The NFS server continues processing.

Operator response: Reenter the same operand of the **modify** command.

GFSA944I (*procname*) *text1* Network File System Server release *text2* last APAR *text3* last changed module: *text4* compiled at *text5*

Explanation: This message is reported by the Network File System server in response to the MODIFY mvsnfs,VERSION operator command. It reports the current maintenance level of the server.

In the message text:

Text1 System level (z/OS, for example).

Text2 Network File System server release level.

Text3 Last APAR maintenance version of Network File System server.

Text4 Name of one module installed by the last maintenance level.

Text5 Compilation date/time of the module.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA945I **(***procname***)** *text1* **|** *text2* **|** *text3*

Explanation: This message is reported by the Network File System server in response to the operator command MODIFY mvsnfs,VERSION=ALL or MODIFY mvsnfs,VERSION=MODULE. Message GFSA945I is generated for each module.

In the message text:

Text1 Module name.

Text2 APAR maintenance level of the module.

Text3 Compile date/time of the module.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA946I **(***procname***) Total modules:** *digit*

Explanation: This message is returned by the Network File System server in response to the MODIFY mvsnfs,VERSION=ALL operator command, after message GFSA945I lists the information on all the modules. This message reports the total module count *digit*.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA947I **(***procname***)** *text1* **|** *text2* **|** *text3*

Explanation: This message is reported by the Network File System server during initialization if debug tracing is turned on. Message GFSA947I is generated for each module.

In the message text:

Text1 Module name.

Text2 APAR maintenance level of the module.

Text3 Compile date/time of the module.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA948I **(***procname***) Non-existent module:** *text1*

Explanation: This message is returned by the Network File System server in response to the MODIFY mvsnfs,VERSION=*text1* operator command, indicating that the module name specified as *text1* does not exist.

System action: The Network File System server continues.

Operator response: None.

System programmer response: None.

GFSA949I **COMMAND** *text* **NOT VALID.**

Explanation: The operator entered the **operator** command or the operand *text*, which is not valid.

System action: NFS processing continues.

Operator response: Reenter the operand of the **modify** command with the correct syntax.

GFSA950I **Unknown flag '-***character***'**

Explanation: The character *character* specified in the **mvslogin**, **mvslogout**, or **showattr** command is not a valid option. A usage message might follow this message.

User response: See *z/OS Network File System Guide and Reference* for a description of the valid options used with the command.

GFSA951I *text* **: can't find name for uid** *d_digits***.**

Explanation: There was an error reading information for the user ID *d-digits* from the etc/passwd file.

User response: Correct the etc/passwd file and retry the command.

GFSA952I **Retyped password does not match**

Explanation: The password entered when message GFSA975I displayed does not match the password entered when message GFSA974A displayed.

User response: Restart the **mvslogin** command sequence.

GFSA953I **Password change required by host.**

Explanation: The multiple virtual system (MVS) password for the user ID passed to the host expired. A new password is required. Message GFSA974I follows this message.

| **GFSA954I** **Host** *text1* **returned status code** *d_digits***:** *text2*

| **Explanation:** An non-zero status was detected during **mvslogin** processing. Host *text1* returned status code *d_digits* and message *text2* to the client.

| The value of *d_digits* and its associated *text2* message
| can be any of the following:

d_digits	*text2*
1	There is no *text2* message associated with this status code. This status code returns when any error is detected during RACF processing.
129	The new password phrase cannot be set when specifying a password, nor can a new password be set when specifying a password phrase. Try again.
130	The site attribute settings on the z/OS NFS Sever being communicated with indicate that **mvslogin** is not required. It is therefore not being processed.
131	The site attribute settings on the z/OS NFS Sever being communicated with indicate that **mvslogout** is not required. It is therefore not being processed.
132	User is currently logged in, this request is ignored.
878	Memory temporary not available. Try again later.

User response: Restart the **mvslogin** command sequence with valid user ID and password when appropriate.

GFSA955I *text* **logged in ok.**

Explanation: The MVS user ID *text* was logged in without any errors.

GFSA956I **usage:** *text***[-p][-n][-pn] [-P** *password***] [-g group][-norpcbind][-a account] hostname [mvs_username]**

Explanation: This message provides usage information for the **text** command.

User response: Enter the command using the correct syntax.

GFSA957I **Host** *text1* **returned error** *d_digits***:** *text2*

Explanation: An error was detected during **mvslogout** processing. Host *text1* returned error code *d_digits* and message *text2* to the client.

User response: Notify your system programmer.

GFSA958I **uid** *text* **logged out ok.**

Explanation: The user ID *text* logged out successfully.

GFSA959I **usage:** *text* **[-norpcbind] hostname**

Explanation: This message provides usage information for the *text* command.

User response: Enter the command using the correct syntax.

GFSA960I *text1***: host "***text2***" unknown.**

Explanation: The host *text2*, specified in the command *text1*, is not known to the network.

User response: Correct the host name specified and retry the command.

GFSA961I *text1***:** *text2*

Explanation: The command *text1* received an error when trying to create a client transport handle using the **clntudp_create** TCP/IP remote procedure call. The value of *text2* is the message produced by the **clnt_pcreateerror** TCP/IP procedure. This message is issued in any of these cases:

- The host name is unknown.
- The host is not operational.
- The NFS on the named host is not operational.

User response: The user response is one of these actions:

- Correct the specified host name and retry the command.
- Make sure the specified host is operational and retry the command.
- Make sure the NFS server on the named host is operational and retry the command.

GFSA962E **Unable to create CLNT handle. rc =** *d_digit1***, rsn =** *hex_digit1***.**

Explanation: The NFS client was unable to create a handle for communicating with the NFS server. The request failed with return code *d_digit1* and reason code *hex_digit1*.

System action: The Network File System Client request fails.

Operator response: None.

System programmer response: Verify that the network connection to the NFS Server is operational and that the NFS Server is functional.

GFSA963E **Call to remote Network File System Server failed. RV=***hex_digit1* **RC=***hex_digit2* **RSN=***hex_digit3***.**

Explanation: The request to the Network File System Server failed. The request failed with return value *hex_digit1*, return code *hex_digit2* , and reason code *hex_digit3*.

System action: The Network File System Client request fails.

Operator response: None.

System programmer response: Verify that the network connection to the NFS Server is operational and that the NFS Server is functional.

GFSA964I *text*: **Error: cannot determine server.**

Explanation: The *text* command found the mount path, but the server name was not returned by the local operating system service that keeps mount point information.

User response: Correct the mount point table and retry the command.

GFSA965I *text1*: **Error:** *text2* **mounted from server** *text3*, **not** *text4*.

Explanation: The wrong host name was specified for the *text1* command. *text2* is mounted from server *text3* instead of server *text4*.

User response: Specify the command again with the correct host name.

GFSA966I *text*: **Error: unknown return from usage routine.**

Explanation: The usage routine used for the **text** command returned an unknown error code.

User response: Contact your programming support personnel.

GFSA967I **Host Error:** *text*.

Explanation: The host returned an error and the message *text*. The error might be due to either of these causes:

- A porting failure occurred for the **showattr** command.
- The NFS is not compatible with the **showattr** command on the client.

User response: Contact your programming support personnel.

GFSA968I **Error: Drive** *text* **not mounted.**

Explanation: The drive *text* was not mounted. The **showattr** command cannot show the attributes for this drive.

User response: Mount the drive and reissue the command.

GFSA969I **Error: Can't open** *text* **for read.**

Explanation: The file *text* could not be opened to read the mount path.

User response: Correct the file and reissue the command.

GFSA970I **Error: Directory** *text* **not mounted.**

Explanation: The directory *text* was not mounted. The **showattr** command cannot show the attributes for this directory.

User response: Mount the directory and reissue the command.

GFSA971I **Error: filesystem** *text* **is local.**

Explanation: The file system *text* is not a NFS file system. The **showattr** command is for NFS file systems only.

User response: Reissue the command for a NFS file system.

| **GFSA972I** **usage:** *text1* **[-t] [-norpcbind] hostname** [*text2*]

Explanation: This message provides usage information for the *text1* command.

In the message text:

text2 The value of *text2* is the operating-system-dependent mount-point format.

GFSA973A **Enter MVS password:**

Explanation: The NFS requires a password for the user.

User response: If a user ID was specified in the **mvslogin** command, enter the password for that user ID. If no user ID was specified, the name from etc/passwd for the user ID that issued the **mvslogin** command was passed to the NFS. Enter the password for this user.

GFSA974A **Enter new MVS password:**

Explanation: The password for the user ID passed to NFS has expired.

User response: Enter a new password.

GFSA975A **Retype new MVS password:**

Explanation: The system requires the new password to be entered twice for verification.

User response: Reenter the new password.

GFSA976I *text1*: *text2*

Explanation: The command *text1* received an error when trying to create a client transport handle using the **clnt_call** TCP/IP remote procedure call.

In the message text:

text1 The value of *text2* is the message produced by the **clnt_perror** TCP/IP procedure.

User response: Contact your system administrator.

GFSA977I *text*:

Explanation: The command *text* received an error when trying to create a client transport handle using the **clnt_call** TCP/IP remote procedure call (RPC). This message is followed by the message produced by the **clnt_perror** TCP/IP procedure.

User response: Contact the system administrator.

GFSA978I *text* **logged in ok. Mismatch in uid/gid: z/OS UNIX uid is** *digit_1*, **gid is** *digit_2*, **client uid is** *digit_3*, **gid is** *digit_4*

Explanation: The z/OS UNIX user ID or group ID does not match the client machine user ID or group ID. The authentication is successful and the message is for information only.

GFSA979E **Unable to create UDP socket rc =** *d_digit1*.

Explanation: The NFS client was unable to create a UDP protocol socket for communicating with the NFS server. The request failed with return code *d_digit1*.

System action: The Network File System Client request fails.

Operator response: None.

System programmer response: Verify that the network connection to the NFS Server is operational and that the NFS Server is functional.

GFSA980E **Unable to get the program's port.**

Explanation: The NFS client was unable to find the port for communicating with the NFS server.

System action: The Network File System Client request fails.

Operator response: None.

System programmer response: Verify that the network connection to the NFS Server is operational and that the NFS Server is functional.

GFSA981I **Unable to establish TCP connection rc =** *d_digit1*. **Trying UDP ...**

Explanation: The NFS client was unable to establish a TCP protocol connection to the NFS server. The request failed with return code *d_digit1*.

System action: The Network File System Client will attempt to create a UDP protocol connection.

Operator response: None.

System programmer response: None.

GFSA982E **The program** *d_digit1* **is not registered on** *string*.

Explanation: The requested program *d_digit1* is not registered on host *string*.

System action: The Network File System Client request fails.

Operator response: None.

System programmer response: Verify that the network connection to the NFS Server is operational and that the NFS Server is functional.

GFSA983E **Remote host does not have AF_INET interface.**

Explanation: The remote host does not support the desired IP interface.

System action: The Network File System Client request fails.

Operator response: None.

System programmer response: Verify that the requested IP version is operational.

GFSA984E **Error in authunix_create.**

Explanation: The Network File System Client was unable to create an Auth_sys security connection to the NFS server.

System action: The Network File System Client request fails.

Operator response: None.

System programmer response: Verify that the system is properly configured.

GFSA985E **Cannot get local host name**

Explanation: The Network File System client utility was unable to determine the NFS Server host name for the specified mount point.

System action: The Network File System Client utility request fails.

Operator response: None.

System programmer response: None.

GFSA986I The z/OS Network File System client utility was unable to get the attribute information for the specified mount point.

Explanation: The Network File System client utility was unable to determine the NFS Server host name for the specified mount point.

System action: The Network File System Client utility request fails.

Operator response: None.

System programmer response: None.

GFSA987I Directory *string* is not a supported Network File System type of directory.

Explanation: The specified directory path *string* is not a Network File System mounted directory.

System action: The Network File System Client request fails.

User response: Correct the directory path specification.

GFSA988I Remote host does not have AF_INET6 interface.

Explanation: The remote host does not support the desired IP interface string.

System action: The Network File System Client switches to using AF_INET (Internet Protocol version 4) and continues request processing.

Operator response: None.

System programmer response: Verify that the requested IP version is operational.

GFSA990I *text*

Explanation: The LOGON or FILE SECURITY user exit routine returned the message *text* to the NFS server.

System action: The Network File System server processing continues.

User response: No action required.

GFSA991E (*procname*) MESSAGE FORMAT FROM USER EXIT ROUTINE(S) IS INCORRECT. USER *text* EXIT ROUTINE(S) HAS(HAVE) ENDED.

Explanation: The *test* exit routine returned a message with an incorrect length.

In the message text:

procname
The name of the start procedure.

text
The value of *text* can be either `LOGON AND FILE SECURITY` or `FILE SECURITY`.

System action: The NFS server processing continues without the user exit routine(s).

Operator response: Record the operator console message and notify the NFS system programmer.

System programmer response: Correct the user exit routine(s), relink the user exit routine(s), and restart the NFS server.

GFSA996E (*procname*) INSTALLATION DEFAULT TRANSLATION TABLE CANNOT BE INITIALIZED.

Explanation: The NFS ends because one of these conditions happens during NFS startup:

- The NFSXLAT DD statement is not coded in the NFS startup catalog procedure, and the **xlat** processing attribute is specified.
- The translation table data set defined in the NFSXLAT DD statement is not a PDS or PDSE.
- The translation table specified in the translation table data set cannot be found.
- The translation table contained in the translation table data set is in an incorrect format.

In the message text:

procname The name of the start procedure.

System action: NFS processing stops.

System programmer response: Take the following actions before restarting the NFS:

- Code the NFSXLAT DD statement in the NFS startup catalog procedure correctly.
- Allocate the translation table data set defined in the NFSXLAT DD statement.
- Make sure the translation table in the translation table data set exists.
- Make sure the translation table specified in the translation table data set is in the correct format.

GFSA997I ERROR IN READING TRANSLATION TABLE, *text*.

Explanation: The NFS tried to read the translation table, *text*, during the mount but was unsuccessful. The value of *text* is the name of the member that contains the translation table. The reason might be one of these causes:

- The NFSXLAT DD statement is not coded in the NFS startup catalog procedure.
- The NFSXLAT DD statement is not coded correctly in the NFS startup catalog procedure.

- The translation table data set defined in the NFSXLAT DD statement is not a PDS or PDSE.
- The translation table specified in the translation table data set cannot be found.
- The format of the translation table contained in the translation table data set is not valid.

System action: NFS processing stops.

User response: This error occurs during the mount operation. Contact your system programmer for the correct member name for the translation table.

GFSA998I TRANSLATION TABLE *text* IS LOADED.

Explanation: The NFS loaded the translation table, *text*, successfully.

System action: NFS processing continues.

GFSA999I Error during rebuild of translation table, *text1*.

Explanation: During NFS server startup, the server detected an error during rebuild of translation table, *text1*.

System action: The NFS skips restoring the current MHDB record and continues processing.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Keep the existing z/OS NFS server traces and contact IBM Support.

GFSA1000E (*procname*) Invalid DD name specified in the *text* command.

Explanation: This message is issued when an operator enters an unacceptable (invalid) DD name in the ADDDS or FREEDS command.

text The name of the operator command used.

System action: The system ignores the command.

Programmer response: None

Operator response: Reenter the command with the correct DD name.

System programmer response: None

GFSA1001I (*procname*) Use any of the DD names: *text1*, *text2*, *text3*, *text4*.

Explanation: This message is issued when an operator enters an unacceptable (invalid) DD name in the ADDDS or FREEDS command. It follows message GFSA1000E.

text1 - text4 The list of valid MHDB and LDB DD names..

System action: The system ignores the command.

Programmer response: None

Operator response: Reenter the command with the correct DD name.

System programmer response: None

GFSA1002I (*procname*) DDNAME=*text* is already freed.

Explanation: This message is issued when an operator issues a FREEDS command for a DD name which is already freed.

text The DD name for which the request was issued.

System action: The system ignores the command.

Programmer response: None

Operator response: None

System programmer response: None

GFSA1003E (*procname*) Unable to perform FREEDS command. The dataset *text1* for the DDNAME=*text2* is busy.

Explanation: This message is issued when an operator issues a FREEDS command for a DD name for which the associated data set is currently in use.

text1 The name of the data set associated with the DD name.

text2 The DD name for which the request was issued.

System action: The system ignores the command.

Programmer response: None

Operator response: Retry the command later.

System programmer response: None

GFSA1004I (*procname*) The data set *text1* for the DDNAME=*text2* is deallocated.

Explanation: This message is issued when an operator issues a FREEDS command and the command completes successfully.

text1 The name of the data set associated with the DD name.

text2 The DD name for which the request was issued.

System action: The system successfully deallocated the data set associated with the specified DD name.

Programmer response: None

Operator response: None

System programmer response: None

GFSA1005E (*procname*) Unable to *text1* the dataset *text2* for DDNAME=*text3*. Operation return code *digit1*, error code *digit2*, info code *digit3*.

Explanation: This message is issued when a file system error occurs when the system tries to allocate/deallocate the data set for the specified DD name in response to an ADDDS or FREEDS operator command.

text1 The name of the operator command used.

text2 The name of the data set associated with the DD name.

text3 The DD name for which the request was issued.

digit1 The return code from the MVS Dynamic Allocation function.

digit2 The error code from the MVS Dynamic Allocation function.

digit3 The information code from the MVS Dynamic Allocation function.

System action: The system ignores the operator command.

Programmer response: None

Operator response: Contact the system programmer

System programmer response: Check the characteristics of the data set and correct as appropriate. See *z/OS MVS Programming: Authorized Assembler Services Guide*, SA22-7608 for a description of the return codes, error codes, and reason codes returned by the MVS Dynamic Allocation function.

GFSA1006W (*procname*) *text1* switching is suspended because one of the data sets is unusable. To resume switching, repair/replace the unusable data set.

Explanation: This message is issued when one of the MHDB, or LDB, data sets is unusable and the system tries to switch between the two data sets during a cleanup operation.

text1 will be either "MHDB" or "LDB" depending on which database is experiencing the problem.

System action: The system suspends cleanup operations for the database. It will continue writing new records to the remaining data set.

Programmer response: None

Operator response: Contact the system programmer

System programmer response: Allocate a new, or repaired, data set for the database.

GFSA1007E (*procname*) The data set *text1* for the DDNAME=*text2* is not allocated because it does not have the correct characteristics.

Explanation: This message is issued when an operator issues an ADDDS command but the specified data set does not have the correct characteristics.

text1 The name of the specified data set.

text2 The DD name for which the request was issued.

System action: The command is ignored.

Programmer response: None

Operator response: Contact the system programmer

System programmer response: Reissue the ADDDS command specifying a data set with the correct characteristics.

GFSA1008E (*procname*) *text1* is not a valid data set specification or MVS data set name. Use the format DDNAME(DSNAME).

Explanation: This message is issued when an operator issues an ADDDS command with an incorrect command format or data set name specification.

text1 The data set specification used in the command.

System action: The command is ignored.

Programmer response: None

Operator response: Reissue the ADDDS command using the correct format and MVS data set name specification rules.

System programmer response: None

GFSA1009E (*procname*) Unable to perform the ADDDS command. The DDNAME=*text1* is not freed.

Explanation: This message is issued when an operator issues an ADDDS command but a data set is still allocated to the specified DD name.

text1 The DD name for which the request was issued.

System action: The command is ignored.

Programmer response: None

Operator response: Use the FREEDS command to break the current allocation (association) for the DD name and then reissue the ADDDS command.

System programmer response: None

GFSA1010E (*procname*) The specified data set *text1* cannot be allocated because it is already in use.

Explanation: This message is issued when an operator issues an ADDDS command but the specified data set is already in use for the other MHDB/LDB data set.

text1 The name of the specified data set.

System action: The command is ignored.

Programmer response: None

Operator response: Reissue the ADDDS command specifying a different data set.

System programmer response: None

GFSA1011I (*procname*) The data set *text1* is successfully allocated for the DDNAME=*text2*.

Explanation: This message is issued when the operator issued ADDDS command completed successfully.

text1 The name of the specified data set.

text2 The DD name for which the request was issued.

System action: The system successfully created the new allocation (association) between the DD name and the provided data set.

Programmer response: None

Operator response: None

System programmer response: None

GFSA1012W (*procname*) The *text1* will not be refreshed because one of the data sets is unusable.

Explanation: This message is issued when one of the MHDB, or LDB, data sets is unusable and the system tries to swap between the two data sets either during a regular resource time out or during a server shutdown operation.

System action: The server did not refresh the MHDB, or LDB state (that is, swap the data sets).

Programmer response: None

Operator response: Contact System Programmer

System programmer response: Correct the problem with the unusable data set. The cause for the problem should have been reported at the original time when the problem was detected by the server.

GFSA1013E (*procname*) The *cmdname* command cannot be executed because at least one *dsname* data set is currently deallocated or corrupted.

Explanation: This message is issued when the operator issued a **SWAPMHDB** or **SWAPLDB** command when both data sets of the database are not available. One, or both, of the data sets is either deallocated or corrupted.

cmdname
is either **SWAPMHDB** or **SWAPLDB**, depending on which swap command was issued.

dsname is either **MHDB** or **LDB**, depending on which swap command was issued.

System action: The command is ignored.

Programmer response: None

Operator response: Allocate the missing data sets with the ADDDS command. Then repeat the SWAPMHDB command if necessary.

System programmer response: None

GFSA1014I (*procname*) *text* data sets swapping is launched.

Explanation: This message is issued when the operator issued the SWAPMHDB or SWAPLDB command. It indicates that the system has started the swapping process, which includes a cleanup procedure.

text1 will be either "MHDB" or "LDB" depending on which data sets were swapped.

System action: The swap process has started. Message GFSA1015I will be generated when it has completed.

Programmer response: None

Operator response: None

System programmer response: None

GFSA1015I (*procname*) *text1* is swapped. The current data set is *text2*.

Explanation: This message is issued when the operator issued the SWAPMHDB or SWAPLDB command. It follows message GFSA1014I and indicates that the system has completed the swapping process successfully.

text1 will be either "MHDB" or "LDB" depending on which data sets were swapped.

text2 This is the name of the currently active data set.

System action: The swap command was completed.

Programmer response: None

Operator response: None

System programmer response: None

GFSA1017E (*procname*) *dstype* data set *dsname* for the DDNAME=*ddname* cannot be opened, VSAM rc=*errcode* rsnc=*rsncode* LastOp=*opername*.

Explanation: This message is issued when the NFS server attempts to open the LDB or MHDB data set, but is unable to do so. This message identifies the data set type, the data set, return and reason codes and the code of the last operation. In the message text:

dstype	will be either "MHDB" or "LDB" depending on which data set was being opened.
dsname	This is the name of the data set that was being opened.
ddname	This is the ddname of the data set that was being opened.
errcode	VSAM error code
rsncode	VSAM reason code
opername	Last VSAM operation executed

See the VSAM publications for more details on the meaning of the VSAM codes.

System action: The data set was not opened and was flagged as unusable by the NFS server.

Programmer response: None

Operator response: Contact System Programmer

System programmer response: Check the data set characteristics to confirm that they match the required characteristics for the NFS database. Deallocate and reallocate a new or repaired data set with the proper characteristics.

GFSA1018E (*procname*) *dstype* data set *dsname* for the DDNAME=*ddname* does not have the proper characteristics: VSAM=*dsorg1* type=*type1* (expect *type2*) KeyOff=*keyoff1* (expect 0) KeySize=*keysize1* (Expect *keysize2*) RecSize=*recsize1* (Expect *recsize2*).

Explanation: This message is issued when the NFS server checks the LDB or MHDB data set characteristics before using the data set and discovers an error. This message identifies the data set type, the data set, the found characteristics, and the expected characteristics. In the message text:

dstype	will be either "MHDB" or "LDB" depending on which data set was being opened.
dsname	This is the name of the data set that was being opened.
ddname	This is the ddname of the data set that was being opened.
dsorg1	VSAM data set organization
type1	VSAM data set type
type2	Required VSAM data set type
keyoff1	VSAM key offset in the data set
keysize1	VSAM key size in the data set
keysize2	Required VSAM key size
recsize1	VSAM data set record size
recsize2	Required VSAM data set record size

System action: The data set was rejected and was flagged as unusable by the NFS server.

Programmer response: None

Operator response: Contact System Programmer

System programmer response: Check the data set characteristics to confirm that they match the required characteristics for the NFS database. Deallocate and reallocate a new or repaired data set with the proper characteristics.

GFSA1030E (*procname*) Network File System Server subtask is waiting for a reply from the *resource* (*FileSystemTypeName*)

Explanation: One of the NFS worker tasks has been waiting for a reply longer than specified by DelayDetectionTimeout parameter.

- *procname* - Network File System Server start up procedure name
- *resource* - The z/OS UNIX file system or MVS data set name
- *FileSystemTypeName* - The name of the type of the file system

System action: None.

Operator response: None.

System programmer response: If needed, collect the dump/trace regarding the type of file system that is being waited.

GFSA1031I (*procname*) Network File System Server subtask was waiting for reply from *sync_call* for the *resource* for *n* sec

Explanation: This message is issued if the message GFSA1030E has been previously issued.

procname
Network File System Server start up procedure name

sync_call
The name of the z/OS UNIX v_xxxxx function or MVS service call

resource The z/OS UNIX file system or MVS data set name

n the amount of time, in seconds, the Network File System Server waited for reply

System action: None.

Operator response: None.

System programmer response: None.

GFSA1032E (*procname*) The Delay Detection timeout value must be between 5 and 60 seconds

Explanation: An incorrect value for the Delay Detection timeout was specified in the MODIFY *mvsnfs*,DDTIMEOUT=x operator command.

- *procname* - Network File System Server start up procedure name

System action: None.

Operator response: Specify the correct value for the Delay Detection timeout.

System programmer response: None.

GFSA1033E (*procname*) There are many delays detected. There is more information in Network File System Log.

Explanation: There are more than five delays detected. The information about the new delays is printed in the Network File Sytem Log.

- *procname* - Network File System Server start up procedure name

System action: None.

Operator response: None.

System programmer response: If needed , collect the dump/trace regarding the type of file system being waited upon.

GFSA1034E (MVSNFS) The value of *text* site attribute of the Network File System Server is out of range. The value of *text* site attribute has been set to *d_digit*.

Explanation: The value of *text* z/OS NFS Server site attribute specified is out of range. The value of the *text* z/OS NFS Server site attributes is set to *d_digit*.

System action: z/OS NFS server processing continues.

Operator response: None.

System programmer response: The value of *text* may be adjusted in the attribute data set and the z/OS NFS server restarted.

Client messages

This is a listing of the messages that the NFS client generates. A message, explanation, and recommended action are supplied where applicable. Data is substituted for any part of a message shown here in *italics*.

Messages that appear on the z/OS operator's console for the NFS client will be in the following example format: **GFSC700I z/OS NETWORK FILE SYSTEM CLIENT (HDZ11VC) STARTED**

Table 62 shows the message format on the NFS client operators console.

Table 62. NFS client z/OS operators console message format

GFSC	Component identifier for the NFS client
700	A unique message number
I	Message type: **A** Action; the user must perform a specific action. **E** Eventual action; the user must perform an action when time is available. **I** Informational; no user action is required.
z/OS NETWORK FILE SYSTEM CLIENT (HDZ11VC) STARTED	Message text

Messages appear in the NFS client log data set in the same format as in the following example:

12:34:18 GFSC100E (E) CCXDR 11 XDR_DISP: RPC REQUEST (7F638E30) FAILED, RETURN VALUE -1 RETURN CODE 00000467 REASON CODE 5 (TIMED OUT)

Table 63 on page 381 shows the message format in the NFS client log data set.

Table 63. NFS client log data set message format

12:34:18	The time stamp (*hours*:*minutes*:*seconds*)
GFSC	Component identifier for the NFS client
100	A unique message number
E	Message type: **A** Action; the user must perform a specific action. **E** Eventual action; the user must perform an action when time is available. **I** Informational; no user action is required.
(E)	The message level: E (error), W (warning), or I (informational)
CCXDR 11	'CCXDR' is the last 5 characters of CSECT name. '11' is the message sequence within the function.
XDR_DISP	The first 8 characters of the function name

Table 63. NFS client log data set message format (continued)

RPC REQUEST (7F638E30) FAILED, RETURN VALUE -1 RETURN CODE 00000467 REASON CODE 5 (TIMED OUT)	Message text

The messages are listed in numerical order (the time stamp, message level, and programming support information are not shown).

Note: Messages GFSC098I and GFSC099I are intended for IBM support personnel to use when they are performing diagnosis. They do not indicate a problem with NFS, but do provide statistics on NFS processing. As such, an extensive number of GFSC098I and GFSC099I messages may be issued.

Table 64 shows common variables in the message text.

Table 64. Common variables

Variable	Meaning
retv	Decimal return value
retc	Decimal return code
rsnc	Decimal reason code
returncd	8-digit hexadecimal return code – see Chapter 20, “Return codes,” on page 417
reasoncd	8-digit hexadecimal reason code – see Chapter 21, “Reason codes,” on page 427
h_digit	8-digit hexadecimal address
d_digit	Decimal digits
dsname	Data set name
text	Place holder for long text of different lengths

GFSC098I

Explanation: This message is intended for IBM support personnel when they are performing diagnosis. They do not indicate a problem with NFS but do provide statistics on NFS processing. As such, an extensive number of GFSC098I messages may be issued.

System action: NFS continues processing.

Operator response: None.

System programmer response: Turn debugging off.

GFSC099I

Explanation: This message is intended for IBM support personnel when they are performing diagnosis. They do not indicate a problem with NFS but do provide statistics on NFS processing. As such, an extensive number of GFSC099I messages may be issued.

System action: NFS continues processing.

Operator response: None.

System programmer response: Turn debugging off.

GFSC100E RPC REQUEST FAILED, RETURN VALUE -1 RETURN CODE *returncd***h REASON CODE** *rsnc***h (***text***)**

Explanation: A remote procedure call (RPC) request failed.

In the message text:

rsnc
The reason code, *rsnc*, (hexadecimal) is the return code returned from TCP/IP.

text
The value of *text* is the failure reason.

returncd
For the explanation of return code *returncd*; see *z/OS UNIX System Services Messages and Codes*.

System action: The client processing continues.

User response: See the return code *returncd* in *z/OS UNIX System Services Messages and Codes* and reason code *rsnc* in the TCP/IP message manual.

GFSC101E *release* **NETWORK FILE SYSTEM SERVER REQUEST FAILED,** *release* **UNIX RETURN CODE** *returncd***h NETWORK FILE SYSTEM SERVER RETURN CODE** *retc***h (***text3***)**

Explanation: The NFS server failed the request from the client.

In the message text:

release
: system release (z/OS)

retc
: The return code, *retc*, (hexidecimal) is returned from the NFS server.

text
: The value of *text* is the failure reason for the request.

returncd
: For the explanation of return code *returncd*, see *z/OS UNIX System Services Messages and Codes*.

System action: The client processing continues.

User response: See the return code *returncd* in *z/OS UNIX System Services Messages and Codes* and the return code *retc* in the *Network File System Protocol Specification, RFC 1094* documentation.

GFSC102E **RPC REQUEST FAILED, RETURN VALUE -1 RETURN CODE** *returncd***h REASON CODE** *rsnc*

Explanation: A remote procedure call (RPC) request failed.

In the message text:

rsnc
: The reason code, *rsnc*, is the return code returned from TCP/IP.

returncd
: The explanation of return code *returncd* is in *z/OS UNIX System Services Messages and Codes*.

System action: The client processing continues.

User response: See the return code *returncd* in *z/OS UNIX System Services Messages and Codes* and the reason code *rsnc* in the TCP/IP message document.

GFSC103E *release* **NETWORK FILE SYSTEM SERVER REQUEST FAILED,** *release* **UNIX RETURN CODE** *returncd***h NETWORK FILE SYSTEM SERVER RETURN CODE** *retc*

Explanation: The NFS server failed the request from the client.

In the message text:

release
: system release (z/OS)

retc
: The return code, *retc*, is returned from the NFS server.

returncd
: For the explanation of return code *returncd* see *z/OS UNIX System Services Messages and Codes*.

System action: The client processing continues.

User response: See the return code *returncd* in *z/OS UNIX System Services Messages and Codes*and the return code *retc* in the *Network File System Protocol Specification, RFC 1094* documentation.

GFSC105E **READ FAILED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd***.**

Explanation: The system detected an error, *returncd*, while reading a block of data from a remote file.

System action: The read operation ends. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and the reason code *reasoncd* to determine corrective action.

GFSC106E **WRITE FAILED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd***.**

Explanation: The system detected an error, *returncd*, while writing a block of data to a remote file.

System action: The write operation ends. NFS client processing continues. The remote file might not be complete.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and the reason code *reasoncd* to determine corrective action.

GFSC107E **FLUSH FAILED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd***.**

Explanation: The system detected an error, *returncd*, while flushing cached data to a remote file during close processing.

System action: The write operation ends. NFS client processing continues. The remote file might not be complete.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and the reason code *reasoncd* to determine corrective action.

GFSC110E *text* **FAILED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd*

Explanation: The NFS client has detected an error in the function *text*.

System action: The request ended. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and the reason code *reasoncd* to determine corrective action.

System programmer response: Collect the detail trace diagnostics from the client and from the server.

GFSC200E **VFS_MOUNT FAILED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd***.**

Explanation: The **mount** command failed because of error *returncd*.

System action: The **mount** command ended abnormally. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code, *returncd* and the reason code *reasoncd* to determine the corrective action, and reissue the **mount** command.

GFSC201E **z/OS NETWORK FILE SYSTEM CLIENT DOES NOT SUPPORT SYNCHRONOUS MOUNT REQUEST.**

Explanation: The NFS client supports only an asynchronous mounts.

System action: The **mount** command ended with an error. NFS client processing continues.

User response: Reissue the **mount** command with the **asynchronous** option.

GFSC202E **A FILE SYSTEM WITH THE SAME NAME IS ALREADY MOUNTED.**

Explanation: The system cannot mount on an existing mount point.

System action: The **mount** command ended with an error. No mount point was established. NFS client processing continues.

User response: Reissue the **mount** command with a different mount point.

GFSC203E **PARSING MOUNT OPTION FAILED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd* **OPTION=**'*text*'**.**

Explanation: The mount option *text* was incorrectly specified.

System action: The **mount** command ended with an error. No mount point is established. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and see Table 73 on page 438 for more details of the reason code *reasoncd*. Correct the mount option, and reissue the **mount** command.

GFSC204E **VFS_UMOUNT FAILED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd***.**

Explanation: The **umount** or **unmount** command failed.

System action: The **umount** or **unmount** command ended with an error. The mount point might still exist. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and the reason code *reasoncd* to determine the corrective action. Correct it and reissue the **unmount** command.

GFSC205E **VFS_STATFS FAILED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd***.**

Explanation: The VFS_STATFS operation failed. The system detected an error, *returncd*, while trying to get the status of a remote file system.

System action: The VFS_STATFS operation ended with an error. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and the reason code *reasoncd* to determine the corrective action.

GFSC206E **VFS_SYNC FAILED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd***.**

Explanation: The VFS_SYNC operation failed. The system detected an error, *returncd* while flushing cached data of remote files.

System action: The VFS_SYNC operation ended with an error. The remote files might not be complete. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and the reason code *reasoncd* to determine the corrective action.

GFSC207E VFS_RECOVER FAILED, RETURN VALUE -1 RETURN CODE *returncd* REASON CODE *reasoncd*.

Explanation: The VFS_RECOVER operation failed. The system detected an error, *returncd*, while trying to recover from a previous abend.

System action: The VFS_RECOVER operation ended with an error. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and the reason code *reasoncd* to determine the corrective action. If the error occurs and the program is not in error, see the messages in the client log data sets for more information. Search problem-reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center. Provide all the printed output and copies of output data sets related to the problem.

GFSC208E VFS_PFSCTL FAILED, RETURN VALUE -1 RETURN CODE *returncd* REASON CODE *reasoncd*.

Explanation: The VFS_PFSCTL operation failed.

System action: The VFS_PFSCTL operation ended with an error. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd* and the reason code *reasoncd* to determine the corrective action. If the error occurs and the program is not in error, look at the messages in the client log data sets for more information. Search problem-reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center. Provide all the printed output and copies of output data sets related to the problem.

GFSC209E THE IP ADDRESS OF REMOTE HOST NAME, *hostname*, COULD NOT BE RESOLVED.

Explanation: The VFS_MOUNT operation failed. The mount processing failed when it tried to resolve the remote host name, *hostname*, to a dotted IP address.

System action: The **mount** command ended with an error. NFS client processing continues.

User response: Correct the remote host name, or use the dotted IP address of the remote host, and reissue the **mount** command.

GFSC210E NFS SERVER *hostname* DOES NOT SUPPORT NFS VERSION 3 PROTOCOL WHILE 'VERS(3)' WAS SPECIFIED

Explanation: The VFS_MOUNT operation failed. The mount processing failed because the server does not support NFS version 3 protocol, which the user requested with the **mount** parameter **vers(3)**.

System action: The **mount** command ended with an error. NFS client processing continues.

User response: Verify that the server actually does not support NFS version 3 protocol. Remove **vers(3)** from the **mount** parameter, and reissue the **mount** command.

GFSC211E NFS SERVER *hostname* DOES NOT SUPPORT 'AUTH_SYS' AUTHENTICATION.

Explanation: The VFS_MOUNT operation failed. The **mount** processing failed because the server does not do AUTH_SYS authentication. z/OS does not support other authentication, such as Kerberos.

System action: The **mount** command ended with an error.

User response: Verify that the server actually does not support AUTH_SYS authentication. Notify the server system administrator.

GFSC212E MOUNT FAILED BECAUSE OF CONVERSION SERVICE CONNECTION FAILURE CCSID *ccsid* RETURN CODE *returncd* REASON CODE *reasoncd*

Explanation: Coded character set identifiers (CCSID) specified in the **mount** command are not supported by the conversion service.

System action: The **mount** command ended abnormally. NFS client processing continues.

User response: See *z/OS UNIX System Services Messages and Codes* for more information about the return code *returncd*, and the reason code *reasoncd* to determine corrective action.

System programmer response: Check the availability of the specified CCSIDs.

GFSC213E REQUEST (*requestid*) : THE NFS SERVER *hostname* DOES NOT SUPPORT THE SPECIFIED 'VERS' AND/OR 'PROTO'.

Explanation: The **mount** request *requestid* failed because the user-specified **vers** and **proto** was not supported by the NFS server on *hostname*.

System action: The **mount** command ended with an error. NFS client continues processing.

User response: Either let the NFS client choose the compatible **vers** and **proto**, or determine the NFS server capabilities (by **orpcinfo**) and reissue the **mount** command with the correct **vers** or **proto**.

GFSC214E REQUEST (*requestid*) : THE NFS SERVER *hostname* DOES NOT HAVE NFS REGISTERED ON PORT 2049

Explanation: The request *requestid* failed because the NFS server on *hostname* did not register or use port 2049.

System action: The operation ended with an error. NFS client continues processing.

User response: Use **orpcinfo** to verify the server and correct the server.

GFSC216E THE NETWORK FILE SYSTEM SERVER *hostname* IS NOT COMPATIBLE WITH THE SPECIFIED 'PUBLIC' AND 'VERS' OPTIONS.

Explanation: z/OS NFS Client only supports the public mount keyword for NFS Servers which support NFS Version 4. It is possible that the *hostname* does not support NFS Version 4, or the mount command has the restrictive vers (version) keyword.

System action: The mount command ended with an error. No mount point is established. z/OS NFS Client processing continues.

Programmer response: None

Operator response: None

System programmer response: Reissue the mount command without the public option.

GFSC217E MOUNTING TO FILE SYSTEM SERVER *hostname* FAILS WITH CLIENTID_INUSE FROM *ip_address*.

Explanation: The NFS Server, *hostname*, denies z/OS NFS Client NFS V4 mount request because there is another NFS Client at *ip_address* with the same identification (clientid). z/OS NFS Client generates its NFS V4 identification as "client@domain#server".

System action: The mount command ended with an error. No mount point is established. z/OS NFS Client processing continues.

Programmer response: None

Operator response: None

System programmer response: Check and correct the z/OS host name and domain name, or check and correct the other system (at *ip_address*) host name and domain name.

GFSC218E THE NETWORK FILE SYSTEM SERVER *servername* IS NOT COMPATIBLE WITH THE SPECIFIED 'RPCBIND(N)' OPTION

Explanation: The identified NFS Server seems to only have an IPv6 address, but the rpcbind(n) option was specified. This option causes the z/OS NFS Client to send a PORTMAP Protocol request that is not compatible with Internet Protocol Version 6 (IPv6).

servername
: Name of the NFS Server to which this situation applies.

System action: The **mount** command ended with an error.

Programmer response: None.

Operator response: None.

System programmer response: Remove the **rpcbind(n)** option from the mount command and reissue the mount command.

GFSC219E ATTEMPTING TO ESTABLISH THE MOUNT PATH *pathname* AT THE NETWORK FILE SYSTEM SERVER *hostname*, BUT DETECTING A SYMBOLIC LINK LOOP AT *symlink*.

Explanation: NFS Client detects a symbolic link loop because it encounters the same *symlink* again when it attempts to establish a NFS Version 4 mountpoint. See “Accessing symbolic links on z/OS NFS version 4” on page 54 for details on NFS version protocol use.

System action: The mount operation fails. NFS Client processing continues.

Programmer response: None.

Operator response: Verify the *pathname* with the NFS Server System Administrator and reissue the mount command with the proper pathname.

System programmer response: None.

GFSC220E ATTEMPTING TO ESTABLISH THE MOUNT PATH *pathname* AT THE NETWORK FILE SYSTEM SERVER *hostname*, BUT THE OBJECT *objectname* IS NEITHER A DIRECTORY NOR A SYMBOLIC LINK.

Explanation: Only directories or symbolic links are allowed in the mount *pathname*. See “Accessing symbolic links on z/OS NFS version 4” on page 54 for details on NFS version protocol use.

System action: The mount operation fails. NFS Client processing continues.

Programmer response: None.

Operator response: Verify the *pathname* with the NFS Server System Administrator and reissue the mount command with the proper pathname.

System programmer response: None.

GFSC221I MOUNT PARAMETER LLOCK(N) IS IGNORED FOR NETWORK FILE SYSTEM PROTOCOL VERSION *version*.

Explanation: Local locking is always in effect for mounts to NFS protocol versions 2 and 3. Parameter llock(n) is ignored.

In the message text:

version
: The NFS protocol version specified in the mount command.

System action: The attribute is ignored and NFS Client processing continues.

Programmer response: If the mount command is in a script, correct it for future use.

Operator response: None.

System programmer response: None.

GFSC275E GSS API *rtn*() FAILED, USER IDENTIFIER *uid* GSS MAJOR STATUS *majstat* GSS MINOR STATUS *minstat*

Explanation: An RPCSEC_GSS request failed because of a failure returned by the GSS API referenced in the message. In the message text:

rtn The name of the GSS API that failed.

uid The user identifier of the user for whom this request was issued.

majstat The major status returned by the failing API.

minstat The minor status returned by the failing API.

System action: The NFS client retries this request once and fails the request if the retry also fails.

Operator response: Contact the system programmer.

System programmer response: Check the reported major status and minor status values in *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926 and take the specified actions.

GFSC276E GSS API gss_import_name() FAILED FOR SERVER, HOSTNAME *host* GSS MAJOR STATUS *majstat* GSS MINOR STATUS *minstat*

Explanation: An RPCSEC_GSS request failed to import the hostname of its target NFS server because of a failure returned by the GSS API gss_import_name( ).

In the message text:

host The DNS hostname of the NFS server to which this request is being directed.

majstat The major status returned by the failing API.

minstat The minor status returned by the failing API.

System action: The NFS client retries this request once and fails the request if the retry also fails.

Operator response: Contact the system programmer.

System programmer response: Check the reported major status and minor status values in *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926 and take the specified actions.

GFSC277E GSS API gss_init_sec_context() FAILED, SERVER HOSTNAME *host* USER IDENTIFIER *uid* GSS MAJOR STATUS *majstat* GSS MINOR STATUS *minstat*

Explanation: An RPCSEC_GSS request failed to create a security context because of a failure returned by the GSS API gss_init_sec_context( ).

In the message text:

host The DNS hostname of the NFS server with which this context was being created.

uid The user identifier of the user for whom this request was issued.

majstat The major status returned by gss_init_sec_context( ).

minstat The minor status returned by gss_init_sec_context( ).

System action: The NFS client retries this request once and fails the request if the retry also fails.

Operator response: Contact the system programmer.

System programmer response: Check the reported major status and minor status values in *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926 and take the specified actions.

GFSC278E KERBEROS CREDENTIALS EXPIRED FOR USER NAME *user* USER IDENTIFIER *uid*

Explanation: An RPCSEC_GSS request failed to acquire the GSS credentials for the user identified in the message, because the Kerberos credentials for the user don't exist. This can happen if the Kerberos credentials for the user have expired or the user has not done a "kinit".

In the message text:

user The RACF user name of the user for whom the credentials were being obtained.

uid The user identifier of the user for whom the credentials were being obtained.

System action: The NFS client fails all requests issued by this user.

Programmer response: The user needs to issue a "kinit" and retry the request.

Operator response: None.

System programmer response: None.

GFSC279E GSS API gss_unwrap() FAILED, USER IDENTIFIER *uid* GSS MAJOR STATUS *majstat* GSS MINOR STATUS *minstat*

Explanation: An RPCSEC_GSS request using the krb5p flavor, failed to decrypt the encrypted response from the NFS server, because of a failure returned by the GSS API gss_unwrap( ).

In the message text:

uid The user identifier of the user for whom this request was issued.

majstat The major status returned by gss_unwrap( ).

minstat The minor status returned by gss_unwrap( ).

System action: The NFS client retries this request once and fails the request if the retry also fails.

Operator response: Contact the system programmer.

System programmer response: Check the reported major status and minor status values in *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926 and take the specified actions.

GFSC280E NETWORK FILE SYSTEM CLIENT COULD NOT GET GSS CREDENTIALS FOR THE USER IDENTIFIER *uid* : GSS API *rtn*() FAILED WITH GSS MAJOR STATUS *majstat* GSS MINOR STATUS *minstat*

Explanation: The NFS client could not get GSS credentials for the user identified in the message, because of a failure returned by the GSS API referenced in the message.

In the message text:

uid The user identifier of the user for whom this request was issued.

rtn The name of the GSS API that failed.

majstat The major status returned by the failing API.

minstat The minor status returned by the failing API.

System action: On experiencing this failure for a regular user the NFS client retries this request once and it fails the request if the retry also fails.

Operator response: Contact the system programmer.

System programmer response: Check the reported major status and minor status values in *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926 and take the specified actions.

GFSC281E NETWORK FILE SYSTEM CLIENT COULD NOT LOAD THE KERBEROS DLL:RETURN CODE *rc*

Explanation: The NFS client could not dynamically load the Kerberos runtime library because of a failure returned by the API krb5_dll_load( ). In the message text:

rc The error code returned by krb5_dll_load().

System action: RPCSEC_GSS activity is not supported by NFS client.

Operator response: Contact the system programmer.

System programmer response: Refer to the documentation on krb5_dll_load() in *z/OS Integrated Security Services Network Authentication Service Programming*, SC24-5927 and take the specified actions.

GFSC282E NETWORK FILE SYSTEM CLIENT COULD NOT CREATE KERBEROS CONTEXT:RETURN CODE *rc*

Explanation: The NFS client could not create a Kerberos Context because of a failure returned by the API krb5_init_context(). In the message text:

rc The error code returned by krb5_init_context().

System action: RPCSEC_GSS activity is not supported by NFS client.

Operator response: Contact the system programmer.

System programmer response: Refer to the documentation on krb5_init_context() in *z/OS Integrated Security Services Network Authentication Service Programming*, SC24-5927 and take the specified actions.

GFSC283E NETWORK FILE SYSTEM CLIENT COULD NOT FIND KERBEROS CREDENTIAL CACHE FOR THE USER IDENTIFIER *uid* : USS API *rtn*() FAILED WITH RETURN CODE *majstat* REASON CODE *minstat*

Explanation: The NFS client could not find the user's Kerberos credential cache in the directory /var/skrb/creds because of a failure the message. In the message text:

uid The user identifier of the user for whom this request was issued.

rtn The name of the USS API that failed.

majstat The return code returned by the failing API.

minstat The reason code returned by the failing API.

System action: The NFS client fails RPCSEC_GSS requests issued by the user identified in the message.

Operator response: Contact the system programmer.

System programmer response: Ensure that the user has done "kinit" and check the reported Return Code and Reason Code values in the z/OS UNIX System Services Messages and Codes guide, SA22-7807 and take the specified actions.

GFSC284I NETWORK FILE SYSTEM CLIENT COULD NOT GET GSS CREDENTIALS FOR THE NFS CLIENT: GSS API *rtn*() FAILED WITH GSS MAJOR STATUS *majstat* GSS MINOR STATUS *minstat*

Explanation: The NFS client could not get GSS credentials for its Kerberos Principal; mvsnfsc , because of a failure returned by the GSS API referenced in the message. In the message text:

rtn The name of the GSS API that failed.

majstat The major status returned by the failing API.

minstat The minor status returned by the failing API.

System action: On experiencing this failure RPCSEC_GSS activity is not supported by the NFS client for any user.

Operator response: Contact the system programmer.

System programmer response: Check the reported major status and minor status values in *z/OS Integrated Security Services Network Authentication Service Administration*, SC24-5926 and take the specified actions.

GFSC300E MISSING LEFT PARENTHESIS IN *text* KEYWORD.

Explanation: The specified keyword, *text*, is missing a left parenthesis.

System action: NFS client processing stops if the error occurs in the NFS client installation parameter. The **mount** command failed if the error is in the **mount** parameter.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: If the error is in the **mount** parameter, correct the parameter and reissue the **mount** command. If the error occurs in the NFS client installation parameter, correct the parameter, stop z/OS UNIX, and restart z/OS UNIX.

GFSC301E PARSE FAILED ON NUMERIC FIELD FOR *text* KEYWORD.

Explanation: The specified keyword, *text*, contains alphabetic data in a numeric field.

System action: NFS client processing stops if the error is in the NFS client installation parameter. The **mount** command failed if the error is in the **mount** parameter.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: If the error is in the **mount** parameter, correct the parameter and reissue the **mount** command. If the error occurs in the NFS client installation parameter, correct the parameter, stop z/OS UNIX, and restart z/OS UNIX.

GFSC302E MISSING RIGHT PARENTHESIS IN *text* KEYWORD.

Explanation: The specified keyword, *text*, is missing a right parenthesis.

System action: NFS client processing stops if the error is in the NFS client installation parameter. The **mount** command failed if the error is in the **mount** parameter.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: If the error is in the **mount** parameter, correct the parameter and reissue the **mount** command. If the error occurs in the NFS client installation parameter, correct the parameter, stop z/OS UNIX, and restart z/OS UNIX.

GFSC303E THE SPECIFIED VALUE *d_digit1* IS NOT IN THE RANGE OF *d_digit2* TO *d_digit3* FOR *text* KEYWORD.

Explanation: The value *d_digit1* specified in the keyword *text* must be between the minimum value, *d_digit2*, and the maximum value, *d_digit3*.

System action: NFS client processing stops if the error is in the NFS client installation parameter. The **mount** command failed if the error is in the **mount** parameter.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: If the error is in the **mount** parameter, correct the parameter and reissue the **mount** command. If the error occurs in the NFS client installation parameter, correct the parameter, stop z/OS UNIX, and restart z/OS UNIX.

GFSC304E PARSE FAILED ON ALPHABETIC FIELD FOR *text* KEYWORD.

Explanation: The specified keyword, *text*, contains numeric data for an alphabetic field.

System action: NFS client processing stops if the error is in the NFS client installation parameter. The **mount** command failed if the error is in the **mount** parameter.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: If the error is in the **mount** parameter, correct the parameter and reissue the **mount** command. If the error occurs in the NFS client installation parameter, correct the parameter, stop z/OS UNIX, and restart z/OS UNIX.

GFSC305E INCORRECT OPTION *text1* SPECIFIED FOR *text2* KEYWORD, VALID OPTION IS Y OR N.

Explanation: An incorrect option, *text1*, was specified for the keyword *text2*.

System action: NFS client processing stops if the error is in the NFS client installation parameter. The **mount** command failed if the error is in the **mount** parameter.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: If the error is in the **mount** parameter, correct the option for the keyword *text2* and reissue the **mount** command. If the error occurs in the NFS client installation parameter, correct the option for the keyword *text2*, stop z/OS UNIX, and restart z/OS UNIX.

GFSC307E *text* IS AN INCORRECT KEYWORD FOR MOUNT PARAMETER.

Explanation: The value of *text* can be specified only as an installation parameter.

System action: The **mount** command failed.

System programmer response: Correct the **mount** parameter keyword *text*.

GFSC308E XLAT(Y) CANNOT BE SPECIFIED AS A MOUNT PARAMETER WHEN TAG OPTION IS ALSO SPECIFIED.

Explanation: The **mount** request with the *requestid* code failed because the user specified both the **xlat(Y)** and the **tag** options together. This is not allowed.

System action: The **mount** command ended with an error. NFS client continues processing.

User response: Either specify **xlat(Y)** to have the NFS client do text translation based on the **cln_ccsid** and **srv_ccsid** values, or specify the **tag** option with the correct coded character set identifier (CCSID) to have the translation done by the logical file system (LFS) based on the CCSID in the **tag** option.

GFSC309E UNKNOWN KEYWORD ENCOUNTERED AROUND POSITION *d_digit*.

Explanation: The keyword specified in position *d_digit* is not a valid keyword.

System action: NFS client processing stops if the error is in the NFS client installation parameter. The **mount** command failed if the error is in the **mount** parameter.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: If the error is in the **mount** parameter, correct the keyword and reissue the **mount** command. If the error occurs in the NFS client installation parameter, correct the keyword, stop z/OS UNIX, and restart z/OS UNIX.

GFSC310I READAHEAD AND DELAYWRITE OPTIONS WILL BE IGNORED AS DATACACHING IS OFF.

Explanation: The keywords **readahead** and **delaywrite** are ignored because **datacaching** has been set to `OFF`.

System action: NFS client processing continues.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Check the parameters to make sure that **datacaching** should be `OFF`.

GFSC311I CLN_CCSID AND SRV_CCSID WILL BE IGNORED AS XLAT OPTION IS OFF.

Explanation: The keywords **cln_ccsid** and **srv_ccsid** are ignored because **xlat** has been set to `OFF`.

System action: NFS client processing continues.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Check the parameters to make sure that **xlat** should be `OFF`.

GFSC312I ACREGMIN, ACREGMAX, ACDIRMIN, AND ACDIRMAX OPTIONS WILL BE IGNORED AS ATTRCACHING IS OFF.

Explanation: The keywords **acregmin**, **acregmax**, **acdirmin**, and **acdirmax** are ignored because **attrcaching** has been set to `OFF`.

System action: NFS client processing continues.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Check the parameters to make sure that **attrcaching** should be `OFF`.

GFSC313I RETRANS OPTION WILL BE IGNORED AS HARD OPTION IS ON.

Explanation: The keyword **retrans** is ignored because **hard** has been set to `ON`.

System action: NFS client processing continues.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Check the parameters to make sure that **hard** should be `ON`.

GFSC315E ERROR ENCOUNTERED WHILE PARSING MOUNT PATH, REASON CODE *reasoncd*.

Explanation: The specified mount path is not correct.

System action: The **mount** command failed.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Messages and Codes* for a description of the reason code *reasoncd*. Correct the mount path and reissue the **mount** command.

GFSC317E ERROR ENCOUNTERED WHILE PARSING HOSTNAME, REASON CODE *reasoncd*.

Explanation: The specified host name is not correct.

System action: The **mount** command failed.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Messages and Codes* for a description of the reason code *reasoncd*. Correct the mount path and reissue the **mount** command.

GFSC318E READ FAILED FOR NETWORK FILE SYSTEM CLIENT MOUNT PARAMETERS.

Explanation: An error occurred while the NFS client was processing the **mount** parameters. This message follows other messages, GFSC3*xx*E, that describe the error in more detail.

System action: The **mount** command failed.

System programmer response: Correct the **mount** parameter options, and reissue the **mount** command.

GFSC319E *text* IS AN INCORRECT KEYWORD FOR NETWORK FILE SYSTEM CLIENT INSTALLATION PARAMETER.

Explanation: The value of *text* can be specified only as a **mount** parameter.

System action: The NFS client processing stops.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Correct the NFS client installation parameter, stop z/OS UNIX, and restart z/OS UNIX.

GFSC320E INCORRECT OPTION *text* SPECIFIED FOR DELIM KEYWORD, VALID OPTION IS BINARY, CR, CRLF, CRNL, LF, LFCR, OR NL.

Explanation: An incorrect option, *text*, has been specified for the keyword **delim**.

System action: NFS client processing stops if the error is in the NFS client installation parameter. The **mount** command failed if the error is in the **mount** parameter.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: If the error is in the **mount** parameter, correct the option for the keyword **delim** and reissue the **mount** command. If the error occurs in the NFS client installation parameter, correct the option for the keyword **delim**, stop z/OS UNIX, and restart z/OS UNIX.

GFSC500I CLIENT LOG DATA SET, *text*, FLUSHED.

Explanation: The data buffer of the active client log data set, *text*, was flushed to disk.

In the message text:

text The value of *text* is the associated data set name of the active client log data set.

System action: NFS client processing continues.

GFSC501I CLIENT LOG DATA SET *text* RE-INITIALIZED.

Explanation: The error log data set is reinitialized.

In the message text:

text The value of *text* is the associated data set name of the active log data set.

System action: NFS client processing continues.

GFSC502E CANNOT OPEN CLIENT LOG DATA SET, *text1*, *text2*.

Explanation: The NFS client failed to open the client log data set.

In the message text:

text1 The value of *text1* is the data definition (DD) associated with the client log data set that cannot be opened.

text2 The value of *text2* is the failure reason of the C function, **fopen**.

System action: NFS client processing stops.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Fix the client log data set and reply to the z/OS UNIX message to restart the NFS client.

GFSC503E CLIENT LOGGING ENDED.

Explanation: The NFS client failed to manipulate the client log data set. See the previous operator console message for the failure reason.

System action: NFS client processing continues.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Fix the client log data set, stop the NFS client, and reply to the z/OS UNIX message to restart the NFS client.

GFSC504I CLIENT LOG DATA SET SWITCHED TO *text*.

Explanation: A `no space` or an I/O error condition was detected while the system was writing to the client log data set. NFS client logging switched to the other log data set, *text*.

System action: NFS client processing continues.

GFSC505E MISSING DD STATEMENT OR INCORRECT DATA SET ORGANIZATION FOR LOG DATA SET.

Explanation: The error log data set has either an incorrect data set organization or a missing data definition (DD) statement.

System action: NFS client processing stops if the error occurred during initialization. NFS client processing continues with client logging ended if the error occurred after initialization.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Correct the data definition statement or data set organization for the error log data set, stop the NFS client, and reply to the z/OS UNIX message to restart the NFS client.

GFSC506E *text1* FAILED FOR *text2*, *text3*.

Explanation: The NFS client failed to manipulate the client log data set, *text2*. The value of *text3* is the failure reason for the C function *text1*.

System action: NFS client processing continues.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Fix the client log data set, stop the NFS client, and reply to the z/OS UNIX message to restart the NFS client.

GFSC510E NETWORK FILE SYSTEM CLIENT CTRACE INITIALIZATION FAILED BECAUSE MACRO *function* HAD RETURN CODE=*rc* REASON CODE=*rsn*.

Explanation: NFS client CTRACE initialization failed to create the data space or to register to the component trace services. The specified macro ended with a non-zero return code. Explanations of the return and reason codes from the specified macro can be found in *z/OS MVS Programming: Authorized Assembler Services Reference ALE-DYN*.

In the message text:

procname
The name of the start procedure.

function The name of the macro with a non-zero return code.

rc The value of the return code.

rsn The value of the reason code.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Check the reported return/reason codes and take the specified actions.

GFSC700I z/OS NETWORK FILE SYSTEM CLIENT *fmid* STARTED

Explanation: The NFS client is initialized and ready to process NFS requests.

In the message text:

fmid The FMID of the NFS client; for example, (HDZ11VC).

System action: NFS client processing continues.

GFSC701I z/OS NETWORK FILE SYSTEM CLIENT SHUTDOWN IN PROGRESS.

Explanation: NFS client shutdown processing has started.

System action: NFS client shutdown processing continues.

GFSC702I z/OS NETWORK FILE SYSTEM CLIENT SHUTDOWN COMPLETE.

Explanation: The NFS client has completed shutdown processing.

System action: The NFS client and its associated subtasks have ended.

GFSC703E z/OS NETWORK FILE SYSTEM CLIENT INITIALIZATION FAILED: NETWORK FILE SYSTEM CLIENT IS ALREADY STARTED.

Explanation: Only one NFS client can be started on an MVS system.

System action: This NFS client initialization ends.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: A prior NFS client session has not ended. UNIX end processing should have ended the NFS client colony address space. Collect installation parameters, dumps, and NFS client log data sets, and contact the IBM Support Center. Use the **force** command to end the NFS client colony address space and then restart UNIX.

GFSC704E z/OS NETWORK FILE SYSTEM CLIENT INITIALIZATION FAILED: DOWN LEVEL SECURITY PRODUCT.

Explanation: Resource Access Control Facility (RACF) is down level.

System action: NFS client processing ends.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Check to determine the required RACF level.

GFSC705E z/OS NETWORK FILE SYSTEM CLIENT INITIALIZATION FAILED: MAIN STORAGE IS UNAVAILABLE.

Explanation: The NFS client was not able to allocate the necessary storage. The cause might be that the value specified in the **bufhigh** attribute is too large or the REGION size is too small.

System action: NFS client ends.

Operator response: Record the MVS console message and notify the system programmer.

System programmer response: The requested memory is not available during NFS client initialization. Take either or both of the following actions before restarting the NFS client.

- Increase the REGION size for the client procedure.
- Decrease the value specified for the **bufhigh** attribute of the FILESYSTYPE parameter in the BPXPRM*xx* parmlib member.

GFSC707E z/OS NETWORK FILE SYSTEM CLIENT INITIALIZATION FAILED: INCORRECT PARAMETER IN INSTALLATION PARAMETERS.

Explanation: The NFS client has detected an error in the installation parameters.

System action: NFS client processing ends.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: Check previous console messages prefixed with `GFSC`. Correct the parameter, stop z/OS UNIX, and restart z/OS UNIX.

GFSC708E z/OS NETWORK FILE SYSTEM CLIENT INITIALIZATION FAILED: CONVERSION SERVICE IS NOT INSTALLED OR NOT AVAILABLE. RETURN CODE *retc*, REASON CODE *rsnc*.

Explanation: Both the Unicode and character data representation architecture (CDRA) initialization requests failed.

System action: NFS client startup ends.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: See *z/OS MVS System Messages, Vol 2 (ARC-ASA)*, SA22-7632 for a description of the return code *retc* and reason code *rsnc* to determine the corrective action.

GFSC709E z/OS NETWORK FILE SYSTEM CLIENT INITIALIZATION FAILED: NOT STARTED IN A STANDALONE COLONY ADDRESS SPACE.

Explanation: The NFS client detected an error during initialization processing. The NFS client was started by some means other than a UNIX kernel.

System action: NFS client processing ends.

Operator response: Record the z/OS operator console message and notify the system programmer.

System programmer response: The NFS client must be initialized by a UNIX kernel.

GFSC710E z/OS NETWORK FILE SYSTEM CLIENT INITIALIZATION FAILED: ESTAE INITIALIZATION FAILED.

Explanation: The NFS client detected an error during initialization processing.

System action: NFS client processing ends.

Operator response: Collect any dumps and NFS client log data sets, and notify the system programmer.

System programmer response: Collect installation parameters, dumps, and NFS client log data sets, and contact the IBM Support Center.

GFSC711E z/OS NETWORK FILE SYSTEM CLIENT INITIALIZATION FAILED: *function_API* UNIX KERNEL SERVICE FAILED, RETURN CODE *retc* REASON CODE *rsnc*.

Explanation: The NFS client detected an error during the second phase of initialization processing.

The z/OS UNIX KERNEL *function_API* could be one of the following:

- BPX1SDD()
- BPX1PCT(PC#TDNames)
- v_reg()
- osi_kmsgget()

See *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803 for BPX1SDD() and BPX1PCT().

See *z/OS UNIX System Services Messages and Codes* for v_reg() and osi_kmsgget().

See *z/OS UNIX System Services File System Interface Reference* for a description of the return code *retc* and reason code *rsnc*.

System action: The NFS client ends.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: Collect installation parameters, dumps, and NFS client log data sets, and contact the IBM Support Center.

GFSC712E z/OS NETWORK FILE SYSTEM CLIENT INITIALIZATION FAILED: SOCKET CALL *function_API* FAILED, RETURN CODE *retc*.

Explanation: The NFS client has detected an error (*returncd*) during initialization processing. This error might be caused by z/OS UNIX and a TCP/IP connection failure.

The z/OS UNIX KERNEL *function_API* could be one of the following:

- BPX1HST()
- __iphost()
- __ipDomainName()

See *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803 for BPX1HST().

See *z/OS UNIX System Services Messages and Codes* for __iphost() and __ipDomainName().

See *z/OS UNIX System Services File System Interface Reference* for a description of the return code *retc*.

System action: NFS client ends.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: Collect installation parameters, dumps, and NFS client log data sets, and contact the IBM Support Center.

GFSC713E z/OS NETWORK FILE SYSTEM CLIENT *daemon* TERMINATED RV(*rtnval*, *rtncode*, *rsncode*)

Explanation: The NFS client has detected an error. The *daemon* daemon has ended with a return value of *rtnval*, return code of *rtncode*, and reason code of *rsncode*.

System action: The NFS client is terminated.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: Collect installation parameters, dumps, and NFS client log data sets, and contact the IBM Support Center.

GFSC714E z/OS NETWORK FILE SYSTEM CLIENT SVC DUMP FAILED RC=*returncd* RSNC=*reasoncd*

Explanation: A request to write an MVS SVC dump failed. See the description of the **sdump** macro in *z/OS MVS Programming: Authorized Assembler Services Reference LLA-SDU* for the meaning of return code *returncd* and reason code *reasoncd*.

System action: NFS client ends.

Operator response: Collect the MVS console log and NFS client log data sets, and notify the system programmer.

System programmer response: Collect installation parameters, dumps, and NFS client log data sets, and contact the IBM Support Center.

GFSC715E z/OS NETWORK FILE SYSTEM CLIENT *daemon* ESTAE EXIT FAILED *abendcd* RSN=*reasoncd*

Explanation: The NFS client recovery exit detected a recursive abend.

In the message text:

daemon
is the name of the NFS client daemon that experienced this failure.

abendcd
is the last abend code encountered by a secondary instance of the ESTAE exit routine.

reasoncd
is the abend reason code.

System action: The ESTAE processing stops and the NFS client terminates.

Operator response: Collect any dumps, the z/OS console log, and NFS client log data sets, and notify the system programmer.

System programmer response: Collect installation parameters, dumps, the z/OS console log, and NFS client log data sets, and contact the IBM Support Center.

GFSC716I z/OS NETWORK FILE SYSTEM CLIENT *daemon* RESTARTED.

Explanation: The NFS client detected that asynchronous daemon text *daemon* was stopped and restarted it.

System action: The NFS client daemon has restarted. NFS Client processing continues.

Operator response: None.

System programmer response: None.

GFSC717E z/OS NETWORK FILE SYSTEM CLIENT *daemon* ABNORMALLY TERMINATED *abendcd* RSN=*reasoncd*

Explanation: The NFS client daemon*daemon* abended. In the message text:

daemon
is the name of the NFS client daemon that experienced this failure.

abendcd
is the abend code encountered by the daemon.

reasoncd
is the abend reason code.

System action: The NFS client terminates.

Operator response: Collect any dumps, the z/OS console log and NFS client log data sets and notify the system programmer.

System programmer response: Collect installation parameters, dumps, the z/OS console log and NFS client log data sets, and contact the IBM Support Center.

GFSC718E z/OS NETWORK FILE SYSTEM CLIENT *daemon* UNABLE TO RETRY

Explanation: The NFS client daemon *daemon* abend recovery exit was unable to retry. The NFS client terminates.

System action: The NFS client terminates.

Operator response: Collect any dumps, the z/OS console log and NFS client log data sets and notify the system programmer.

System programmer response: Collect installation parameters, dumps, the z/OS console log and NFS client log data sets, and contact the IBM Support Center.

GFSC721E UNABLE TO SETUP ERROR RECOVERY (ESTAE), RETURN CODE *returncd*.

Explanation: The NFS client daemon or thread failed to setup error recovery. See *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG* for information about the **estae** macro return code *returncd*.

System action: If the error occurred in the daemon, the NFS client has initiated shutdown processing. If the error occurred in the thread, the associated operation ends and the NFS client processing continues.

Operator response: Collect any dumps, the z/OS console log, and NFS client log data sets, and notify the system programmer.

System programmer response: If the program is not in error, see the messages in the client log data sets for more information. Search problem-reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center. Provide all the printed output and copies of output data sets related to the problem.

GFSC722E A SOCKET COULD NOT BE CREATED, RETURN VALUE -1 RETURN CODE *returncd* REASON CODE *reasoncd*.

Explanation: The NFS client daemon or thread processing failed to create a socket for network communications.

System action: If the error occurred in the daemon, the NFS client has initiated shutdown processing. If the error occurred in the thread, the associated operation ends and the NFS client processing continues.

Operator response: Collect any dumps, the z/OS console log, and NFS client log data sets, and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Messages and Codes* for a description of return code *returncd* and reason code *reasoncd* to determine the corrective action. If the program is not in error, see the messages in the client log data sets for more information. Search problem-reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center. Provide all the printed output and copies of output data sets related to the problem.

GFSC723E z/OS NETWORK FILE SYSTEM CLIENT ABEND @ *module*+*offset*.

Explanation: The NFS client encountered a programming error. An SVC dump was issued to capture the diagnostic information.

System action: The NFS client has initiated shutdown processing.

System programmer response: Look at the messages in the client log data sets for more information. Search problem-reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center. Provide all the printed output and copies of output data sets related to the problem

GFSC724E UNABLE TO BIND A RESERVED PORT TO SOCKET *socketnum* **RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd***.**

Explanation: z/OS client failed to obtain a reserved port for socket *socketnum* for network communications.

System action: If the error occurred during z/OS client initialization, it the z/OS client has initiated shutdown processing. If the error occurred in the thread (owing to recent tcpip.profile change and z/OS client/Server restart), the associated operation ends and the NFS client processing continues.

System programmer response: Check that the tcpip.profile file has defined reserved (privileged) ports for the z/OS client. See *z/OS UNIX System Services Messages and Codes*, SA22-7807 for a description of return code *returncd* and reason code *reasoncd* to determine the corrective action. If the program is not in error, look at the messages in the client log data sets for more information. Search problem-reporting databases for a fix for the problem. If no fix exists, contact the IBM Support Center. Provide all the printed output and copies of the output data sets related to the problem.

GFSC725E SOCKET, *socketnum***, COULD NOT BE CLOSED, RETURN VALUE -1 RETURN CODE** *returncd* **REASON CODE** *reasoncd***.**

Explanation: While closing a socket *socketnum*, the system detected an error *returncd* was detected.

System action: The NFS client continues processing.

System programmer response: See *z/OS UNIX System Services Messages and Codes* for a description of return code *returncd* and reason code *reasoncd* to determine the corrective action. If the program is not in error, look at the messages in the client log data sets for more information. Search problem-reporting databases for a correction for the error. If no fix exists, contact the IBM Support Center. Provide all the printed output and copies of output data sets related to the problem.

GFSC726E THE PORTMAPPER OF THE SERVER *hostname* **DOES NOT RESPOND TO PMAP_GETMAPS**

Explanation: The **mount** request with code *requestid* failed because the user specified **proto(tcp)** and the *hostname* did not publish its registered remote program port for the remote procedure call (RPC).

System action: The **mount** command ended with an error. NFS client continues processing.

User response: Verify that the server does not respond to **orpcinfo**. Reissue the **mount** command without **proto(tcp)**.

GFSC727W THE PORTMAPPER OF THE SERVER *hostname* **DOES NOT RESPOND TO PMAP_GETMAPS. ATTEMPTING UDP RPC WITH NFS VERSION** *d_digit* **PROTOCOL.**

Explanation: The NFS client warns the user who mounts to *hostname* that the host might not have its portmapper running; the NFS client uses the user datagram protocol (UDP) remote procedure call (RPC). A UDP RPC is attempted with the NFS protocol.

System action: The **mount** command might or might not succeed. NFS client continues processing.

User response: Verify that the server does not respond to the code **orpcinfo**. If the **mount** command subsequently fails, then the host likely does not have the NFS server.

GFSC728E UNABLE TO CONNECT SOCKET=*socketno* **PORT=***portno1* **WITH THE SERVER hostname PORT=***portno2*

Explanation: The NFS client uses TCP/IP to connect socket *socketno* with port *portno1* to the server *hostname* with port *portno2*; however, the connect system call failed.

System action: The operation ended with an error. The NFS client processing continues.

User response: Verify that the NFS server on *hostname* is available and running. Verify that the TCP/IP subsystem on z/OS is available and running. If the **mount** command subsequently fails, then the host likely does not have the NFS server.

GFSC729W z/OS NFS CLIENT IS UNABLE TO RESERVE *d_digit* **SOCKETS, CURRENT MAXFILEPROC IS** *d_digit*

Explanation: The maximum number of socket descriptors that a process can have open concurrently has been exceeded. Increase the MAXFILEPROC parameter value in the BPXPRM*xx* member to bypass this problem. See *z/OS UNIX System Services Planning* for more information about specifying the MAXFILEPROC value.

System action: The operation ended with an error. The NFS client processing continues.

System programmer response: Increase the MAXFILEPROC parameter value in the BPXPRM*xx* member to bypass the problem.

GFSC734I NETWORK FILE SYSTEM NO LONGER SUPPORTS THE STOP COMMAND. PLEASE ISSUE: F OMVS,STOPPFS=NFS.

Explanation: Operator entered STOP MVSNFSC command to stop NFS Client. This command is no longer supported.

System action: The NFS client processing continues.

Programmer response: None.

Operator response: Issue 'F OMVS,STOPPFS=NFS' command to stop the NFS Client.

System programmer response: None.

GFSC735W NETWORK FILE SYSTEM CLIENT IS UNABLE TO USE STRINGPREP CONVERSION SERVICES.

Explanation: Calling the Unicode Service CUNLSTRP (stringprep) for the 3 stringprep profiles of CUNSTCSP (case sensitive), CUNSTCIS (case insensitive), and CUNSTMX1 (case mixed) failed.

System action: The z/OS NFS Client processing continues without the Unicode Stringprep Services for NFS V4 requests.

Programmer response: None

Operator response: None

System programmer response: Check and correct the Unicode Service Configuration. The z/OS NFS Client must be stopped and restarted after correcting the Unicode Stringprep so that it can pick up the new Unicode Configuration.

GFSC736E z/OS NETWORK FILE SYSTEM CLIENT initialization failed: default local host name or local domain not found.

Explanation: The NFS client has detected that the local z/OS installation has no local host name and/or no local domain name.

System action: The NFS client ends.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Planning* and *z/OS Communications Server: IP Configuration Guide* with regards to z/OS Network Configuration to take the corrective action.

GFSC737W Found *numconfig* TCPIP Stacks, but the maximum number of supported TCPIP Stacks is *maxstacks*.

Explanation: The z/OS NFS client found that the local z/OS installation has *numconfig* CINET TCP/IP Stacks, but it is greater than the maximum number of supported CINET TCP/IP Stacks of *maxstacks*. The network communication is fine. However, the NFS Version4 name-to-uid/gid mapping (and vice versa) may not be correct.

System action: The NFS client processing continues but it will use the default local domain name of the default CINET TCP/IP Stack for any communication on the TCP/IP Stack that is greater than *maxstacks*.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Planning* and *z/OS Communications Server: IP Configuration Guide* with regards to z/OS Network Configuration to take the corrective action.

GFSC738W z/OS NETWORK FILE SYSTEM CLIENT will use the default domain name *local_domain*.

Explanation: Due to the preceding error messages, the z/OS NFS client will use the default local domain name under CINET for the NFS Version4 name-to-uid/gid mapping (and vice versa).

System action: The NFS client processing continues but it will use the default local domain name of the default CINET TCP/IP Stack.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Planning* and *z/OS Communications Server: IP Configuration Guide* with regards to z/OS Network Configuration to take the corrective action.

GFSC739E z/OS UNIX service BPX1PCT(PC#DirGetHost) for TCPIP Stack *StackName* failed, return code *retc* reason code *rsnc*.

Explanation: z/OS NFS client is unable to retrieve the local z/OS TCP/IP *StackName* from the established network connection. See *z/OS UNIX System Services Messages and Codes* for a description of the return code *retc* and reason code *rsnc*.

System action: The NFS client processing continues but it will use the default local domain name of the default CINET TCP/IP Stack. The NFS Version4 name-to-uid/gid mapping (and vice-versa) may not be correct.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Planning* and *z/OS Communications Server: IP Configuration Guide* with regards to z/OS Network Configuration to take the corrective action.

GFSC740E z/OS UNIX service GETADDRINFO() for HOST *hostname* failed, return code *retc* reason code *rsnc*.

Explanation: The z/OS NFS client is unable to resolve the given *hostname* to the fully-qualified host name or an IP address.

System action: The NFS client processing continues but it will use the default local domain name of the default CINET TCP/IP Stack. The NFS Version4 name-to-uid/gid mapping (and vice versa) may not be correct.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Planning* and *z/OS Communications Server: IP Configuration Guide* with regards to z/OS Network Configuration to take the corrective action.

GFSC741E z/OS UNIX service w_ioctl(IOCC_GETSTACKS) for socket *socketnum* failed, return code *retc* reason code *rsnc*.

Explanation: The z/OS NFS client is unable to find the local CINET TCP/IP Stack associated with the locally connected TCP/IP socket.

System action: The NFS client processing continues but it will use the default local domain name of the default CINET TCP/IP Stack. The NFS Version4 name-to-uid/gid mapping (and vice versa) may not be correct.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Planning* and *z/OS Communications Server: IP Configuration Guide* with regards to z/OS Network Configuration to take the corrective action.

GFSC742W Socket socketnum has affinity to *numstack* Stacks

Explanation: The z/OS NFS client found that there is more than one local CINET TCP/IP stack with affinity to a locally connected TCP/IP socket.

System action: The NFS client processing continues but it will use the default local domain name of the default CINET TCP/IP Stack. The NFS Version4 name-to-uid/gid mapping (and vice versa) may not be correct.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Planning* and *z/OS Communications Server: IP Configuration Guide* with regards to z/OS Network Configuration to take the corrective action.

GFSC743E Unable to add TCPIP Stack *StackName*. The default domain name will be used.

Explanation: The z/OS NFS client is unable to find information about the recently activated TCPIP stack *StackName*.

System action: The NFS client processing continues but it will use the default local domain name of the default CINET TCP/IP Stack. The NFS Version4 name-to-uid/gid mapping (and vice versa) may not be correct.

Operator response: Collect any dumps and NFS client log data sets and notify the system programmer.

System programmer response: See *z/OS UNIX System Services Planning* and *z/OS Communications Server: IP Configuration Guide* with regards to z/OS Network Configuration to take the corrective action.

| **GFSC840I usage:** *text* **[-a] [-d] [-e] [-norpcbind] [host]**

Explanation: This is the usage for the **showmount** command. The value of *text* is the command as entered by the user. The valid options follow:

-a Display all mounts in the format *hostname:directory* from host NFS server

-d Display only directory names of all mounts from host NFS server

-e Display the list of exported directories from host NFS server

GFSC841E Unknown host *text*

Explanation: The user entered incorrect host address information, *text*.

System action: The system stops processing the command.

User response: Correct the syntax and reissue the command.

GFSC842E Cannot resolve local host name

Explanation: The local host name was not found.

System action: The system stops processing the command.

User response: Contact the system administrator to check the TCP/IP configuration.

GFSC843E Unknown flag '-*character*'

Explanation: An incorrect option, '-*character*', was specified.

System action: The system stops processing the command.

User response: Correct the syntax and reissue the command.

GFSC845I usage: *text input output*

Explanation: This is the usage for the **crnl2nl** and **nl2crnl** commands.

In the message text:

text The value of *text* is the command as entered by the user.

The valid parameters follow:

input Absolute path name of the input file to be converted.

output Absolute path name of the output file.

GFSC846E Cannot open input file, *text1***:***text2*

Explanation: The system cannot open the input file, *text1*.

In the message text:

text1 The value of *text1* is the input path name entered by the user.

text2 The value of *text2* is the failure information returned during an attempt to open the input file.

System action: The system stops processing the command.

User response: Check the input file, *text1*.

GFSC847E Cannot open output file, *text1***:***text2*

Explanation: The system cannot open output file, *text1*.

In the message text:

text1 The value of *text1* is the output path name entered by the user.

text2 The value of *text2* is the failure information returned during an attempt to open the output file.

System action: The system stops processing the command.

User response: Check the output file, *text1*.

GFSC848E Cannot read input file, *text1***:** *text2*

Explanation: The system cannot read the input file, *text1*.

In the message text:

text1 The value of *text1* is the input path name entered by the user.

text2 The value of *text2* is the failure information returned during an attempt to read the input file.

System action: The system stops processing the command.

User response: Check the input file, *text1*.

GFSC849E Cannot write output file, *text1***:** *text2*

Explanation: The system cannot write to the output file, *text1*.

In the message text:

text1 The value of *text1* is the output path name entered by the user.

text2 The value of *text2* is the failure information returned during an attempt to write to the output file.

System action: The system stops processing the command.

User response: Check the output file, *text1*.

GFSC850E Input path name cannot be equal to output path name.

Explanation: The input path name cannot be the same as the output path name.

System action: The system stops processing the command.

User response: Correct the syntax and reissue the command.

GFSC854I usage: *text* **[-crnzmi <mount point>]**

Explanation: This is the usage for the **nfsstat** command.

In the message text:

text The value of *text* is the command entered by the user.

The valid parameters follow:

-c Display both NFS and remote procedure call (RPC) statistics about the NFS client.

-n Display NFS statistics about the NFS client.

-r Display remote procedure call (RPC) statistics about the NFS client.

-z Initializes statistics to zero. This option is for use by the root user only and can be combined with any of the preceding options. Zero a particular set of statistics after printing them.

-m Display the name of each NFS-mounted file system.

-mi Displays the name of each NFS mounted file system and checks server's IP address validity (compare current server's IP address and IP address established at mount time) .

-m mount point
Display information about the NFS mounted file system on the specified mount point.

-mi mount point
Displays information for the NFS mounted file system on the specified mount point and checks server's IP address validity (compare current server's IP address and IP address established at mount time)

GFSC855E Must be a root user to issue '*character*' flag

Explanation: The option '*character*' can be issued only with the root authority.

System action: The system stops processing the command.

User response: Contact your system administrator to issue this command.

GFSC856E z/OS Network File System Client command, *text*, failed, return value -1 return code *returncd* reason code *reasoncd*

Explanation: The command *text* failed.

System action: The system stops processing the command.

User response: See *z/OS UNIX System Services Messages and Codes* for a description of the return code *returncd*. See Table 73 on page 438 for more information about the reason code *reasoncd*.

GFSC857I z/OS Network File System *type*:*status*

Explanation: The header message before the statistics sections

type Client or Server

status One of the following:
- "not available"
- "no statistics"
- The server startup procedure

System action: None.

Operator response: If the status is "not available" verify whether the Client or Server is started. If the status is "no statistics" then it is possible that the Client or the Server is about to terminated.

System programmer response: If the status is "not available" verify whether the Client or Server is started, and start the appropriate service.

GFSC858E Directory *text* not mounted.

Explanation: The directory *text* was not mounted.

System action: The system stops processing the command.

User response: Issue **nfsstat -m** to view the list of active mount points. If the mount point does not exist, contact the system administrator to mount the directory.

GFSC859E Incomplete command.

Explanation: The entered command is incomplete. The request of module's version (-v option) should be completed with the module name or 'a' for all modules.

System action: The system stops processing the command.

User response: Check the input command. Specify version details

GFSC860W Name Resolver indicates the Server=*servername* is at *ipaddr* [, *ipaddr*]

Explanation: The Name Resolver indicates that the ip address *ipaddr* for NFS server *servername* has been changed. In the message text:

servername
The name of NFS server.

ipaddr The new IP address of the NFS server.

System action: The system continues processing the command.

User response: Check the existing mounts related to this NFS server. Remount if needed.

GFSC869E Call to remote NFS Server failed. ERRP = *rpc_retvalue uss_retcode uss_rsncode*

Explanation: The NFS Client showmount utility encountered a failure attempting to send a request to the NFS Server. In the message text:

rpc_retvalue
The return value returned by RPC clnt_call( ).

uss_retcode
The return code returned by RPC clnt_call( ).

uss_rsncode
The reason code returned by RPC clnt_call( ).

System action: The showmount utility fails the request.

User response: Check network availability.

System programmer response: Check the reported rpc_retvalue in z/OS TCP/IP Programmer's Reference, and uss_retcode and uss_rsncode in *z/OS UNIX System Services Messages and Codes* and take the specified actions.

Client/Server Shared Messages

This is a listing of the messages generated by the NFS server or NFS client. Each message description gives an explanation and recommended actions where applicable. The system substitutes data for any part of a message shown here in *italics*.

Messages appear in the NFS server or NFS client log data sets in the same format as this example: **GFSN5001E Both z/OS Unix ACL and Mode pointers are NULL. Nothing converted.**

Table 65 on page 402 shows the message format for the NFS server or NFS client log data sets:

Table 65. Message format for the NFS server or NFS client log data sets

GFSN	The component identifier for the NFS server
5001	A unique message number
E	The message level: E (error), W (attention), or I (informational). The system programmer can use the message level to determine which type of messages are shown by specifying log=error, log=warn, or log=info.
Both z/OS Unix ACL and Mode pointers are NULL. Nothing converted.	The message text

The messages are listed in numerical order (the time stamp, message level, and programming support information are not shown).

1. A value of *h_digits* is a hexadecimal number, and *d_digits* is a decimal number. A value of *text* or *dsname* is variable text (such as a data set name).
2. For messages written to the console, the name of the start procedure is substituted for *procname*.

GFSN5001E Both z/OS Unix ACL and Mode pointers are NULL. Nothing converted.

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the z/OS UNIX to NFS ACL conversion function, but did not pass the location of any ACL, or permission mode bit, values to be converted.

System action: No ACLs are converted from z/OS UNIX to NFS ACLs. The request fails and normal processing continues.

Programmer response: None.

Operator response: None.

System programmer response: Capture a trace of the situation and report it to the IBM Support Center.

GFSN5002E Invalid z/OS UNIX ACL header: ***h_digit1*** **ACE count:** ***d_digit*****.**

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the z/OS UNIX to NFS ACL conversion function, but an invalid z/OS UNIX ACL header control block was received. This is the result of a program logic error.

In the message text:

h_digit1	The contents of the control block for IBM support personnel analysis.
d_digit	The number of USS ACL entries to be converted.

System action: No ACLs are converted from z/OS UNIX to NFS ACLs. The request fails and normal processing continues.

Programmer response: None.

Operator response: None.

System programmer response: Capture a trace of the situation and report it to the IBM Support Center.

GFSN5003E Unsupported Network File System ACL Who Type. Supported Values are: OWNER, GROUP, EVERYONE, User, Group. ACE: *d_digit* **, WType:** *h_digit1***, WId:** *h_digit2***, AType:** *h_digit3***, AFlag:** *h_digit4***, AMask:** *h_digit5***.**

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the z/OS UNIX to, or from, NFS ACL conversion function. In the process of merging the z/OS UNIX ACLs into the existing NFS ACLs, an NFS ACL entry with an unsupported WHO type was encountered. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with WHO types which are not supported by z/OS UNIX.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the NFS ACL contains entries which specify unsupported WHO types.

z/OS NFS only supports the following WHO types:

OWNER@
specifies the owner of the file or directory

GROUP@
specifies the group of the owner of the file or directory

EVERYONE@
specifies the base Other permissions for the file or directory

user@domain
specifies an explicit user identifier

group@domain
specifies an explicit user group identifier

In the message text: (primarily intended for IBM support personnel use):

d_digit	0-origin index of the NFS ACL entry containing the error.
h_digit1	hexadecimal value of the NFS ACL who type. (1-owner, 2-user, 3-owner group, 4-user group, 8-everyone)
h_digit2	hexadecimal value of the user, or group, id.
h_digit3	hexadecimal value of the NFS ACL type field (see RFC3530).
h_digit4	hexadecimal value of the NFS ACL flag field (see RFC3530).
h_digit5	hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL values and update them to comply with the z/OS NFS ACL management restrictions. See "Displaying and modifying remote file system access control lists" on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5004E Directory Default ACL delete is not allowed. This is only allowed for z/OS Network File System Client/Server connections.

Explanation: A z/OS NFS Client Directory Default ACL delete request was attempted but the remote NFS Server is not a z/OS system. Directory Default requests are only supported when the remote NFS Server is the z/OS NFS Server. Otherwise, Default ACLs must be managed using File Default ACL requests. Most non-z/OS platforms only have a single set of Default ACLs, not separate File and Directory Default ACLs. Therefore, z/OS NFS ACL management support uses the File Default ACL requests to manage the NFS Default ACLs. To avoid potential confusion, or even errors due to duplicate requests, Directory Default ACL requests are not supported.

System action: The request fails and normal processing continues.

Programmer response: Use the z/OS UNIX File Default ACL management requests, not the z/OS UNIX Directory Default ACL requests.

Operator response: None.

System programmer response: None.

GFSN5005E z/OS Unix Directory Default ACL entry does not match existing Network File System Default ACL entry.

Explanation: The z/OS NFS ACL conversion service encountered a UNIX System Services Directory Default ACL entry which does not match the existing NFS Default ACL entry. Management of mismatched Directory and File Default ACLs is only supported between the z/OS NFS Client and the z/OS NFS Server. The conversion request cannot be processed.

On the z/OS NFS Server this could occur when the z/OS UNIX File and Directory Default ACL entries do not match. z/OS NFS does not support remote management of such ACLs.

On the z/OS NFS Client this can occur when the Directory Default does not match the remote system's existing Default ACLs. Most non-z/OS platforms only have a single set of Default ACLs, not separate File and Directory Default ACLs. Therefore, z/OS NFS ACL management support uses the File Default ACL requests to manage the NFS Default ACLs. To avoid potential errors, in this scheme, Directory Default ACL requests are not supported.

System action: The request fails and normal processing continues.

Programmer response: Non-z/OS NFS Clients can not be used to manage z/OS UNIX directory Default ACLs when the File and Directory Defaults do not match. Either change them via local requests so that they match, or manage them locally. For managing non-z/OS NFS Default ACLs from the z/OS NFS Client, use z/OS UNIX System Services File Default ACL management requests.

Operator response: None.

System programmer response: None.

GFSN5006E Directory Default and File Default ACLs do not match. This is only allowed for z/OS Network File System Client/Server connections.

Explanation: The z/OS UNIX System Services Directory Default and File Default ACLs for this object do not match. Management of mismatched Directory and File Default ACLs is only supported between the z/OS NFS Client and the z/OS NFS Server. For other NFS Client/Server combinations, the z/OS Directory and File Defaults must match and z/OS File Default requests should be used for managing non-z/OS NFS ACLs. The conversion request cannot be processed.

On the z/OS NFS Server this could occur when the z/OS UNIX File and Directory Default ACL entries do not match. z/OS NFS does not support remote management of such ACLs.

On the z/OS NFS Client this can occur when the Directory Default does not match the remote system's existing Default ACLs. Most non-z/OS platforms only have a single set of Default ACLs, not separate File and Directory Default ACLs. Therefore, z/OS NFS ACL management support uses the File Default ACL requests to manage the NFS Default ACLs. To avoid potential errors, in this scheme, Directory Default ACL requests are not supported.

System action: The request fails and normal processing continues.

Programmer response: Non-z/OS NFS Clients can not be used to manage z/OS UNIX directory Default ACLs when the File and Directory Defaults do not match. Either change them via local requests so that they match, or manage them locally. For managing non-z/OS NFS Default ACLs from the z/OS NFS Client, use z/OS UNIX System Services File Default ACL management requests.

Operator response: None.

System programmer response: None.

GFSN5007E z/OS UNIX to Network File System ACL conversion error. USS ACL Type: *h_digit1* **ObjType:** *d_digit* **RsnCode:** *h_digit2***.**

Explanation: The z/OS UNIX to NFS ACL conversion function experienced an error. See the previous z/OS NFS error messages for details.

In the message text:

h_digit1
The z/OS UNIX ACL type (1-Access, 2-File Default, 3-Dir Default)

d_digit
The object type (1-file, 2-directory)

h_digit2
The error reason code. This is intended for IBM support personnel use only.

System action: No ACLs are converted from z/OS UNIX System Services to NFS ACLs. The request fails and normal processing continues.

Programmer response: See the previous z/OS NFS ACL conversion error message to identify the actual error that occurred.

Operator response: None.

System programmer response: None.

GFSN5008E Invalid Network File System ACL header: (*h_digit1***)->***h_digit2***.**

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function, but an error was detected in the NFS ACL header control block. This is the result of a program logic error.

In the message text:

h_digit1
A pointer to the control block for IBM support personnel analysis.

h_digit2
The contents of the NFS ACL control block header for IBM support personnel analysis.

System action: No ACLs are converted from NFS to z/OS UNIX ACLs. The request fails and normal processing continues.

Programmer response: None.

Operator response: None.

System programmer response: Capture a trace of the situation and report it to the IBM Support Center.

GFSN5009E Invalid Mask ACL. First Mask: *h_digit0***. ACE:** *d_digit***, WType:** *h_digit1***, WId:** *h_digit2***, AType:** *h_digit3***, AFlag:** *h_digit4***, AMask:** *h_digit5***.**

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. All of the NFS Mask ACL entries must have the same Mask (permission) value. The function encountered an NFS ACL entry with a different Mask value. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with Mask ACL entries with different Mask (permission) values. This is not supported by z/OS UNIX.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains Mask ACL entries with different Mask (permission) values. This is not supported by z/OS UNIX.

In the message text: (primarily intended for IBM support personnel use)

h_digit0
The hexadecimal value of the first NFS Mask ACL mask (permission) value.

d_digit
The 0-origin index of the NFS ACL entry containing the error.

h_digit1
The hexadecimal value of the NFS ACL who type. (1-owner, 2-user, 3-owner group, 4-user group, 8-everyone)

h_digit2
The hexadecimal value of the user, or group, id.

h_digit3
The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4
The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5
The hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See “Displaying and modifying remote file system access control lists” on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5010E Base Group Mask and Deny ACL mismatch: Group Mask: ***h_digit0*****. Deny ACE:** ***d_digit*****, WType:** ***h_digit1*****, WId:** ***h_digit2*****, AType:** ***h_digit3*****, AFlag:** ***h_digit4*****, AMask:** ***h_digit5*****.**

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. z/OS UNIX does not support storing a Mask ACL entry. Therefore, z/OS NFS dynamically computes the Mask ACL permission value from the Base Group Deny permissions. This requires that the Base Group Mask and Deny permissions must match. The function encountered an NFS Deny ACL entry whose Mask (permission) value does not match the Base Group Mask value. Mismatched values can not be supported by z/OS NFS. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with mismatched NFS Base Group Mask and Deny ACL Mask (permission) values. This is not supported by z/OS UNIX.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains mismatched NFS Base Group Mask and Deny ACL Mask (permission values. This is not supported by z/OS UNIX.

In the message text: (primarily intended for IBM support personnel use)

h_digit0
The hexadecimal value of the NFS Base Group Mask ACL entry mask field.

d_digit
The 0-origin index of the NFS Base Group Deny ACL entry containing the error.

h_digit1
The 0-origin index of the NFS Base Group Deny ACL entry containing the error.

h_digit2
The hexadecimal value of the user, or group, id.

h_digit3
The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4
The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5
The hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See “Displaying and modifying remote file system access control lists” on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5011E Access and Default Base ACL mismatch. BMask: *h_digit0***. MaskACE:** *yn***. ACE:** *d_digit***, WType:** *h_digit1***, WId:** *h_digit2***, AType:** *h_digit3***, AFlag:** *h_digit4***, AMask:** *h_digit5***.**

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. z/OS UNIX does not support storing separate Access and Default Base ACLs. Therefore, z/OS NFS uses the Access Base ACLs for both. The function encountered a mismatch between an NFS Access Base ACL entry and the NFS Default Base ACL entry. Mismatched values can not be supported by z/OS NFS. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with different NFS Access Base and NFS Default Base ACL values. This is not supported by z/OS NFS.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains entries which have different NFS Access Base and NFS Default Base ACL values. This is not supported by z/OS NFS.

In the message text: (primarily intended for IBM support personnel use)

h_digit0
The hexadecimal value of the first (Access or Default) NFS Base ACL entry mask field.

yn
'Y' - yes, this is a Mask entry. 'N' - no, this is not a Mask ACE (It is an Allow or Deny entry).

d_digit
The 0-origin index of the mismatched NFS Base ACL entry.

h_digit1
The hexadecimal value of the NFS ACL who type. (1-owner, 2-user, 3-owner group, 4-user group, 8-everyone).

h_digit2
The hexadecimal value of the user, or group, id.

h_digit3
The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4
The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5
The hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See “Displaying and modifying remote file system access control lists” on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5012E Invalid Mask ACL. BMask: *h_digit0***. ACE:** *d_digit***, WType:** *h_digit1***, WId:** *h_digit2***, AType:** *h_digit3***, AFlag:** *h_digit4***, AMask:** *h_digit5* **RsnCode:** *h_digit6***.**

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. All of the NFS Mask ACL entries must have the same Mask (permission) value. The function encountered a user or user group NFS ACL entry with a different Mask value. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with Mask ACL entries with different Mask (permission) values. This is not supported by z/OS UNIX.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains Mask ACL entries with different Mask (permission) values. This is not supported by z/OS UNIX.

In the message text: (primarily intended for IBM support personnel use)

h_digit0
The hexadecimal value of the NFS Base Group ACL entry mask field.

d_digit
The 0-origin index of the user or user group NFS ACL entry containing the error.

h_digit1
The hexadecimal value of the NFS ACL who type. (2-user, 4-user group)

h_digit2
The hexadecimal value of the user, or group, id.

h_digit3
The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4
The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5
The hexadecimal value of the NFS ACL mask field (see RFC3530).

h_digit6
The error reason code. This is intended for IBM support personnel use only.

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See “Displaying and modifying remote file system access control lists" on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5013E Unsupported Network File System ACL Type. Supported ACL Types: Allow or Deny. ACE: *d_digit*, **WType:** *h_digit1*, **WId:** *h_digit2*, **AType:** *h_digit3*, **AFlag:** *h_digit4*, **AMask:** *h_digit5*.

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. The function encountered an NFS ACL entry with an unsupported ACL Type. z/OS NFS only supports Allow and Deny NFS ACL Types. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with an NFS ACL Type which is not supported by z/OS NFS.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains entries which specify an NFS ACL Type which is not supported by z/OS NFS.

In the message text: (primarily intended for IBM support personnel use)

d_digit The 0-origin index of the NFS ACL entry containing the error.

h_digit1 The hexadecimal value of the NFS ACL who type. (1-owner, 2-user, 3-owner group, 4-user group, 8-everyone)

h_digit2 The hexadecimal value of the user, or group, id.

h_digit3 The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4 The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5 The hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See "Displaying and modifying remote file system access control lists" on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5014E Unsupported Network File System ACL Mask. Supported Mask values: *h_digit0*. **ACE:** *d_digit*, **WType:** *h_digit1*, **WId:** *h_digit2*, **AType:** *h_digit3*, **AFlag:** *h_digit4*, **AMask:** *h_digit5*.

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. The function encountered an NFS ACL entry with an unsupported Mask (permission) value. Only the supported (*h_digit0*) bits can be processed. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with an NFS ACL Mask value which is not supported by z/OS UNIX.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains entries which specify an NFS ACL Mask value which is not supported by z/OS UNIX.

In the message text: (primarily intended for IBM support personnel use)

h_digit0 The hexadecimal value of supported NFS ACL entry mask field bits (see RFC3530).

d_digit The 0-origin index of the NFS ACL entry containing the error.

h_digit1 The hexadecimal value of the NFS ACL who type. (1-owner, 2-user, 3-owner group, 4-user group, 8-everyone)

h_digit2 The hexadecimal value of the user, or group, id.

h_digit3 The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4 The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5 The hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See "Displaying and modifying remote file system access control lists" on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5015E Unsupported Network File System ACL Mask. Required Allow Mask values: ***h_digit0*****. ACE:** ***d_digit*****, WType:** ***h_digit1*****, WId:** ***h_digit2*****, AType:** ***h_digit3*****, AFlag:** ***h_digit4*****, AMask:** ***h_digit5*****.**

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. z/OS UNIX does not support all NFS Allow permission values. To accommodate this, some NFS Allow Mask ACL entry mask (permission) values (*h_digit0*) are required. Otherwise the Allow Mask entry can not be supported. The function encountered an NFS Allow Mask ACL entry in which the required mask (permission) bits were not all specified. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with an NFS Allow Mask ACL entry in which not all the required mask bits are specified. This is not supported by z/OS NFS.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains entries with an NFS Allow Mask ACL entry in which not all the required mask bits are specified. This is not supported by z/OS NFS.

In the message text: (primarily intended for IBM support personnel use)

h_digit0 The hexadecimal value of the required NFS Allow Mask ACL entry mask field bits.

d_digit The 0-origin index of the NFS ACL entry containing the error.

h_digit1 The hexadecimal value of the NFS ACL who type. (1-owner, 2-user, 3-owner group, 4-user group, 8-everyone)

h_digit2 The hexadecimal value of the user, or group, id.

h_digit3 The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4 The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5 The hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See "Displaying and modifying remote file system access control lists" on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5016E Unsupported Network File System ACL Allow Read Permissions. Required values are: ***h_digit0*****. ACE:** ***d_digit*****, WType:** ***h_digit1*****, WId:** ***h_digit2*****, AType:** ***h_digit3*****, AFlag:** ***h_digit4*****, AMask:** ***h_digit5*****.**

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. z/OS UNIX does not support all NFS Allow read permission values. To accommodate this, read permission requires specific (*h_digit0*) NFS Allow ACL entry mask (permission) values. Otherwise, the NFS Allow ACL entry can not be supported. The function encountered an NFS Allow ACL entry in which the required mask (permission) bits were not all specified. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with an unsupported combination of NFS Allow ACL read mask (permission) bits. This is not supported by z/OS NFS.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains entries with an unsupported combination of NFS Allow ACL read mask (permission) bits. This is not supported by z/OS NFS.

In the message text: (primarily intended for IBM support personnel use)

h_digit0 The hexadecimal value of the required NFS Allow ACL entry read mask field bits.

d_digit The 0-origin index of the NFS ACL entry containing the error.

h_digit1 The hexadecimal value of the NFS ACL who type. (1-owner, 2-user, 3-owner group, 4-user group, 8-everyone)

h_digit2 The hexadecimal value of the user, or group, id.

h_digit3 The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4 The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5 The hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See "Displaying and modifying remote file system access control lists" on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5017E Unsupported Network File System ACL Allow Write Permissions. Required values are: *h_digit0*. **ACE:** *d_digit*, **WType:** *h_digit1*, **WId:** *h_digit2*, **AType:** *h_digit3*, **AFlag:** *h_digit4*, **AMask:** *h_digit5*.

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. z/OS UNIX does not support all NFS Allow write permission values. To accommodate this, write permission requires specific (h_digit0) NFS Allow ACL entry mask (permission) values. Otherwise, the NFS Allow ACL entry can not be supported. The function encountered an NFS Allow ACL entry in which the required mask (permission) bits were not all specified. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with an unsupported combination of NFS Allow ACL write mask (permission) bits. This is not supported by z/OS NFS.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains entries with an unsupported combination of NFS Allow ACL write mask (permission) bits. This is not supported by z/OS NFS.

In the message text: (primarily intended for IBM support personnel use)

h_digit0 The hexadecimal value of the required NFS Allow ACL entry write mask field bits.

d_digit The 0-origin index of the NFS ACL entry containing the error.

h_digit1 The hexadecimal value of the NFS ACL who type. (1-owner, 2-user, 3-owner group, 4-user group, 8-everyone)

h_digit2 The hexadecimal value of the user, or group, id.

h_digit3 The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4 The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5 The hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See "Displaying and modifying remote file system access control lists" on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5018E Unsupported Network File System ACL Allow Execute Permissions. Required values are: *h_digit0*. **ACE:** *d_digit*, **WType:** *h_digit1*, **WId:** *h_digit2*, **AType:** *h_digit3*, **AFlag:** *h_digit4*, **AMask:** *h_digit5*.

Explanation: The z/OS NFS Client, or the z/OS NFS Server, called the NFS to z/OS UNIX ACL conversion function. z/OS UNIX does not support all NFS Allow execute permission values. To accommodate this, execute permission requires specific (*h_digit0*) NFS Allow ACL entry mask (permission) values. Otherwise, the NFS Allow ACL entry can not be supported. The function encountered an NFS Allow ACL entry in which the required mask (permission) bits were not all specified. The ACLs cannot be converted and the request is terminated.

If this message occurs on the z/OS NFS Server, it means that a remote NFS Client is attempting to set ACL entries with an unsupported combination of NFS Allow ACL execute mask (permission) bits. This is not supported by z/OS NFS.

If this message occurs on the z/OS NFS Client, it means that the z/OS NFS Client is attempting to manage ACLs for a remote NFS Server and the remote NFS ACL contains entries with an unsupported combination of NFS Allow ACL execute mask (permission) bits. This is not supported by z/OS NFS.

In the message text: (primarily intended for IBM support personnel use)

h_digit0 The hexadecimal value of the required NFS Allow ACL entry execute mask field bits.

d_digit The 0-origin index of the NFS ACL entry containing the error.

h_digit1 The hexadecimal value of the NFS ACL who type. (1-owner, 2-user, 3-owner group, 4-user group, 8-everyone)

h_digit2 The hexadecimal value of the user, or group, id.

h_digit3 The hexadecimal value of the NFS ACL type field (see RFC3530).

h_digit4 The hexadecimal value of the NFS ACL flag field (see RFC3530).

h_digit5 The hexadecimal value of the NFS ACL mask field (see RFC3530).

System action: The request fails and normal processing continues.

Programmer response: Check the remote system NFS ACL entries and update them to comply with the z/OS NFS ACL management restrictions. See “Displaying and modifying remote file system access control lists” on page 55 for more information.

Operator response: None.

System programmer response: None.

GFSN5019E z/OS UNIX ACL Max size exceeded. Max: *d_digit1* **NFS ACE:** *d_digit2***.**

Explanation: z/OS UNIX does not support ACLs with more than d_digit1 entries. The NFS to z/OS UNIX ACL conversion function received an ACL conversion request which will exceed this limit even though NFS Mask, Allow and Deny ACL entries for a given userid are combined into a single z/OS UNIX ACL entry. The ACLs cannot be converted and the request is terminated.

In the message text:

d_digit The maximum number of z/OS UNIX ACL entries supported for an object.

d_digit2 The 0-origin index of the NFS ACL entry being processed that will cause this limit to be exceeded.

System action: The request fails and normal processing continues.

Programmer response: Reduce the size of the NFS ACL list for the object.

Operator response: None.

System programmer response: None.

GFSN5020E Network File System to z/OS UNIX ACL conversion error. USS ACL Type: *h_digit1* **Mode:** *h_digit2* **ObjType:** *d_digit1* **USS Cnt:** *d_digit2* **RsnCode:** *h_digit3***.**

Explanation: The NFS to z/OS UNIX ACL conversion function experienced an error. See the previous z/OS NFS error messages for details.

In the message text:

h_digit1 Identifies the z/OS UNIX ACL type (1-Access, 2-File Default, 3-Dir Default)

h_digit2 The hexadecimal value of the z/OS UNIX permission mode bits.

d_digit1 Identifies the object type (1-file, 2-directory).

d_digit2 The number of z/OS UNIX ACL entries created before the error was encountered.

h_digit3 The error reason code. This is intended for IBM support personnel use only.

System action: The request fails and normal processing continues.

Programmer response: See the previous z/OS NFS ACL conversion error message to identify the actual error which occurred.

Operator response: None.

System programmer response: None.

GFSN5025E Temporary buffer overflow.

Explanation: NFS detected an error building the load module statistics during startup.

System action: NFS will shut down.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Check that the latest NFS maintenance install completed successfully. Keep the existing z/OS NFS traces and contact IBM Support.

GFSN5026E (*procname***) Module** *module* **is inconsistent between Network File System load modules** *loadmod1***(***fmid1***,** *apar1***,** *compdate1***,** *comptime1***) and** *loadmod2***(***fmid2***,** *apar2***,** *compdate2***,** *comptime2***).**

Explanation: During the Network File System initialization a module was found to be at different maintenance levels in two load modules.

In the message text:

module The module that is at inconsistent maintenance levels.

loadmod1
The name of the first load module containing the module in error.

fmid1 The release level of the module in load module *loadmod1*.

apar1 The APAR maintenance version of the module in load module *loadmod1*.

compdate1
The compile date of the module in load module *loadmod1*.

comptime1
The compile time of the module in load module *loadmod1*.

loadmod2
The name of the second load module containing the module in error.

fmid2 The release level of the module in load module *loadmod2*.

apar2 The APAR maintenance version of the module in load module *loadmod2*.

compdate2
The compile date of the module in load module *loadmod2*.

comptime2
The compile time of the module in load module *loadmod2*.

System action: The Network File System terminates.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Verify that the last maintenance was installed correctly by SMP/E. When a maintenance update is installed for a module SMP/E should install it in all load modules.

GFSN5027E (*procname***) Module** *module* **release** *rel1* **in** *loadmod* **is incompatible with Network File System release** *rel2***.**

Explanation: During the Network File System initialization a module was found in a task which is at an incompatible release level.

In the message text:

module The module that is in error.

rel1 The release level of the module in error.

loadmod The task load module in which the module in error was found.

rel2 The Network File System release level.

System action: The Network File System terminates.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Verify that the last maintenance was installed correctly by SMP/E. When a maintenance update is installed for a module SMP/E should install it in all load modules.

GFSN5028E Last MODULE INFO not found for *text1***.**

Explanation: NFS detected an error building the load module statistics during startup. The maintenance level statistics were not found for module *text1*.

System action: NFS processing continues.

Programmer response: None.

Operator response: Contact the system programmer.

System programmer response: Check that the latest NFS maintenance install completed successfully. Keep the existing z/OS NFS traces and contact IBM Support.

GFSN5030E REQUESTED MEMORY NOT AVAILABLE.

Explanation: An operation to allocate virtual memory was tried but was unsuccessful. If this condition persists, the cause might be either of these problems.

- The value specified in the **region** parameter is too small.
- The value specified in the **bufhigh** attribute is too large.

System action: The request is stopped. The NFS server processing continues.

GFSN5035E UNKNOWN OPTION=*option*

Explanation: The operator entered an invalid CTRACE option that was not recognized by NFS.

In the message text:

option A CTRACE option that was not recognized by NFS.

System action: The invalid option is ignored.

Programmer response: None.

Operator response: Repeat the TRACE CT command with the invalid option corrected.

System programmer response: None.

Messages from the client platform (AIX)

This section provides a list of messages from the client operating system, in response to NFS reply results (messages without message numbers). These messages are platform dependent.

Note: The message text is from an AIX system.

Cross device link

Explanation:

1. An attempt has been made to rename a member of a partitioned data set (PDS) or partitioned data set extended (PDSE), but the target file is not a member of the same PDS or PDSE.
2. An attempt has been made to rename a non-PDS or non-PDSE file, but the target file is a member of a PDS or PDSE.

User response: Try a copy and remove operation, instead of a rename operation.

Directory Not Empty

Explanation: An attempt was made to remove a PDS or PDSE with members.

User response: Delete all members before trying to remove a PDS or PDSE.

File exists

Explanation: An attempt was made to rename a PDS or PDSE, but the target file already exists.

User response: Delete the target file before renaming the PDS or PDSE. This is not required for a regular file or for a PDS or PDSE member.

File Name Too Long

Explanation: The file name is not a valid z/OS data set or member name.

User response: File names must follow the z/OS naming conventions. See *z/OS DFSMS Using Data Sets* for data set naming conventions.

Invalid

Explanation:

1. The specified parameters were incorrect.
2. The creation of a virtual storage access method (VSAM) data set failed.

User response: Respecify the correct parameters, or contact the system programmer to determine the VSAM data set failure on the NFS server.

I/O Error (with possible system programmer response)

Explanation:

1. An unexpected error from catalog management occurred.
2. Dynamic allocation failed during an action other than a read or write.
3. A file could not be opened during an action other than a read or write.
4. An error occurred during the reading of a PDS or PDSE directory.
5. No space is available in the task input/output table (TIOT).
6. The TIOT resource is unavailable.
7. The system is unable to release enough resources.
8. Insufficient units are available.
9. The server could not get enough memory to perform this function.

System programmer response: For explanation 9, stop the server and change the region field in the job control language (JCL) before restarting, or modify parameters in the attributes file. For the other explanations, perform the appropriate action.

I/O Error (with possible user response)

Explanation:

1. An attempt was made to nest PDSs or PDSEs.
2. The maximum number of file allocations was exceeded.
3. The file is being written in text mode, the new write request offset is determined to fall within the end-of-line (EOL) sequence (**lf**, **cr**, **crlf**, **lfcr**) of a previous line, and the new data does not contain the correct EOL characters.
4. The file is being written in text mode, with a nonzero EOL (**lf**, **cr**, **lfcr**, **crlf**). The number of bytes of data in the written line is larger than the maximum record size of the file.
5. The file is being written in text mode, with fixed records, a nonzero EOL, and **blankstrip** not set (no padding blanks on write), and the number of bytes of data in the written line is less than or greater than the record size of the file.
6. The file is being written in text mode, with fixed records and **blankstrip** set, and the line of written data contains trailing blanks as part of the data.
7. When a file containing a 0 length line is written to z/OS as **recfm(u)** in text mode, a write error occurs.

8. An access method services alias name was specified in a remove (**rm** or **rmdir**) or rename (**mv**) request.
9. An “s” was specified in the **recfm** data set creation attribute for a PDS or PDSE.
10. If you try to append data to a member of a PDS or PDSE, an I/O error occurs.
11. An incorrect attribute was specified in the command.

User response: Perform the appropriate action.

Is a directory

Explanation: A nondirectory operation has been tried on a PDS or PDSE.

User response: Use directory operations on the data set.

NFS server *name* not responding still trying

Explanation: Long delays occurred between operations.

User response: Possible user responses follow:

- The server might need extra time to service client requests. Wait until the message "`NFS server` *`name`* `ok`" appears.
- Determine if the network traffic is heavy and overloaded. Try to isolate the path where the client workstation communicates with the server machine.
- Determine if the client workstation transmits the same requests over and over. Commands such as **nfsstat -c** on AIX or UNIX platforms show the number of client retransmissions (**retrans**) as well as the number of **badcalls** and **badxid**. When the values of **badcalls** and **badxid** are high, the client machine usually has bad retransmissions. High retransmissions might be caused by an overloaded network or a slow server.
- If the network is overloaded, contact the network administrator.
- If the server is slow, determine if the client workstation tends to transmit requests out of sequence or incompletely, and you are performing I/O to a file in text mode. If the request is incomplete, the client might not send the remainder of the request until a later time. In this case, increase the value of the server's **cachewindow** attribute. The cachewindow buffers store out of sequence and incomplete requests from clients. The general rule in setting the *n* value of cachewindow(*n*) is

  ```
  n  = (( num of BIOD  + 1 ) *
     (client_max_IO_buffer_size/transfer_size))
  ```

  In the formula:

  num of BIOD
  Is the number of blocked I/O daemons set by the client workstation. This value is usually set to defaults at the installation of the operating system or by the system administrator.

  client_max_IO_buffer_size
  Is the amount of I/O data that is requested by the client (for example, the client writes 8192 bytes of data to the remote file system). This value is determined by the application programs.

  transfer_size
  Is the actual size of data being sent across the network (for example, the 8192 bytes of data can be broken down into 16 smaller packets of 512 bytes (16x512=8192)). This value is determined dynamically by the client workstation.
- If your client workstation tends to send duplicate transmissions for the same request too often, thus increasing the workload on the server, you might want to decrease the client's retransmission rate and increase the request timeout. In the **mount** command, you can specify these values:

  ```
  mount -o retrans=3,timeout=30 IO_buffer_size
  ```

  In the command:

  retrans
  Is the number of retransmissions allowed before a timeout. The default value is vendor specific, ranging from 3 to 5.

  timeout
  Is the timeout value in tenths of a second. timeout=30 means 3 seconds. The default value is vendor specific, ranging from 5 to 11.

  If the reply is not received by the client within the timeout period, a minor timeout occurred for this request. The timeout period is doubled, and the request is resent. The process is repeated until the retransmission count specified by **retrans** is reached; if no reply is received, a major timeout occurred.

No such device

Explanation:

Step 1. The file resides on an off-line device

Step 2. The file was migrated to another storage level. Whether the file is being recalled depends on the **retrieve** attribute

User response: For Step 1, contact the system operator or system programmer.

For Step 2, try the request again later, if the **retrieve** attribute is enabled. If not, try the request again, with the **retrieve** attribute enabled.

No space left on device

Explanation:

1. The file exceeded the space that was allocated for it.
2. The PDS or PDSE exceeded the space that was allocated for it.
3. The PDS directory exceeded the space that was allocated for it.

User response:

1. For explanation 1, save the file into a larger file, and then rename it to the old name, if necessary.
2. For explanation 2, create a larger PDS or PDSE and store the member there. Then copy the members of the old PDS or PDSE to the new PDS or PDSE. Rename the new PDS or PDSE, if necessary.
3. For explanation 3, create a new PDS with a larger directory (use the **dir** attribute). Store the member in the new PDS. Then copy the members from the old PDS to the new PDS and rename them, if necessary.

No such file or directory

Explanation: A **locate** command failed for this file. The file is not cataloged, or the system operator might have unmounted the file before you issued the **umount** command.

User response: Check the spelling. If the spelling is correct, contact the system administrator.

Not a directory

Explanation: A directory operation was made on a file that is not a PDS or PDSE.

User response: Use nondirectory operations on the file.

Not Owner

Explanation:

1. File has not timed out yet.
2. File is open. Another client (maybe even the same client) has the file open for writing.
3. The client tried to change the mode in a **nfsattr** NFS procedure call.

User response: Possible user responses follow:

- For explanation 1, follow the steps in waiting and retrying that are described under the "`Permission denied`" message,
- For explanation 3, do not try to change the mode.

Permission denied

Explanation:

1. The file is not in use but has not timed out yet (occurs most often when writing to PDS or PDSE members without **writetimeout** expiring between saving members).
2. The file is in use by a z/OS user or another client.
3. Dynamic allocation–authorized function requested by an unauthorized user. The error codes from dynamic allocation are printed to the log data set.
4. The Resource Access Control Facility (RACF) is not active.
5. Dynamic allocation–The request was denied by the operator (dsname allocation). The error codes from dynamic allocation are printed to the log data set.
6. Dynamic allocation–The installation validation routine denied this request. The error codes from dynamic allocation are printed to the log data set.
7. You are not authorized for this request.
8. The system operator suspended mount request processing (only if this message appears following a mount attempt).
9. The file/prefix is not exported to the client (only if this message appears following a mount attempt).
10. IDCAMS failed during a rename or remove procedure. Usually this happens because the file is in use. The output from IDCAMS is printed to the log data set.
11. With OS/2, you might get a "SYS0055 Access denied" message if the **noretrieve** attribute is set and a **dir** command is done against a mounted file system containing migrated files.

User response: For explanation 1, to determine if this is the problem, check the timeout values (**attrtimeout**, **readtimeout**, **writetimeout**). Retry the request after the shortest timeout expires. If the request still fails, retry the request after the next shortest timeout expires. If the request still fails, retry after the longest timeout expires. If the request still fails, this is not the problem.

For explanations 2 and 10, retry the request later.

For explanations 3, 4, 5, 6, and 9, notify the system programmer.

For explanation 7, enter the **mvslogin** command again and retry the request. If the request still fails, notify your system programmer.

For explanation 8, notify your system operator or system programmer.

For explanation 11, specify the **retrieve** attribute in the **mount** command, or the system administrator can make that the default.

Read Only File System

Explanation: One of these NFS procedures was tried on a read-only file system: **link**, **write**, **rename**, **remove**, **mkdir**, or **create**.

User response: See the documentation on the exports data set to see how a file system is designated read only (see *z/OS Network File System Guide and Reference*). The exports data set needs to be changed, or you are using it incorrectly.

Stale NFS File Handle

Explanation: A file handle is used by the client and server sides of the NFS to specify a particular file or prefix. A stale file handle occurs when the name is no longer valid, possibly due to one of the these conditions:

1. The file or prefix was removed by the system operator.
2. The server was stopped and restarted. This affects files and members below mount points.

User response: For case 1, unmount and remount the file system or prefix.

For case 2, if SAF, or if SAFEXP security is being used, reenter the **mvslogin** command and retry the request as follows:

1. Change the directory back to the mount point, using the **cd** command
2. Verify the NFS mount point is still accessible using the **df** command
3. Re-navigate to the desired object (MVS data set or member or z/OS UNIX objects)
4. Re-isssue the command with the desired object.

Weak Authorization

Explanation: The authorization data in the remote procedure call (RPC) message was not valid. This is a client side error.

User response: UNIX style authorization is required.

value too large

Explanation: The NFS version 4 protocol defines 64-bit file ids which may be returned to the client application as a 64-bit value. Client applications may receive err79 "Value too large" when using 32-bit system calls such as UNIX stat() against remote NFS version 4 server objects.

User response: Change your application to use 64-bit system calls such as UNIX stat64() to prevent err79 messages in client applications when using NFS version 4.

Chapter 20. Return codes

Table 66 lists the externalized return codes that are defined by the NFS version 2 protocol.

Table 66. Externalized return codes defined by the NFS version 2 protocol

Return Value	Return Code	Description
NFS_OK	0	Requests completed successfully and the results are valid.
NFSERR_PERM	1	Not owner. The operation was not allowed because: 1. The caller does not have correct ownership (not a privileged user as root). 2. The caller is not the owner of the target of the operation. 3. The server requires the use of a privileged port (an available reserved port of 1023 or lower).
NFSERR_NOENT	2	No such file or directory. The file or directory specified does not exist.
NFSERR_IO	5	A hard error occurred when the operation was in progress. For example, this could be a disk error.
NFSERR_NXIO	6	No such device or address.
NFSERR_ACCESS	13	Permission denied. The caller does not have the correct permission to perform the requested operation.
NFSERR_EXIST	17	File exists. The file specified already exists.
NFSERR_NODEV	19	No such device.
NFSERR_NOTDIR	20	Not a directory. The caller specified a non-directory in a directory operation.
NFSERR_ISDIR	21	Is a directory. The caller specified a directory in a non-directory operation.
NFSERR_EINVAL	22	An argument was passed to the z/OS NFS server that was not valid.
NFSERR_FBIG	27	File too large. The operation caused a file to grow beyond the server's limit.
NFSERR_NOSPC	28	No space left on device. The operation caused the server's file system to reach its limit.
NFSERR_ROFS	30	Read-only file system. Write tried on a read-only file system.
NFSERR_NAMETOOLONG	63	File name too long. The file name in an operation was too long.
NFSERR_NOTEMPTY	66	Directory not empty. Tried to remove a directory that was not empty.
NFSERR_DQUOT	69	Disk quota exceeded. The client's disk quota on the server has been exceeded.
NFSERR_STALE	70	The file handle given in the arguments was not valid. That is, the file referred to by that file handle no longer exists, or access to it has been revoked.

Table 67 lists the externalized return codes that are defined by the NFS version 3 protocol.

Table 67. Externalized return codes defined by the NFS version 3 protocol

Return Value	Return Code	Description
NFS_OK	0	Requests completed successfully and the results are valid.
NFS3ERR_PERM	1	Not owner. The caller does not have correct ownership to perform the requested operation.

© Copyright IBM Corp. 1991, 2010

Table 67. Externalized return codes defined by the NFS version 3 protocol (continued)

Return Value	Return Code	Description
NFS3ERR_NOENT	2	No such file or directory. The file or directory specified does not exist.
NFS3ERR_IO	5	A hard error occurred when the operation was in progress. For example, this could be a disk error.
NFS3ERR_NXIO	6	No such device or address.
NFS3ERR_ACCESS	13	Permission denied. The caller does not have the correct permission to perform the requested operation.
NFS3ERR_EXIST	17	File exists. The file specified already exists.
NFS3ERR_XDEV	18	Attempt to do an operation across the file system.
NFS3ERR_NODEV	19	No such device.
NFS3ERR_NOTDIR	20	Not a directory. The caller specified a non-directory in a directory operation.
NFS3ERR_ISDIR	21	Is a directory. The caller specified a directory in a non-directory operation.
NFS3ERR_INVAL	22	An argument was passed to the z/OS NFS server that was not valid.
NFS3ERR_FBIG	27	File too large. The operation caused a file to grow beyond the server's limit.
NFS3ERR_NOSPC	28	No space left on device. The operation caused the server's file system to reach its limit.
NFS3ERR_ROFS	30	Read-only file system. Write tried on a read-only file system.
NFS3ERR_MLINK	31	Too many links.
NFS3ERR_NAMETOOLONG	63	File name too long. The file name in an operation was too long.
NFS3ERR_NOTEMPTY	66	Directory not empty. Tried to remove a directory that was not empty.
NFS3ERR_DQUOT	69	Disk quota exceeded. The client's disk quota on the server has been exceeded.
NFS3ERR_STALE	70	The file handle given in the arguments was not valid. That is, the file referred to by that file handle no longer exists, or access to it has been revoked.
NFS3ERR_BADHANDLE	10001	File handle is not valid.
NFS3ERR_NOT_SYNC	10002	Synchronization mismatch on SETATTR.
NFS3ERR_BAD_COOKIE	10003	READDIR and READDIRPLUS cookie is stale.
NFS3ERR_NOTSUPP	10004	Operation is not supported.
NFS3ERR_TOOSMALL	10005	Buffer or request is too small.
NFS3ERR_SERVERFAULT	10006	Server abandons the request.
NFS3ERR_BADTYPE	10007	Type of an object is not supported.
NFS3ERR_JUKEBOX	10008	Request was initiated, but not completed.

Table 68 lists the externalized return codes that are defined by the NFS version 4 protocol.

Table 68. Externalized return codes defined by the NFS version 4 protocol

Return Value	Return Code	Description
NFS4_OK	0	Requests completed successfully and the results are valid.
NFS4ERR_PERM	1	Not owner. The caller does not have correct ownership to perform the requested operation.
NFS4ERR_NOENT	2	No such file or directory. The file or directory specified does not exist.

Table 68. Externalized return codes defined by the NFS version 4 protocol (continued)

Return Value	Return Code	Description
NFS4ERR_IO	5	A hard error occurred when the operation was in progress. For example, this could be a disk error.
NFS4ERR_NXIO	6	No such device or address.
NFS4ERR_ACCESS	13	Permission denied. The caller does not have the correct permission to perform the requested operation.
NFS4ERR_EXIST	17	File exists. The file specified already exists.
NFS4ERR_XDEV	18	Attempt to do an operation across the file system.
	19	Not used/reserved.
NFS4ERR_NOTDIR	20	Not a directory. The caller specified a non-directory in a directory operation.
NFS4ERR_ISDIR	21	Is a directory. The caller specified a directory in a non-directory operation.
NFS4ERR_INVAL	22	An argument was passed to the z/OS NFS server that was not valid.
NFS4ERR_FBIG	27	File too large. The operation caused a file to grow beyond the server's limit.
NFS4ERR_NOSPC	28	No space left on device. The operation caused the server's file system to reach its limit.
NFS4ERR_ROFS	30	Read-only file system. Write tried on a read-only file system.
NFS4ERR_MLINK	31	Too many hard links.
NFS4ERR_NAMETOOLONG	63	File name too long. The file name in an operation was too long.
NFS4ERR_NOTEMPTY	66	Directory not empty. Tried to remove a directory that was not empty.
NFS4ERR_DQUOT	69	Disk quota exceeded. The client's disk quota on the server has been exceeded.
NFS4ERR_STALE	70	The file handle given in the arguments was not valid. That is, the file referred to by that file handle no longer exists, or access to it has been revoked.
NFS4ERR_BADHANDLE	10001	File handle is not valid.
NFS4ERR_BAD_COOKIE	10003	READDIR and READDIRPLUS cookie is stale.
NFS4ERR_NOTSUPP	10004	Operation is not supported.
NFS4ERR_TOOSMALL	10005	Buffer or request is too small.
NFS4ERR_SERVERFAULT	10006	Server abandons the request.
NFS4ERR_BADTYPE	10007	Type of an object is not supported.
NFS4ERR_DELAY	10008	Request was initiated, but not completed. File was busy; retry.
NFS4ERR_SAME	10009	Request was initiated, but attributes are the same.
NFS4ERR_DENIED	10010	Lock was unavailable.
NFS4ERR_EXPIRED	10011	Lock release expired.
NFS4ERR_LOCKED	10012	I/O failure due to lock.
NFS4ERR_GRACE	10013	In grace period.
NFS4ERR_FHEXPIRED	10014	File handle expired.
NFS4ERR_SHARE_DENIED	10015	Share reserve denied.
NFS4ERR_WRONGSEC	10016	Wrong security level.
NFS4ERR_CLID_INUSE	10017	Client id in use.
NFS4ERR_RESOURCE	10018	Resource exhaustion.
NFS4ERR_MOVED	10019	Filesystem relocated.
NFS4ERR_NOFILEHANDLE	10020	Current file handle is not set.
NFS4ERR_MINOR_VERS_MISMATCH	10021	Minor version not supported.

Table 68. Externalized return codes defined by the NFS version 4 protocol (continued)

Return Value	Return Code	Description
NFS4ERR_STALE_CLIENTID	10022	Server has rebooted.
NFS4ERR_STALE_STATEID	10023	Server has rebooted.
NFS4ERR_OLD_STATEID	10024	State is not in synch.
NFS4ERR_BAD_STATEID	10025	Incorrect state id.
NFS4ERR_BAD_SEQID	10026	Request is out of sequence.
NFS4ERR_NOT_SAME	10027	Verify - attributes not the same.
NFS4ERR_LOCK_RANGE	10028	Lock range not supported.
NFS4ERR_SYMLINK	10029	Should be file/directory.
NFS4ERR_RESTOREFH	10030	No saved file handle.
NFS4ERR_LEASE_MOVED	10031	Some file system moved.
NFS4ERR_ATTRNOTSUPP	10032	Recommended attribute not supported.
NFS4ERR_NO_GRACE	10033	Reclaim attempt was not within the grace period.
NFS4ERR_RECLAIM_BAD	10034	Reclaim error occurred at server.
NFS4ERR_RECLAIM_CONFLICT	10035	Conflict occurred on reclaim.
NFS4ERR_BADXDR	10036	XDR decode failed.
NFS4ERR_LOCKS_HELD	10037	File locks held at CLOSE.
NFS4ERR_OPENMODE	10038	Conflict in OPEN and I/O.
NFS4ERR_BAD_OWNER	10039	Owner translation was not correct.
NFS4ERR_BADCHAR	10040	utf-8 character was not supported.
NFS4ERR_BADNAME	10041	Name is not supported.
NFS4ERR_BAD_RANGE	10042	Lock range was not supported.
NFS4ERR_LOCK_NOTSUPP	10043	No atomic upgrade or downgrade.
NFS4ERR_OP_ILLEGAL	10044	Operation was undefined.
NFS4ERR_DEADLOCK	10045	Deadlock occurred in file locking.
NFS4ERR_FILE_OPEN	10046	An open file blocked the operation.
NFS4ERR_ADMIN_REVOKED	10047	The lock owner's state was revoked.
NFS4ERR_CB_PATH_DOWN	10048	The callback path was down.

Table 69 lists the z/OS UNIX return codes and their equivalent NFS Version 2 server return codes.

Table 69. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 2 return codes

OMVS Codes	Dec	Hex	NFS V2 Codes (NFSERR_)	Dec	Hex	Description
EACCES	111	006F	ACCES	13	000D	Permission denied
EAGAIN	112	0070	IO	5	0005	Resource is temporarily unavailable
EBUSY	114	0072	IO	5	0005	Resource is busy
EDEADLK	116	0074	IO	5	0005	A resource deadlock is avoided
EEXIST	117	0075	EXIST	17	0011	The file exists
EFAULT	118	0076	IO	5	0005	The address is incorrect

Table 69. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 2 return codes (continued)

OMVS Codes	Dec	Hex	NFS V2 Codes (NFSERR_)	Dec	Hex	Description
EFBIG	119	0077	FBIG	27	001B	The file is too large
EINVAL	121	0079	IO	5	0005	The parameter is incorrect
EIO	122	007A	IO	5	0005	An I/O error occurred
EISDIR	123	007B	ISDIR	21	0015	The file specified is a directory
EMFILE	124	007C	IO	5	0005	Too many files are open for this directory
EMLINK	125	007D	IO	5	0005	Too many links occurred
ENAMETOOLONG	126	007E	NAMETOOLONG	63	003F	The file name is too long
ENFILE	127	007F	IO	5	0005	Too many files are open
ENODEV	128	0080	NODEV	19	0013	No such device exists
ENOENT	129	0081	NOENT	2	0002	No such file, directory or IPC member exists
ENOMEM	132	0084	IO	5	0005	Not enough space is available
ENOSPC	133	0085	NOSPC	28	001C	No space is left on device
ENOTDIR	135	0087	NOTDIR	20	0014	Not a directory
ENOTEMPTY	136	0088	NOTEMPTY	66	0042	Directory is not empty
ENXIO	138	008A	NXIO	6	0006	No such device or address exists
EPERM	139	008B	IO	5	0005	The operation is not permitted
EROFS	141	008D	ROFS	30	001E	The specified file system is read only
EXDEV	144	0090	XDEV	18	0012	A link to a file on another file system was attempted
E2BIG	145	0091	IO	5	0005	The parameter list is too long
ELOOP	146	0092	IO	5	0005	A loop is encountered in symbolic links
EILSEQ	147	0093	IO	5	0005	The byte sequence is illegal

Table 69. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 2 return codes (continued)

OMVS Codes	Dec	Hex	NFS V2 Codes (NFSERR_)	Dec	Hex	Description
EMVSERR	157	009D	IO	5	0005	MVS environmental or internal error
EMVSPARM	158	009E	IO	5	0005	Bad parameters were passed to the service
EMVSPFSFILE	159	009F	IO	5	0005	z/OS UNIX encountered a permanent file error
EMVSPFSPERM	162	00A2	IO	5	0005	z/OS UNIX encountered a system error
EMVSSAFEXTRERR	163	00A3	IO	5	0005	SAF/RACF extract error
EMVSSAF2ERR	164	00A4	IO	5	0005	SAF/RACF error
EDQUOT	1133	046D	DQUOT	69	0045	Disk quota exceeded
ESTALE	1134	046E	STALE	70	0046	Stale NFS file handle
EREMOTE	1135	046F	IO	5	0005	Too many levels of remote in path

Table 70 lists the z/OS UNIX return codes and their equivalent NFS Version 3 server return codes.

Table 70. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 3 return codes

OMVS Codes	Dec	Hex	NFS V3 Codes (NFS3ERR_)	Dec	Hex	Description
EACCES	111	006F	ACCES	13	000D	Permission denied
EAGAIN	112	0070	JUKEBOX	10008	2718	Resource is temporarily unavailable
EBUSY	114	0072	JUKEBOX	10008	2718	Resource is busy
EDEADLK	116	0074	DEADLOCK	10045	273D	A resource deadlock is avoided
EEXIST	117	0075	EXIST	17	0011	The file exists
EFAULT	118	0076	SERVERFAULT	10006	2716	The address is incorrect
EFBIG	119	0077	FBIG	27	001B	The file is too large
EINVAL	121	0079	INVAL	22	0016	The parameter is incorrect
EIO	122	007A	IO	5	0005	An I/O error occurred

Table 70. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 3 return codes (continued)

OMVS Codes	Dec	Hex	NFS V3 Codes (NFS3ERR_)	Dec	Hex	Description
EISDIR	123	007B	ISDIR	21	0015	The file specified is a directory
EMFILE	124	007C	JUKEBOX	10008	2718	Too many files are open for this directory
EMLINK	125	007D	MLINK	31	001F	Too many links occurred
ENAMETOOLONG	126	007E	NAMETOOLONG	63	003F	The file name is too long
ENFILE	127	007F	DQUOT	69	0045	Too many files are open
ENODEV	128	0080	NODEV	19	0013	No such device exists
ENOENT	129	0081	NOENT	2	0002	No such file, directory or IPC member exists
ENOMEM	132	0084	JUKEBOX	10008	2718	Not enough space is available
ENOSPC	133	0085	NOSPC	28	001C	No space is left on device
ENOTDIR	135	0087	NOTDIR	20	0014	Not a directory
ENOTEMPTY	136	0088	NOTEMPTY	66	0042	Directory is not empty
ENXIO	138	008A	NXIO	6	0006	No such device or address exists
EPERM	139	008B	PERM	1	0001	The operation is not permitted
EROFS	141	008D	ROFS	30	001E	The specified file system is read only
EXDEV	144	0090	XDEV	18	0012	A link to a file on another file system was attempted
E2BIG	145	0091	SERVERFAULT	10006	2716	The parameter list is too long
ELOOP	146	0092	IO	5	0005	A loop is encountered in symbolic links
EMVSERR	157	009D	SERVERFAULT	10006	2716	MVS environmental or internal error
EMVSPARM	158	009E	SERVERFAULT	10006	2716	Bad parameters were passed to the service

Table 70. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 3 return codes (continued)

OMVS Codes	Dec	Hex	NFS V3 Codes (NFS3ERR_)	Dec	Hex	Description
EMVSPFSFILE	159	009F	IO	5	0005	z/OS UNIX encountered a permanent file error
EMVSPFSPERM	162	00A2	IO	5	0005	z/OS UNIX encountered a system error
EMVSSAFEXTRERR	163	00A3	IO	5	0005	SAF/RACF extract error
EMVSSAF2ERR	164	00A4	IO	5	0005	SAF/RACF error
EDQUOT	1133	046D	DQUOT	69	0045	Disk quota exceeded
ESTALE	1134	046E	STALE	70	0046	Stale NFS file handle
EREMOTE	1135	046F	REMOTE	71	0077	Too many levels of remote in path

Table 71 lists the z/OS UNIX return codes and their equivalent NFS Version 4 server return codes.

Table 71. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 4 return codes

OMVS Codes	Dec	Hex	NFS V4 Codes (NFS4ERR_)	Dec	Hex	Description
EACCES	111	006F	ACCES	13	000D	Permission denied
EAGAIN	112	0070	DELAY	10008	2718	Resource is temporarily unavailable
EBUSY	114	0072	DELAY	10008	2718	Resource is busy
EDEADLK	116	0074	DEADLOCK	10045	273D	A resource deadlock is avoided
EEXIST	117	0075	EXIST	17	0011	The file exists
EFAULT	118	0076	SERVERFAULT	10006	2716	The address is incorrect
EFBIG	119	0077	FBIG	27	001B	The file is too large
EINVAL	121	0079	INVAL	22	0016	The parameter is incorrect
EIO	122	007A	IO	5	0005	An I/O error occurred
EISDIR	123	007B	ISDIR	21	0015	The file specified is a directory
EMFILE	124	007C	DQUOT	69	0045	Too many files are open for this directory

Table 71. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 4 return codes (continued)

OMVS Codes	Dec	Hex	NFS V4 Codes (NFS4ERR_)	Dec	Hex	Description
EMLINK	125	007D	MLINK	31	001F	Too many links occurred
ENAMETOOLONG	126	007E	NAMETOOLONG	63	003F	The file name is too long
ENFILE	127	007F	DQUOT	69	0045	Too many files are open
ENODEV	128	0080	NXIO	6	0006	No such device exists
ENOENT	129	0081	NOENT	2	0002	No such file, directory or IPC member exists
ENOMEM	132	0084	DELAY	10008	2718	Not enough space is available
ENOSPC	133	0085	NOSPC	28	001C	No space is left on device
ENOTDIR	135	0087	NOTDIR	20	0014	Not a directory
ENOTEMPTY	136	0088	NOTEMPTY	66	0042	Directory is not empty
ENXIO	138	008A	NXIO	6	0006	No such device or address exists
EPERM	139	008B	PERM	1	0001	The operation is not permitted
EROFS	141	008D	ROFS	30	001E	The specified file system is read only
EXDEV	144	0090	XDEV	18	0012	A link to a file on another file system was attempted
E2BIG	145	0091	SERVERFAULT	10006	2716	The parameter list is too long
ELOOP	146	0092	IO	5	0005	A loop is encountered in symbolic links
EMVSERR	157	009D	SERVERFAULT	10006	2716	MVS environmental or internal error
EMVSPARM	158	009E	SERVERFAULT	10006	2716	Bad parameters were passed to the service
EMVSPFSFILE	159	009F	IO	5	0005	z/OS UNIX encountered a permanent file error
EMVSPFSPERM	162	00A2	IO	5	0005	z/OS UNIX encountered a system error

Table 71. z/OS NFS Server: z/OS UNIX return codes mapped to NFS Version 4 return codes (continued)

OMVS Codes	Dec	Hex	NFS V4 Codes (NFS4ERR_)	Dec	Hex	Description
EMVSSAFEXTRERR	163	00A3	IO	5	0005	SAF/RACF extract error
EMVSSAF2ERR	164	00A4	IO	5	0005	SAF/RACF error
EMVSNORTL	167	00A7	IO	5	0005	Access to z/OS UNIX version of C RTL denied
EDQUOT	1133	046D	DQUOT	69	0045	Disk quota exceeded
ESTALE	1134	046E	STALE	70	0046	Stale NFS file handle

Chapter 21. Reason codes

This topic lists reason codes that are returned by the z/OS Network File System server and the z/OS Network File System client.

Network File System reason codes are generic eight-digit hexadecimal codes that provide an indication of the problem location. They appear in the format **6n***xxyyyy*, where:

6D
Indicates NFS Client and Server Common Shared modules (that is, GFSNxxxx; for example, GFSNETRC).

6E Indicates NFS Client modules (that is, GFSCxxxx).

6F Indicates NFS Server modules (that is, GFSAxxxx).

xx Is a two-digit hexadecimal number with one of the following values:

> **00 - 0F**
> Have special meaning. This range applies only to the z/OS NFS Client reason codes. See "Special reason codes (*xx* is 00-0F)" on page 428.
>
> **10 - 3F**
> Identifies the NFS client module where the reason code was generated.
>
> **40 – FF**
> Identifies the NFS server module where the reason code was generated.

yyyy
Is a four-digit hexadecimal number with one of the following values:

> **0000 - 7FFF**
> NFS reason codes.
>
> > **0000 - 0FFF**
> > Reason codes that match USS JR*xxxx* reason codes. See "Reason codes from NFS Client or NFS Server modules (*xx* is 10-FF)" on page 439.
> >
> > **1000 - 3FFF**
> > Global reason codes that have the same meaning independent of module id. See "Global reason codes (*yyyy* = 1000 - 3FFF)" on page 443.
> >
> > **4000 - 4FFF**
> > Module specific reason codes. A given value has different meanings depending on the module id. See "Module specific reason codes (*yyyy* = 4000 - 4FFF)" on page 447.
> >
> > **5000 - 70FF**
> > Reserved
> >
> > **7100 - 73FF**
> > Reason codes that match TCPIP JR*xxxx* reason codes. See *z/OS UNIX System Services Messages and Codes*.
>
> **8000 - FFFF**
> The line number of the location in the code where the error occurred. This line number is intended for IBM Service use only.

© Copyright IBM Corp. 1991, 2010

Special reason codes (*xx* is 00-0F)

Reason codes whose *xx* part is in the range 00 through 0F have special meanings. Table 72 contains the rest of the reason code information that is presented in client messages, including the return codes *retc*.

Table 72. Special NFS reason codes

xx	**Error type**	*yyyy*	**Description**
01	Parsing error		See Table 73 on page 438 and Table 74 on page 439.
02	TCP/IP common error	0001	**clntudp_create( )** failed
02	TCP/IP common error	0002	Server NFS port is not 2049
02	TCP/IP common error	0003	**authunix_create( )** failed
02	TCP/IP common error	0004	**clnt_control( )** timeout failed
02	TCP/IP common error	0005	**clnt_control( )** total timeout failed
02	TCP/IP common error	0006	**clnttcp_create( )** failed
02	TCP/IP common error	0007	**clntudp_bufcreate( )** failed
02	RPCSEC_GSS error	0008	RPCSEC_GSS request failed. *Probable cause*: This failure can happen because of the following reasons: 1. The Kerberos Dynamic Load Library could not be loaded by the NFS client. 2. The kerberos context could not be created by the NFS client. 3. The GSS credentials for the NFS client's Kerberos Principal could not be obtained. *Action*: Check your kerberos configurations, fix any configuration issues identified, restart the NFS client and retry the operation. Contact the system programmer if the problem persists.
02	TCP/IP common error	*lnum*	**clnt_call( )** timeout (*retc*=0467h, ETIMEDOUT) *Action*: Increase the value of the timeo client attribute on the **mount** command.
02	TCP/IP common error	*lnum*	**clnt_call( )** EINTR (*retc*=0078h, EINTR)
03	TCP/IP error	*yyyy*	TCP error - **clnt_control( )** failed
03	TCP/IP error	*yyyy*	TCP error - **authunix_create( )** failed
03	TCP/IP error	*yyyy*	TCP error - **clnt_call( )** failed
04	TCP/IP error	*yyyy*	Authentication error - **authunix_create( )** failed
05	NFS protocol error	0001	**EPERM** Not owner (operation not permitted) *Probable cause*: An attempt was made to perform an operation limited to processes with appropriate privileges or to the owner of a file or other resource. *Action*: Ensure that uid and gid are correct.
05	NFS protocol error	0002	**ENOENT** No such file or directory *Probable cause*: A component of a specified path name does not exist, or the path name is an empty string. *Action*: Ensure that the file/directory exists.

Table 72. Special NFS reason codes (continued)

xx	Error type	*yyyy*	Description
05	NFS protocol error	0005	**EIO** I/O error *Probable cause*: Some physical input or output error occurred. *Action*: No action required.
05	NFS protocol error	0006	**ENXIO** No such device or address *Probable cause*: Input or output on a special file refers to a device that does not exist, or makes a request beyond the capabilities of the device. It may also occur when, for example, a tape drive is not on-line. *Action*: Ensure that the mount address is correct.
05	NFS protocol error	000D	**EACCES** Permission denied *Probable cause*: An attempt was made to access a file in a way forbidden by its file access permissions. *Action*: Verify that the correct access authority is being requested.
05	NFS protocol error	0011	**EEXIST** File/dir exists *Probable cause*: An existing file/dir was mentioned in an inappropriate context; for example, as a new link name in the link function. *Action*: Try to use another file/dir name.
05	NFS protocol error	0012	**EXDEV** Cross-device link *Probable cause*: A hard link to a file on another file system was attempted. *Action*: Ensure that the link is correct.
05	NFS protocol error	0013	**ENODEV** No such device *Probable cause*: An attempt was made to apply an inappropriate function to a device; for example, trying to read a write-only device such as printer. *Action*: Ensure that the command parameters are correct.
05	NFS protocol error	0014	**ENOTDIR** Not a directory *Probable cause*: A component of the specified path name exists, but it is not a directory, when a directory was expected. *Action*: Ensure that the specified path name is correct.
05	NFS protocol error	0015	**EISDIR** Is a directory *Probable cause*: An attempt was made to open a directory with write mode specified. *Action*: Ensure that the access mode is correct.

Table 72. Special NFS reason codes (continued)

xx	Error type	*yyyy*	Description
05	NFS protocol error	0016	**EINVAL** Incorrect arguments *Probable cause*: Some invalid argument was supplied, source had an invalid superblock or, a remount was attempted, while source was not already mounted on target. *Action*: Ensure that the arguments are correct. If remount is used, determine that the mount command was made.
05	NFS protocol error	001B	**EFBIG** File too large *Probable cause*: The size of a file would exceed the maximum file size of implementation or offset maximum established in the corresponding file description. *Action*: Operate with files that satisfy maximum file size requirement and use offsets that satisfy maximum offset requirement.
05	NFS protocol error	001C	**ENOSPC** No space left on device *Probable cause*: During the write function on a regular file or when extending a directory, there is no free space left on the device. *Action*: Use another directory or device for store file or clear hash/garbage (be sure that these data are useless).
05	NFS protocol error	001E	**EROFS** Read-only file system *Probable cause*: An attempt was made to modify a file or directory on a file system that is read only. *Action*: Ensure that you have sufficient privileges.
05	NFS protocol error	001F	**EMLINK** Too many links *Probable cause*: An attempt was made to have the link count of a single file exceed (LINK_MAX). *Action*: –
05	NFS protocol error	003F	**ENAMETOOLONG** File name is too long *Probable cause*: The length of a pathname exceeds (PATH_MAX), or a pathname component is longer than (NAME_MAX) and (_POSIX_NO_TRUNC) was in effect for that file. *Action*: Use names that correspond with filename requirements.
05	NFS protocol error	0042	**ENOTEMPTY** Directory is not empty *Probable cause*: A directory with entries other than dot and dot-dot was supplied when an empty directory was expected. *Action*: Ensure that the directory is empty.

Table 72. Special NFS reason codes (continued)

xx	Error type	*yyyy*	Description
05	NFS protocol error	0045	**EDQUOT** Disk quota exceeded *Probable cause*: The user's quota of disk blocks or user's quota of inodes was exhausted. *Action*: Ensure that free space is available on the disk.
05	NFS protocol error	0046	**ESTALE** File handle is stale *Probable cause*: An attempt was made to access an open file (on an NFS file system) which is now unavailable as referenced by the file descriptor. This may indicate the file was deleted on the NFS server or some other catastrophic event occurred. *Action*: Ensure that file was not deleted on the NFS server.
05	NFS protocol error	0047	**EREMOTE** Too many levels of remote *Probable cause*: Server made attempt to handle an NFS request by generating a request to another NFS server, which is not allowed. *Action*: No action required.
05	NFS protocol error	2711	**EBADF (NFSERR_BADHANDLE)** Bad file descriptor *Probable cause*: A file descriptor argument is out of range, refers to no open file, or a read (write) request is made to a file that is only open for writing (reading). *Action*: Ensure that the descriptor is right.
05	NFS protocol error	2712	**EBADF (NFSERR_NOT_SYNC)** Not sync *Probable cause*: Update synchronization mismatch was detected during a SETATTR operation. *Action*: –
05	NFS protocol error	2713	**EFAULT (NFSERR_BAD_COOKIE)** Bad cookie *Probable cause*: READDIR cookie is stale. *Action*: Retry the read dir operation.
05	NFS protocol error	2714	**ENOSYS (NFSERR_NOTSUPP)** Operation not supported *Probable cause*: An attempt was made to use a function or operation that is not available in this implementation. *Action*: Ensure that the operation is available and the command is right.
05	NFS protocol error	2715	**ENOBUFS (NFSERR_TOOSMALL)** Buffer too small *Probable cause*: Insufficient buffer resources were available in the system to perform the socket operation. *Action*: Increase the size of the socket buffers.

Table 72. Special NFS reason codes (continued)

xx	**Error type**	*yyyy*	**Description**
05	NFS protocol error	2716	**EIO (NFSERR_SERVERFAULT)** Server fault *Probable cause*: An error occurred on the server which does not map to any of the legal NFS version 4 protocol error values. *Action*: –
05	NFS protocol error	2717	**ENXIO (NFSERR_BADTYPE)** Bad type *Probable cause*: The named file is a character-special or block-special file and the associated device does not exist. *Action*: Ensure that the mount address is correct.
05	NFS protocol error	2718	**EAGAIN (NFSERR_JUKEBOX/NFSERR_DELAY** Jukebox *Probable cause*: The server initiated the request, but was not able to complete it in a timely fashion. *Action*: No action required. (This is a temporary condition and later calls to the same routine may complete normally.)
05	NFS protocol error	2719	**EIO (NFSERR_SAME)** nverify says attrs same *Probable cause*: This error is returned by the NVERIFY operation to signify that the attributes compared were the same as provided in the client request. *Action*: No action required.
05	NFS protocol error	271A	**ENOLCK (NFSERR_DENIED)** Lock unavailable *Probable cause*: A system-imposed limit on the number of simultaneous file locks was reached. *Action*: Wait and retry the operation.
05	NFS protocol error	271B	**EIO (NFSERR_EXPIRED)** Lock lease expired *Probable cause*: A lease has expired that is being used in the current operation. *Action*: Retry the operation.
05	NFS protocol error	271C	**EIO (NFSERR_LOCKED)** I/O failed due to lock *Probable cause*: A read/write operation was attempted on a locked file. *Action*: Wait and retry the operation.
05	NFS protocol error	271D	**EIO (NFSERR_GRACE)** In grace period *Probable cause*: This server is in its recovery or grace period which should match the lease period of the server. *Action*: Wait and retry the operation.

Table 72. Special NFS reason codes (continued)

xx	**Error type**	*yyyy*	**Description**
05	NFS protocol error	271E	**EIO (NFSERR_FHEXPIRED)** File handle expired *Probable cause*: The file handle provided is volatile and has expired at the server. *Action*: Try to remount.
05	NFS protocol error	271F	**EIO (NFSERR_SHARE_DENIED)** Share reserve denied *Probable cause*: An attempt to OPEN a file with a share reservation has failed because of a share conflict. *Action*: Solve a share conflict.
05	NFS protocol error	2720	**EACCES (NFSERR_WRONGSEC)** Wrong security flavor *Probable cause*: An open with write mode has been attempted to a write protected tape. *Action*: Ensure that the user has sufficient privileges.
05	NFS protocol error	2721	**EIO (NFSERR_CLID_INUSE)** Clientid in use *Probable cause*: The SETCLIENTID operation has found that a client id is already in use by another client. *Action*: Try to remount.
05	NFS protocol error	2722	**EAGAIN (NFSERR_RESOURCE)** Resource exhaustion *Probable cause*: Requested resource, such as a lock or a process, is temporarily unavailable. *Action*: Retry the operation. (This is a temporary condition and later calls to the same routine may complete normally.)
05	NFS protocol error	2723	**EREMCHG (NFSERR_MOVED)** Filesystem moved *Probable cause*: The file system that contains the current file handle object is not present at the server. It may have been relocated, migrated to another server or may have never been present. The client may obtain the new file system location by obtaining the "fs_locations" attribute for the current file handle. *Action*: No action required.
05	NFS protocol error	2724	**EMVSERR (NFSERR_NOFILEHANDLE)** No file handle *Probable cause*: The logical current file handle value (or, in the case of RESTOREFH, the saved file handle value) has not been set properly. This may be a result of a malformed COMPOUND operation (that is, no PUTFH or PUTROOTFH before an operation that requires the current file handle be set). *Action*: Try to remount.

Table 72. Special NFS reason codes (continued)

xx	Error type	*yyyy*	Description
05	NFS protocol error	2725	**EIO (NFSERR_MINOR_VERS_MISMATCH)** Minor version is not supported *Probable cause*: The server has received a request that specifies an unsupported minor version. The server must return a COMPOUND4res with a zero length operations result array. *Action*: No action required.
05	NFS protocol error	2726	**EIO (NFSERR_STALE_CLIENTID)** Server has rebooted *Probable cause*: A clientid not recognized by the server was used in a locking or SETCLIENTID_CONFIRM request. *Action*: Try to remount.
05	NFS protocol error	2727	**EIO (NFSERR_STALE_STATEID)** Server has rebooted *Probable cause*: A stateid generated by an earlier server instance was used. *Action*: Try to remount.
05	NFS protocol error	2728	**EIO (NFSERR_OLD_STATEID)** State is out of sync *Probable cause*: A stateid that designates the locking state for a lockowner-file at an earlier time was used. *Action*: Try to remount.
05	NFS protocol error	2729	**EIO (NFSERR_BAD_STATEID)** Incorrect stateid *Probable cause*: A stateid generated by the current server instance, but which does not designate any locking state (either current or superseded) for current lockowner-file pair, was used. *Action*: Try to remount.
05	NFS protocol error	272A	**EIO (NFSERR_BAD_SEQID)** Request is out of sequence *Probable cause*: The sequence number in a locking request is neither the next expected number or the last number processed. *Action*: No action required.
05	NFS protocol error	272B	**EIO (NFSERR_NOT_SAME)** verify - attrs not same *Probable cause*: This error is returned by the VERIFY operation to signify that the attributes compared were not the same as provided in the client's request. *Action*: No action required.

Table 72. Special NFS reason codes (continued)

xx	Error type	*yyyy*	Description
05	NFS protocol error	272C	**EIO (NFSERR_LOCK_RANGE)** Lock range not supported *Probable cause*: A lock request is operating on a sub-range of a current lock for the lock owner and the server does not support this type of request. *Action*: No action required.
05	NFS protocol error	272D	**EIO (NFSERR_SYMLINK)** Should be file/directory *Probable cause*: The current file handle provided for a LOOKUP is not a directory but a symbolic link. Also used if the final component of the OPEN path is a symbolic link. *Action*: Ensure that the path name is correct.
05	NFS protocol error	272E	**EMVSERR (NFSERR_RESTOREFH)** No saved file handle *Probable cause*: The RESTOREFH operation does not have a saved file handle (identified by SAVEFH) to operate upon. *Action*: No action required.
05	NFS protocol error	272F	**EREMCHG (NFSERR_LEASE_MOVED)** Lease moved *Probable cause*: A lease being renewed is associated with a file system that has been migrated to a new server. *Action*: No action required.
05	NFS protocol error	2730	**EIO (NFSERR_ATTRNOTSUPP)** Recommended attr not supported *Probable cause*: An attribute specified is not supported by the server. Does not apply to the GETATTR operation. *Action*: No action required.
05	NFS protocol error	2731	**EIO (NFSERR_NO_GRACE)** Reclaim outside of grace *Probable cause*: A reclaim of client state was attempted in circumstances in which the server cannot guarantee that conflicting state has not been provided to another client. This can occur because the reclaim has been done outside of the grace period of the server, after the client has done a RECLAIM_COMPLETE operation, or because previous operations have created a situation in which the server is not able to determine that a reclaim-interfering edge condition does not exist. *Action*: No action required.
05	NFS protocol error	2732	**EIO (NFSERR_RECLAIM_BAD)** Reclaim error at server *Probable cause*: The reclaim provided by the client does not match any of the server’s state consistency checks and is bad. *Action*: No action required.

Table 72. Special NFS reason codes (continued)

xx	Error type	*yyyy*	Description
05	NFS protocol error	2733	**EIO (NFSERR_RECLAIM_CONFLICT)** Conflict on reclaim *Probable cause*: The reclaim provided by the client has encountered a conflict and cannot be provided. Potentially indicates a misbehaving client. *Action*: No action required.
05	NFS protocol error	2734	**EMVSERR (NFSERR_BADXDR)** XDR decode failure *Probable cause*: The server encountered an XDR decoding error while processing an operation. *Action*: No action required.
05	NFS protocol error	2735	**EIO (NFSERR_LOCKS_HELD)** Locks held at close *Probable cause*: A CLOSE was attempted and file locks would exist after the CLOSE. *Action*: No action required.
05	NFS protocol error	2736	**EIO (NFSERR_OPENMODE)** Conflict in OPEN & I/O *Probable cause*: The client attempted a READ, WRITE, LOCK or SETATTR operation not sanctioned by the stateid passed (for example, writing to a file opened only for read). *Action*: Ensure that the user has sufficient privileges.
05	NFS protocol error	2737	**EIO (NFSERR_BADOWNER)** Owner translation bad *Probable cause*: An owner, owner_group, or ACL attribute value can not be translated to local representation. *Action*: No action required.
05	NFS protocol error	2738	**EIO (NFSERR_BADCHAR)** Utf-8 char not supported *Probable cause*: A UTF-8 string contains a character which is not supported by the server in the context in which it being used. *Action*: Use correct character encoding symbols.
05	NFS protocol error	2739	**EIO (NFSERR_BADNAME)** Name not supported *Probable cause*: A name string in a request consists of valid UTF-8 characters supported by the server but the name is not supported by the server as a valid name for current operation. *Action*: Change the name on supported by the server.
05	NFS protocol error	2740	**ERANGE (NFSERR_BAD_RANGE)** Lock range not supported *Probable cause*: The result of the function is too large (overflow) or too small (underflow) to be represented in the available space. *Action*: No action required.

Table 72. Special NFS reason codes (continued)

xx	**Error type**	*yyyy*	**Description**
05	NFS protocol error	2741	**EIO (NFSERR_LOCK_NOTSUPP)** No atomic up/downgrade *Probable cause*: – *Action*: No action required.
05	NFS protocol error	2742	**EIO (NFSERR_OP_ILLEGAL)** Undefined operation *Probable cause*: An illegal operation value has been specified in the argop field of a COMPOUND or CB_COMPOUND procedure. *Action*: Ensure that the operation is correct.
05	NFS protocol error	2743	**EDEADLK (NFSERR_DEADLOCK)** File locking deadlock *Probable cause*: An attempt was made to lock a system resource that would have resulted in a deadlock situation. *Action*: No action required.
05	NFS protocol error	2744	**EIO (NFSERR_FILE_OPEN)** Open file blocks operation *Probable cause*: The operation cannot be successfully processed because a file involved in the operation is currently open. *Action*: Ensure that the appropriate file is not opened. Then retry the operation.
05	NFS protocol error	2745	**EIO (NFSERR_ADMIN_REVOKED)** Lockowner state revoked *Probable cause*: Due to administrator intervention, the lock owner's record locks, share reservations, and delegations have been revoked by the server. *Action*: No action required. Consult with the server's administrator.
05	NFS protocol error	2746	**EIO (NFSERR_CB_PATH_DOWN)** Callback path down *Probable cause*: *Action*: –
0C	System abend without SDWA	0sss	sss - System abend code
0D	User abend without SDWA	0uuu	uuu - User abend code
0E	System abend	0sss	sss - System abend code
0F	NFS abend	0uuu	uuu - User abend code

You can use Table 73 on page 438 for initial translation of the reason code *reasoncd* information presented in client messages related to parsing errors.

Table 73. Parsing error (when reason code is 6E01xxxx)

Last 4 hex digits of reason code	Description
7*xxx*	Unknown keyword
11*yy*	Host name
12*yy*	Path name
13*yy*	Keyword **acdirmax**
14*yy*	Keyword **acdirmin**
15*yy*	Keyword **acregmax**
16*yy*	Keyword **acregmin**
17*yy*	Keyword **cln_ccsid**
18*yy*	Keyword **srv_ccsid**
19*yy*	Keyword **hard**
1A*yy*	Keyword **soft**
1B*yy*	Keyword **retrans**
1C*yy*	Keyword **timeo**
1D*yy*	Keyword **wsize**
1E*yy*	Keyword **rsize**
1F*yy*	Keyword **retry**
21*yy*	Keyword **biod**
22*yy*	Keyword **bufhigh**
23*yy*	Keyword **delaywrite**
24*yy*	Keyword **readahead**
25*yy*	Keyword **attrcaching**
26*yy*	Keyword **datacaching**
27*yy*	Keyword **dynamicsizeadj**
28*yy*	Keyword **delim**
29*yy*	Keyword **xlat**
2A*yy*	Keyword **vers**
2B*yy*	Keyword **proto**
2C*yy*	Keyword **disablella**
2D*yy*	Keyword **tcpsok**
2E*yy*	Keyword **tag**
2F*yy*	Keyword **convserv**
30*yy*	Keyword **secure**
31*yy*	Keyword **rpcbind**
32*yy*	Keyword **accesschk**
33*yy*	Keyword **public**
34*yy*	Keyword **stringprep**

Notes:

xxx Offset to the beginning of the **mount** parameter of the bad keyword.

yy See Table 74 on page 439 for more details.

You can use Table 74 for any additional translation of the reason code information presented in client messages related to parsing errors.

Table 74. Parsing error (when reason code is from 6E0111yy to 6E0133yy

Last 2 hex digits of reason code	Description
01	Null host name or null path name
02	Blank detected
03	Incorrect member name in the path name
04	Missing double quotation mark
05	No member name found
06	Missing left parenthesis
07	Incorrect number
08	Number is larger than 2 GB
09	Incorrect multiplier; must be K, M, or G
0A	Missing right parenthesis
0B	The specified number is not within the allowable range
0C	Incorrect keyword parameter value
0D	Mutually exclusive keyword/option
0E	Keyword is not allowed in the mount option
0F	Keyword is not allowed in the installation parameter

Reason codes from NFS Client or NFS Server modules (*xx* is 10-FF)

Reason codes with an *xx* value in the range 10 to FF are issued by NFS Client or NFS Server modules. The *xx* value indicates which module issued the reason code.

USS JR*cccc* reason codes (0000-0FFF)

A *yyyy* value in the range 0000 to 0FFF indicates that the reason code matches one of the USS JR*cccc* reason codes described in *z/OS UNIX System Services Messages and Codes*, SA22-7807. These reason codes can potentially appear for any NFS Client or NFS Server module.

Table 75 lists the NFS reason codes that have a *yyyy* value in the range 0000 to 0FFF.

Table 75. NFS reason codes that match USS JRcccc reason codes (0000-0FFF)

yyyy	Name	Description
001D	JREstaeErr	The ESTAE macro failed. **Action**: See your IBM service representative.
0026	JRKernel Ready	The system is not in a ready state. **Action**: Retry after OMVS has been allowed to complete initialization.
002E	JRFilesysNotThere	The file system named does not exist. **Action**: The file system specified on the service could not be found.

Table 75. NFS reason codes that match USS JRcccc reason codes (0000-0FFF) (continued)

yyyy	Name	Description
003D	JRDirNotFound	Request failed because the z/OS NFS component cannot find the specified directory. **Probable cause**: 1. The requested directory is removed / renamed / migrated / etc. 2. The name of a non-existing directory is specified. **Action**: Consult the system administrator to ensure use of proper parameters.
0051	JRInvalidName	Request failed because the z/OS NFS component cannot validate the specified name of the object to operate. **Probable cause**: 1. An incorrect parameter (such as, file system name or remote object name) has been specified. 2. No name is specified when it is mandatory. 3. The requested object was removed / renamed / migrated / etc. **Action**:
0058	JRLskOffsetIsInvalid	The offset given for lseek service is incorrect. **Probable cause**: The offset value is higher than the maximum file size that the remote side allows. **Action**: Consult the system administrator about remote system limits to ensure use of proper parameters.
006B	JRBuffTooSmall	The buffer for return information is too small. **Action**: The length of the buffer specified on the service was not large enough to contain the data to be returned.
0078	JRPathNotDir	Request failed because the z/OS NFS component cannot find the specified path or the specified path is not a directory. **Probable cause**: 1. The directory name is expected in operations with the directory (such as, opendir and readdir). 2. The requested directory is removed / renamed / migrated / etc. 3. The name of a non-existent directory is specified. **Action**: Consult the system administrator to ensure use of proper parameters.
007A	JRDirInUse	Request failed because the z/OS NFS component has found that the specified directory is in use. **Probable cause**: The requested directory is in use. You tried to operate on (such as, remove or rename) a directory that contains a file that is open by some process. **Action**: Close or wait for all files of this directory in all other processes.
0083	JRKernelDown	The kernel has ended during this service. **Action**: z/OS UNIX ended during this service. Ask the operator to enter the command to start z/OS UNIX. Then reissue the failing service.
0090	JRNotForDir	The path specified for the READ/WRITE operation is directory. **Action**: Consult the system administrator to ensure use of proper parameters.

Table 75. NFS reason codes that match USS JRcccc reason codes (0000-0FFF) (continued)

yyyy	Name	Description
00A9	JRInvalidParms	Request failed because of a combination of inconsistent parameters. **Probable cause**: 1. The operator specified inconsistent parameters (for example, the NFS version number or operation code). Different NFS versions support different sets of operations. 2. Parameters are set to invalid values (for example, pointers/addresses are NULL or length is zero). **Action**: Consult the system administrator to ensure use of proper parameters.
00AB	JRFsUnmountInProgress	An unmount service is already in progress. **Action**: The file system named is being unmounted.
00B4	JRQuiesced	There was a previous quiesce request. **Action**: The file system required for the current function has been quiesced. After the file system has been unquiesced, retry this service.
00B6	JRPfsSuspend	The PFS is waiting to restart. **Action**: If there is a WTOR prompt on the operator console, the PFS will be restarted when the reply is issued. Otherwise, the PFS will be restarted by its own procedures. Close and re-open the socket or file descriptor and retry the request again after the PFS is active. This value may also be returned if there is a configuration problem and the address space is not connected to the proper PFS.
00B7	JRPfsAbend	z/OS NFS abended. **Action**: z/OS NFS abended on the last request. Try the request again. Report this to support personnel if the problem persists.
00C5	JRResrPortNotAvail	The TCP MOUNT/UNMOUNT request requires the reserved port. The z/OS NFS component cannot find the required port on the specified server. **Probable cause**: 1. The NFS server on the remote side did not start up. 2. The system administrator does not allow access to reserved ports on the remote side. **Action**: Consult the system administrator to ensure the TCP/IP configuration is correct.
0115	JRBuffLenInvalid	The length of the buffer is less than or equal to zero or it is less than a minimum length. **Probable cause**: The buffer length specified for this request was a negative number, zero, or less than a minimum length. **Action**: Retry the request specifying a valid buffer length parameter.
011A	JRInvalidSymLinkLen	The content of the symbolic link is NULL or empty. **Action**: The mount operation fails.
011C	JRFileNotOpen	The file is not opened. **Action**: Reissue the request specifying an open file descriptor.
0130	JRSigDuringWait	A signal occurred during a wait. **Action**: While the service was waiting, a signal was received to interrupt it.

Table 75. NFS reason codes that match USS JRcccc reason codes (0000-0FFF) (continued)

yyyy	Name	Description
014E	JRFsFailLock	Request failed because the z/OS NFS component could not set the lock. **Probable cause**: The lock is held by another request. **Action**: Wait until the lock is released.
018F	JRQuiescing	The call did not complete. The file system is unmounting. **Action**: The requested function cannot be performed while an unmount is in progress for a file system. Retry when the file system is mounted again.
01AB	JRFsInUse	The requested file system is still in use. **Action**: A normal unmount was requested for the file system. There is at least one process still using the file system, so the request to unmount cannot be honored.
0211	JRTimeOut	The time for the service to wait has expired. **Action**: While the process was waiting for signals or a condition to occur, the wait time specified expired.
0291	JRTcpError	TCP returned an error identified by the return code. **Action**: Correct the error reported by the return code.
0296	JRTcpNotActive	No AF_INET socket provider is active. **Action**: Start the AF_INET socket provider you specified in parmlib and retry this socket request.
029E	JRInvalidVlok	The supplied VLock structure is not valid. **Action**: Ensure that the values provided in the VLock_Length parameter and the VlokID and VlokLen fields are correct.
0350	JRAsynchMount	The request to mount a file system will complete asynchronously. The system rejects all vnode (file) operations against the file system. **Action**: Use w_getmntent to determine when the mount completes.
0351	JRPfsOpNotSupported	z/OS NFS does not support the pfsctl command. **Action**: Ensure that z/OS NFS supports the operation.
0352	JRPfsOpNotPermitted	Not authorized to perform this pfsctl operation. **Action**: The request must be made by an authorized user.
03A5	JROWaitSetupErr	An error occurred while attempting OsiWait setup **Action**: Contact your IBM service representative.
03BD	JRFsUnAuthClnt	Request failed because of no authorization on the remote server (no AUTH_UNIX). **Probable cause**: The remote server cannot properly recognize the user credential. **Action**: Consult the system administrator to ensure the authorization is correct.

Table 75. NFS reason codes that match USS JRcccc reason codes (0000-0FFF) (continued)

yyyy	Name	Description
03CA	JROutOfMountEntries	Request failed because the z/OS NFS component cannot find the specified mount point. **Probable cause**: Incorrect mount parameters (such as the file system name and mount point) have been specified. **Action**: Consult the system administrator to ensure the use of proper parameters.
0421	JRFileInUse	Request failed because the z/OS NFS component has found that the specified file is in use. **Probable cause**: The requested file is in use. An attempt was made to operate (for example, remove or rename) on a file that is open by some process. **Action**: Close or wait for this file in all other processes.
0469	JRInvalidOption	The option specified is not supported. **Action**: Reissue the request with a supported option.
04B3	JRInRecovery	A required file system is being recovered. **Action**: Retry the operation later.

Global reason codes (*yyyy* = 1000 - 3FFF)

A *yyyy* value in the range 1000 to 3FFF indicates that the reason code is a global reason code, whose meaning is the same regardless of which NFS module issued it.

Table 76 shows the *yyyy* values that can be issued by any NFS client module:

Table 76. NFS client global reason codes (1000 - 3FFF)

yyyy	Name	Description
1001	JRNfs_FileidChanged	NFSC recovering FHEXPIRED but the lookup object has a different fileid than before. It is possible that during the NFSS interruption, the "old" object is removed and a "new" object is created with the same name, but the fileid is different. NFSC treats it as ESTALE. **Action:** The operation fails. Try to traverse backward to the parent directory, or to check the object existence, and reissue the operation.
1002	JRNfs_StaleObject	NFSC recovering FHEXPIRED but the object seems stale or fails its consistency check. It is possible that the intermediate directory at the Server is removed during NFSS interruption. **Action:** The operation fails. Try to traverse backward to the parent directory, or to check the object existence, and reissue the operation.
1003	JRNfs_MntRetry	Exhausting the retry count while attempting to contact the specified NFS Server. **Action:** The mount command fails. Verify the NFS Server availability and reissue the mount command with a larger "retry" value.
1004	JRNfs_PermissionBits	The Object Permission Mode bits deny the operation. **Action:** The operation fails. Verify the Object Permission Mode bits (RWX), obtain the proper authority (Owner, Group, Others), and reissue the operation.

Table 76. NFS client global reason codes (1000 - 3FFF) (continued)

yyyy	Name	Description
1005	JRNfs_SymLinkLoop	NFSv4 Mount Emulation encounters an already-resolved symbolic link. A loop of symbolic links is detected. **Action:** The mount operation fails.
1006	JRNfs_MntObjNotDirSymlnk	NFSv4 Mount Emulation encounters an object that is neither a directory nor a symbolic link. **Action:** The mount operation fails.
1007	JRNfs_RpcsecNotSupported	An RPCSEC_GSS request failed because of one of the following reasons: 1. The Kerberos Dynamic Load Library could not be loaded by the NFS Client. 2. The Kerberos context could not be created by the NFS client. 3. The GSS credentials for the NFS client's Kerberos Principal could not be obtained. **Action:** The operation fails. Check your kerberos configurations, fix any configuration issues identified, restart the NFS client and retry the operation. Contact the system programmer if the problem persists.
1008	JRNfs_RpcsecCryptographicFailure	An RPCSEC_GSS request failed because either the cryptographic generation or cryptographic verification of the request failed. **Action:** The operation is retried once and if the retry fails, the operation is failed. Contact the system programmer if the problem persists.
1009	JRNfs_RpcsecKinitNeeded	An RPCSEC_GSS request failed because the user's Kerberos Credentials have expired. **Action:** The operation is retried once and if the retry fails, the operation is failed. Reacquire the Kerberos credentials by issuing a "kinit" and retry the operation. Contact the system programmer if the problem persists.
100A	JRNfs_HostNotFound	Name Resolver is unable to resolve the given host name to an IP address. Action: The command fails. • Try **nslookup** to ensure the given host name can be resolved into an IP address by the Name Resolver. • If **nslookup** fails, contact the responsible network administrator to correct the Name Resolver configuration or the Name Server. • If **nslookup** works, ensure SYSTCPD DDname in the StartUp Procedure or Jobname DOES NOT alter the Name Resolver search order or search path. It is possible to remove SYSTCPD DDname so the Name Resolver search order for the successful nslookup applies to z/OS NFS. Changing the z/OS NFS StartUp Procedure requires restarting the z/OS NFS component.
100B	JRNfs_NotTimedout	The z/OS NFS component has found that no actions are required due to requested resource is not timed out yet. **Probable cause**: The requester has not waited long enough, or a request has been processed that updated the timeout value. **Action**: Wait until the timeout is expired.
100C	JRNfs_Inconsistency	The z/OS NFS component loadlib contains inconsistent modules. **Probable cause**: Modules of different versions are used in loadlib. **Action**: Correct loadlib and restart the z/OS NFS Client.

Table 76. NFS client global reason codes (1000 - 3FFF) (continued)

yyyy	Name	Description
100D	JRNfs_UnicodeFail	The Unicode Conversion Service fails on translation. **Probable cause**: Code page ids (ccsid) in z/OS NFS parameters (specified as clnccsid() in BPXPRMxx NFS section or in MOUNT command) don't match code page ids that are available in Unicode Conversion Service. **Action**: Setup ccsids to proper value and restart the z/OS NFS Client.
100E	JRNfs_NoLocalHost	The z/OS NFS component cannot get local host name. **Probable cause**: There is an error in the resolver configuration file – (HOSTname keyword). **Action**: Ensure the resolver configuration file is correct and restart z/OS NFS Client.
100F	JRNfs_NoLocalDomain	The z/OS NFS component cannot get its own local domain name. **Probable cause**: There is an error in the TCPIP configuration data sets – (keywords Domain or DomainOrigin). **Action**: Ensure the TCPIP configuration data sets are correct and restart the z/OS NFS Client.
1010	JRNfs_PortmapUnsupp	The z/OS NFS component cannot use PORTMAP RPC protocol on a true IPv6 network. **Probable cause**: The remote side of a TCPIP address isn't type IPv4 or IPv4mapped_IPv6, or rpcbind(n) option is specified. **Action**: Remove 'rpcbind(n)' or use IPv4 addressing..
1011	JRNfs_InvalidFileMode	The request failed because the specified file mode is not supported by the z/OS NFS component. **Probable cause**: An incorrect file mode has been specified. **Action**: Consult the system administrator to ensure use of proper parameters.
1012	JRNfs_ServerNotActive	The request failed because no active NFS server was not found on the remote side. **Probable cause**: An incorrect NFS server address has been specified or NFS server on remote side hasn't started yet. **Action**: Consult the system administrator to ensure use of proper parameters or wait until remote NFS server starts.

Table 76. NFS client global reason codes (1000 - 3FFF) (continued)

yyyy	Name	Description

1013	JRNfs_Wrongsec	The request failed because the RPC authentication flavor with which the request was issued was not allowed by the security policy being enforced at the server's name space and a security negotiation either could not be done for this request or a security negotiation was attempted for this request, but it failed.
		Probable cause: The security negotiation may not have been done for this request for one of the following reasons:
		1. This failure was encountered during a mount, where a security flavor was specified in the "secure" keyword in the mount parameters.
		2. This failure happened on an existing file or directory that had datacaching enabled.
		The security negotiation may have failed for this request for one of the following reasons:
		1. The security policy on the server name space does not support any of the following security flavors:
		`sys`
		`krb5`
		`krb5i`
		`krb5p`
		2. There was a change in the server security policy whereby it was downgraded from a more secure security flavor to a less secure security flavor.
		Note: The flavors are listed below in descending order of the level of protection that they provide:
		`krb5p`
		`krb5i`
		`krb5`
		`sys`
		Action: The request fails.
		Ensure the following:
		1. If this failure is encountered during a Mount, do one of the following:
		• Do not specify a security flavor in the "secure" keyword, to facilitate the client to dynamically negotiate the security policy with the server.
		• If a security flavor must be specified in the "secure" keyword, ensure that the specifed flavor conforms to the security policy at the server's name space.
		2. If this failure was encountered on an existing file or directory that had datacaching enabled, then to recover from this failure, datacaching would have to be disabled on that object by mounting with datacaching(n).
		3. The NFS server being communicated with must support at least one of the following security flavors:
		`sys`
		`krb5`
		`krb5i`
		`krb5p`
		4. To get around a security policy downgrade on the server name space for an existing mount point,the mount point must be unmounted and remounted and the request must be retried.

Module specific reason codes (*yyyy* = 4000 - 4FFF)

A *yyyy* value in the range 4000 to 4FFF indicates that the reason code is a module specific reason code, whose meaning can vary, depending on which NFS module issued it.

Table 77 shows the *yyyy* values that can occur when the *xx* value is 12 (module GFSCVNAT) and their meanings:

Table 77. Reason codes for module GFSCVNAT (xx = 12)

yyyy	Name	Description
4001	JRNfsInvalidTimeAttr	Versions 2 and 3 of the NFS Protocol do not support second time values larger than 2**31-1. The request attempted to set an atime and/or mtime value larger than 2**31-1
4002	JRNfsInvalidAttr	The request attempted to set one or more of the following attributes, which are not supported by the NFS Protocol: general attribute flags (at_setgen) audit (at_auditchg) ctime (at_ctimechg) reftime (at_reftimechg) file format (at_filefmtchg)
4003	JRNfsInvalidSetIdAttr	The request attempted to set the character set id (at_charsetidchg) attribute, which is not supported by the NFS Protocol.

Table 78 shows the *yyyy* values that can occur when the *xx* value is 18 (module GFSCVMNT) and their meanings:

Table 78. Reason codes for module GFSCVMNT (xx = 18)

yyyy	Name	Description
4001	JRNfsSyncMountNotSupp	NFS mount requests must be asynchronous. The mt_synchonly option is not supported. The unsupported option may be the result of a chmount request issued against a function shipped NFS mounted file system. #action = Correct the request.
4002	JRNfsRemountNotSupp	An NFS mount request specified the mt_remount. NFS does not support this option.

Appendix A. File size value for MVS data sets

Many NFS procedures (such as nfs_lookup and nfs_getattr) in the NFS protocol require the file size to be returned. This Topic explains some performance and accuracy considerations in obtaining the file size value for MVS data sets. For z/OS UNIX files, the file size is directly available from the underlying file system because it is saved as part of the file metadata.

The meaning of the file size value that is returned by the NFS and how fast the file size is returned depends on the following conditions:

- Whether you use text or binary processing mode
- The type of MVS data set being accessed
- If the data set is system-managed
- If **fastfilesize** processing is used

Storage of the file size value

How the file size value is stored affects how quickly files are accessed and depends on the type of MVS data set used.

System-managed PS, VSAM, and PDSE data sets

Text and binary file size are saved on non-volatile storage (DASD) for quick access and maintained by the server for these data set types:

- Physical sequential (including striped)
- VSAM ESDS
- VSAM KSDS
- VSAM RRDS
- PDSE members

These data sets must be SMS managed. When the NFS accesses a data set for the first time, it performs a read-for-size to get the text or binary file size and stores this value on DASD. Subsequent file size requests from clients do not cause the server to read for size, thus improving performance. However, when the data set is modified outside the server by a non-NFS application (for example, by the TSO/E editor), the stored file size could be incorrect. When the data set is accessed again by the server, read-for size must be done to determine the correct file size.

Migrated system-managed data sets

z/OS DFSMS allows data set attribute accessibility for SMS managed data sets, without having to recall the data set if the data set is migrated under DFSMS/MVS V1R3 or later. Supported data set types are SMS managed PS, VSAM ESDS, VSAM KSDS, VSAM RRDS, PDS, and PDSE. Migrated PDS/PDSE members are not supported.

The z/OS NFS server is able to obtain the attributes of a supported SMS managed migrated data set without recalling the data set. Attributes such as the record format and file size are saved to DASD. Subsequent file size requests do not cause a recall of the supported SMS managed migrated data set, thus improving performance. However, when the data set is modified outside the server by a

© Copyright IBM Corp. 1991, 2010

non-Network File System application (for example, by the TSO/E editor) before it was migrated, the stored file size could be incorrect. When the data set is accessed again by the server, a recall must be done to determine the correct file size.

Non-system-managed, PDS, and direct data sets

The file size value for non-system-managed data sets, PDS members, and direct (DA) data sets is cached in virtual storage until timeout but not written to DASD. Therefore, for these types of MVS data sets, the file size value is regenerated after the file is closed or after the server is restarted.

How the file size value is generated

When a file is first accessed (for example with **ls -l** or **dir**), usually the entire file is read to determine its size, except for **recfm(f)** or **recfm(fbs)** where the binary size can be computed without reading the file. If the file is a system-managed PS, VSAM, or PDSE member, both binary and text file sizes are stored on DASD, so that subsequent file size requests do not require the file to be read.

Binary file size can be quickly generated by using **recfm(f)** or **recfm(fbs)** to specify a fixed-length record format for the MVS data set. With this format type, the server pads the last logical record with binary zeros in binary mode processing, because MVS always expects complete logical records. If the application tolerates these zeros, using **recfm(f)** or **recfm(fbs)** allows the binary size to be computed quickly because the number of bytes can be computed from the number of blocks, which is stored by MVS.

If you need the exact file size and are using binary mode processing, map it to a variable-format, sequential data set on DASD so that the NFS does not need to pad a partially filled last MVS logical record to a record boundary.

For reading small files or the beginning of files, the read-for-size might not add any processing time. As the file is being read for size, the beginning of the file is stored in the buffers set aside by the **maxrdforszleft** site attribute, until the buffers are full. When the application reads the beginning of the file, this read is fast because it reads directly from the buffer.

MVS stores the number of blocks (rather than the number of bytes) in an MVS file. For most files, therefore, without reading the entire file, the NFS can only give an estimate of the number of bytes in the file, not the exact number of bytes in the file. Even when the server could get the exact byte count without reading the file, the file size could change depending on the file's processing attributes.

For example, selecting **text** mode processing introduces end-of-line terminators such as **lf**, **crlf**, or **\n** into the file, thus changing the perceived size of the file. As another example, suppose you select text mode processing with blank stripping enabled on a fixed-length record format file. That causes the server to remove trailing blanks from each record, again changing the perceived size of the file. In these examples, when you first request a file, the server must read the entire file to determine its exact size in bytes.

Using fastfilesize to avoid read-for-size

If you can use an approximate file size for a PDS, PDSE, DA, or non-system-managed data set, you can specify the **fastfilesize** attribute to improve performance. With this attribute, the server estimates the size without opening and reading the entire file.

PDS members
For PDS and PDSE members, the fastfilesize attribute gets the file size from SPF statistics if they exist; otherwise, a zero file size is returned.

DA data sets
For direct access (DA) data sets, an approximate file size is calculated based on the device characteristics, the number of disk tracks in use, and the block size of the data set.

VSAM
For non-system-managed VSAM data sets, the estimated size using fastfilesize is zero. Therefore, such commands as **cat** or **vi** will usually not show any data. In some instances, if the NFS server already has the actual file size cached, data may be shown. Once the cache times out, however, the size is lost and the commands do not show any data.

The **fastfilesize** attribute speeds up data set access by calculating approximate file sizes during data set access. Use this only when you are browsing through files (using the **ls** UNIX command for example) because some commands (such as **cp** or **copy**) might not work correctly if fastfilesize is set. When reading or modifying a data set, the **nofastfilesize** attribute should be used to ensure accurate results.

Using nofastfilesize

When you use the default, **nofastfilesize** attribute, the NFS reads the entire file or member to get the file size. It stores the file size value in cache until timeout. If the server's default has been changed to **fastfilesize**, you can still use the nofastfilesize attribute to override it.

```
$ ls -l "filename,nofastfilesize"
```

Using this attribute might cause a delay when first accessing very large data sets.

Note: When directly mounting on a fully qualified data set name and nofastfilesize is specified, the server must return the mount size as part of getting the attributes for the mount. This can slow down the completion of the mount command.

Appendix B. Time stamps for MVS data sets

This topic explains how to obtain time stamps for MVS data sets. For z/OS UNIX files, the time stamp is directly available from the underlying file system because it is saved as part of the file metadata.

UNIX file attributes define the following time stamps:

atime The last time the file was accessed (read).

mtime The last time the file was modified (write).

ctime The last time the file status was changed (chmod).

The NFS handles time stamps differently for these types of data sets:

- System-managed PS data sets and system-managed VSAM data sets
- Direct data sets and non-system-managed PS data sets
- Non-system-managed VSAM data sets
- PDS and PDSE members

The z/OS NFS server treats all MVS data sets as belonging to the same file system (having the same file system id). Since not all attributes are supported for all DFSMSdfp access methods, but the NFS protocols track attribute support at the file system level, the z/OS NFS server reports to the client that **atime** and **mtime** are supported for all MVS data sets. The z/OS NFS server will generate values as described in this topic.

The **metatime** attribute, which is new in NFS version 4, is not supported by any DFSMSdfp access method, nor by the z/OS NFS server. The new NFS version 4 **change** attribute will be based on **mtime** values as described in this topic.

Time stamps for system-managed VSAM and PS data sets

For system-managed PS data sets and system-managed VSAM data sets, **atime** and **mtime** are fully maintained, and the **ctime** value is set to the **mtime** value.

Time stamps for non-system-managed PS and DA data sets

For non-system-managed physical sequential (PS) and direct access (DA) data sets, consider the following conditions:

- How time stamps are stored.
- The requirements of your workstation programs.
- The type of multiple virtual system (MVS) data set used to store the file.

Storage of time stamps

For non-system-managed physical sequential (PS) and direct access (DA) data sets, the Network File System temporarily stores the time stamps in virtual storage, but not on direct access storage device (DASD). These cached attributes are purged when the file times out and closes or when the server is restarted. When the file is accessed again, the time stamps are regenerated.

© Copyright IBM Corp. 1991, 2010

Client program requirements

Some workstation-based utilities (such as **make**) rely on date and time stamps to determine whether to recompile. For example, **make** checks the update time of the object file with the source file and recompiles if the source has been updated. Before storing these types of files using the MVS server, examine them to ensure that these attributes are unimportant. In an environment which relies on such utilities, use system-managed PS data sets.

Generating time stamps

This is how NFS generates **atime** and **mtime** values for non-system-managed PS and DA data sets from the MVS dates:

```
atime = mtime = reference_date + time_increment
ctime = creation_date + time_increment
```

The *time_increment* value is either the server local time or 23:59 hours. If *reference_date* value or *creation_date* value is equal to the server local date, the server local time is added. Otherwise, a fixed value of 23 hours and 59 minutes is added.

If the *reference_date* value is zero (that is, the file has not yet been referenced), the **atime** value and the **mtime** value are set equal to **ctime**.

Time stamps for non-system-managed VSAM data sets

The time stamps for these types of data sets are set to the current time.

Time stamps for PDSs and PDSEs

An MVS PDS or PDSE data set can act as a UNIX directory, when mounted by an NFS client to the z/OS NFS server. Members of the PDS or PDSE data sets are files within the UNIX directory. When the client accesses the directory, UNIX-format file time stamps are expected for each file on the client side. File time stamps in UNIX format are part of the attributes required by the NFS protocol for NFS client/server communication.

Based on the NFS protocol, the z/OS NFS server generates the following UNIX time stamps to send to the client:

atime
the time when the file data was last accessed

mtime
the time when the file data was last modified

ctime
the time when the attributes of the file were last changed.

The z/OS NFS server converts MVS time stamps to UNIX time stamps (and vice versa) to match NFS protocol requirements. The server uses the following main time stamp sources to generate UNIX time stamps for MVS z/OS conventional (legacy) file systems:

- DSCB (data set control block)
- Master Catalog data set attribute extension (AX) cell
- PDSE member attribute extension (AX) cell
- ISPF member statistics
- TOD (current time_of_day on the server side).

Time stamp generation depends on the NFS operation (such as read, write, or setattr) and the type of data set.

TOD is used to set up current times in internal NFS control blocks if needed.

For PDS or PDSE member create/update access with ISPF, some specific additional statistics for the member are maintained by ISPF. They include the creation date and the last modification date and time. The server supports ISPF statistics for compatibility with TSO/ISPF. The server always creates ISPF statistics for new PDS/PDSE members created by NFS clients. For existing PDS/PDSE member updates by the client, the server creates/updates the member ISPF statistics.

The following tables summarize the time stamp sources for NFS operations when obtaining file attributes for PDS (SMS-managed and non SMS-managed) and PDSE data sets and members.

Table 79. Time stamp sources for PDS and PDSE members

		Time stamp sources for members (files)		
Data set type	**ISPF statistics**	**atime**	**mtime**	**ctime**
PDS (note 1)	Available	ISPF_ modification_ date + ISPF_ modification_ time	ISPF_ modification_ date + ISPF_ modification_ time	ISPF_ creation_ date + time_ increment
	Not available	DSCB_ reference_ date + time_ increment	DSCB_ reference_ date + time_ increment	DSCB_ creation_ date + time_ increment
PDSE (note 2)	Available (not used for time generation)	PDSE AX cell	PDSE AX cell	ctime = mtime

Notes:

1. MVS does not maintain time stamps for members of a PDS, only for the PDS data set. MVS creation and reference dates are maintained in the DSCB control block.

   The UNIX time stamps for PDS members are generated from the DSCB creation and reference dates of the PDS data set containing the members, if the time stamps cannot be generated from the member's ISPF statistics.
2. MVS maintains the PDSE member create/change time stamp (mtime) in the PDSE AX cell. The Server uses a FileAccessMethodService (FAMS) call to retrieve/save the member attributes (containing time stamps) from/to the PDSE AX cell.

   For a PDSE data set/member, the server generates the UNIX time stamps obtained from the Catalog AX cell/PDSE AX cell. The server supports PDSE member's ISPF statistics just for compatibility with ISPF but does not return them to the client.

Table 80. Time stamp sources for PDS and PDSE data sets (directories)

	Time stamp sources for data sets (directories)		
Data set type	**atime**	**mtime**	**ctime**
PDS (non-SMS)	DSCB_ reference_ date + time_ increment	DSCB_ reference_ date + time_ increment	DSCB_ creation_ date + time_ increment

Table 80. Time stamp sources for PDS and PDSE data sets (directories) (continued)

PDS (SMS)	catalog AX cell	catalog AX cell	ctime = mtime
PDSE (SMS)	catalog AX cell	catalog AX cell	ctime = mtime

In Table 79 on page 455 and Table 80 on page 455, *time_increment* is either the server local TOD or 23:59 hours. If *reference_date* or *creation_date* is equal to the server local date, the server local TOD is added. Otherwise, a fixed value of 23 hours and 59 minutes is added. If *reference_date* = 0 (that is, the file has not yet been referenced), **atime** and **mtime** are set equal to **ctime**.

MVS maintains PDSE/SMS-managed PDS atime, mtime time stamps in the Catalog AX cell. The Server uses an SVC26 call to retrieve/save the data set attributes (containing time stamps) from/to the Catalog AX cell.

The server keeps all file time stamps current in internal control blocks in readiness to return time stamps when servicing NFS requests issued from NFS clients. The server updates the time stamp sources (in the DSCB, member ISPF statistics, or attribute extension cells) from internal control blocks only during close file operations depending on file timeout expiration values (attrtimeout, readtimeout, writetimeout) as described in section 'Timeout attributes'.

Setting time stamps

NFS clients can issue SETATTR requests to set **atime** and **mtime** for a system-managed PS or VSAM data set. For PDSE members, setting **mtime** is allowed, but setting **atime** is not supported. PDSE member **mtime** is also maintained by PDSE access methods, so it is modified when a TSO/E user modifies the PDSE member.

Appendix C. NFS server attributes

This topic lists NFS server attributes and how they are supported on the z/OS NFS server.

NFS file system attributes for MVS data sets

The z/OS NFS server generates MVS-specific values for certain UNIX file system attributes. Table 81, Table 82, and Table 83 illustrate the MVS values that the z/OS NFS server generates.

Table 81. File system values to get dynamic file system information

Value	Description	z/OS Conventional MVS Value
tbytes	Total size, in bytes, of the file system	10000000000
fbytes	Amount of free space, in bytes, in the file system	8000000000
abytes	Amount of space, in bytes, available to the user identified by the authentication in the RPC	80000000
tfiles	Total number of file slots in the file system	200000
ffiles	Number of free file slots in the file system	20000
afiles	Number of free file slots that is available to the user corresponding to the authentication information in the RPC	2000
invarsec	Number in seconds for which the file system is not expected to change	0

Table 82. File system values to get static file system information

Value	Description	z/OS Conventional MVS Value
rtmax	Maximum number in bytes for the read request supported by the server	65536 (64KB)
rtpref	Preferred size of the read request	32768 (32KB)
rtmult	Suggested multiple for the size read request	4096
wtmax	Maximum size of a write request supported by the server	65536 (64KB)
wtpref	Preferred size of the write request	32768 (32KB)
wtmult	Suggested multiple for the size of a write request	4096
dtpref	Preferred size of the readdir request	8192
maxfilesize	Maximum size of a file on the system	2 ** 63 - 1
time_delta	File time using setattr	(0,1000000)
Properties:	FSF_LINK FSF_SYMLINK FSF_HOMOGENEOUS FSF_CANSETTIME	1 0 1 1

Table 83. File system values to retrieve POSIX information

Value	Description	z/OS Conventional MVS Value
linkmax	Maximum number of hard links	1
name_max	Maximum length of a component file name (file name + attributes)	255

© Copyright IBM Corp. 1991, 2010

Table 83. File system values to retrieve POSIX information (continued)

Value	Description	z/OS Conventional MVS Value
no_trunc	Server will reject any name that is longer than the name_max	True
chown_restricted	Change either the owner or the group associated with the data set	True
case_insensitive	Server does not distinguish the case when interpreting file names	True
case_preserving	If True, the server file system will preserve the case of a name during a create, mkdir, mknod, symlink, rename, or link	False

NFS file system attributes for z/OS UNIX file systems

For z/OS UNIX files, Table 84 and Table 85 show the file system values that are returned for NFS attributes.

Note: Static UNIX values are for general reference only and may change. Actual values are provided by UNIX System Services or an underlying physical file system such as zFS, TFS, HFS, or NFS client. Refer to these products for current values.

Table 84. File system values to get static file system information

Value	Description	UNIX value
rtmax	Maximum number in bytes for the read request supported by the server	65536 (64KB)
rtpref	Preferred size of the read request	32768 (32KB)
rtmult	Suggested multiple for the size read request	4096
wtmax	Maximum size of a write request supported by the server	65536 (64KB)
wtpref	Preferred size of the write request	32768 (32K)
wtmult	Suggested multiple for the size of a write request	4096
dtpref	Preferred size of the readdir request	8192

Table 85. File system values to retrieve POSIX information

Value	Description	UNIX Value
linkmax	Maximum number of hard links	2 ** 31
name_max	Maximum length of a component file name (file name + attributes)	255
no_trunc	Server will reject any name that is longer than the name_max	True
chown_restricted	Change either the owner or the group associated with the data set	False
case_insensitive	Server does not distinguish the case when interpreting file names	False
case_preserving	If True, the server file system will preserve the case of a name during a create, mkdir, mknod, symlink, rename, or link	True

NFS protocol attributes

Table 86 provides general reference information about the z/OS NFS server's support of NFS protocol attributes. To determine which attributes are communicated between the client and server for the version of the NFS protocol that you are using, see the corresponding RFC at the Internet Engineering Task Force (IETF) home page http://www.ietf.org/ :

- NFS protocol version 2: RFC 1094
- NFS protocol version 3: RFC 1813
- NFS protocol version 4: RFC 3530.

Table 86. NFS Version 4 Attributes

Num	Attribute Name	Supp_attr value (Note 1, 5)	Comments (Note 2)
1	type	1	File object type (File, Directory, Link, etc.) (Note 5) **UNIX value:** at_mode, NFALL_B **MVS value:** NFS generated
2	fh_expire_type	1	Indicates a file handle is persistent or volatile across server restarts. (Note 5) **UNIX value:** NFSV4 fh4_volatile_any **MVS value:** NFSV4 fh4_volatile_any
3	change	1	Server generated value which is updated when an NFS object attribute or content changes. **UNIX value:** at_ctime64, at_ctimemsec **MVS value:** NFS server generated (Notes 4, 5)
4	size	1	Object size in bytes. **UNIX value:** at_size or 512 if an empty directory (Note 5) **MVS value:** (Note 3)
5	link_support	1	Objects file system supports hard links (Note 5) **UNIX value:** fs_nfsprop.fs_fsf_link **MVS value:** 0 (not supported)
6	symlink_support	1	Objects file system supports symbolic links (Note 5) **UNIX value:** fs_nfsprop.fs_fsf_symlink **MVS value:** 0 (not supported)
7	named_attr	1	Object has named attributes (Note 5) **UNIX value:** 0 (not supported) **MVS value:** 0 (not supported)
8	fsid	1	Unique file system id (Note 5) **UNIX value:** at_dev **MVS value:** (Note 8)
9	unique_handles	1	Distinct file handles are guaranteed to represent different objects. **UNIX value:** 0 **MVS value:** 0
10	lease_time	1	Duration of lease in seconds. **UNIX value:** NFS server site attribute. **MVS value:** NFS server site attribute
11	rdattr_error	1	Readattr error during getattr operation.**UNIX value:** error enum **MVS value:** error enum
12	ACL	UNIX: 1 MVS: 0	Object's associated access control list **UNIX value:** Object's ACL values **MVS value:** undefined
13	aclsupport	1	Types of ACLs supported by the server **UNIX value:** 1 **MVS value:** 0
14	archive	0	Object has been archived since last modification. **UNIX value:** undefined **MVS value:** undefined
15	cansettime	1	Server is able to change times as specified in a setattr **UNIX value:** fs_nfsprop.fs_fsf_CanSetTime **MVS value:** 1
16	case_insensitive	1	Server file name comparisons are case insensitive. (Note 5) **UNIX value:** pcfgcaseinsensitive **MVS value:** 1
17	case_preserving	1	Filename case is persevered on this file system. **UNIX value:** pcfgcaseonpreserving **MVS value:** 0
18	chown_restricted	1	Changes to file owner or group requires privileged user. (Note 5) **UNIX value:** 0 **MVS value: 1**
19	filehandle	1	Objects associated file handle. (Note 5) **UNIX value:** NFS server generated **MVS value:** NFS server generated
20	fileid	1	A number uniquely identifying the file on this file system. (Notes 5, 7) **UNIX value:** BPXYATTR.AttrFid **MVS value:** NFS server generated
21	files_avail	1	Available file slots. (Note 5) **UNIX value:** fs_favail **MVS value:** 2,000

Table 86. NFS Version 4 Attributes (continued)

22	files_free	1	Free file slots. (Note 5) **UNIX value:** fs_ffree **MVS value:** 20,000
23	files_total	1	Total file slots. (Note 5) **UNIX value:** fs_files **MVS value:** 200,000
24	fs_locations	0	Alternate locations where this file system may be found. **UNIX value:** undefined **MVS value:** undefined
25	hidden	0	Hidden file with respect to the Windows API. **UNIX value:** undefined **MVS value:** undefined
26	homogeneous	1	file system attributes are the same for all objects within the file system. **UNIX value:** fs_nfsprop.fs_fsf_homogeneous **MVS value:**1
27	maxfilesize	1	Maximum file size. (Notes 3, 5) **UNIX value:** fs_maxfilesize **MVS value:** see notes.
28	maxlink	1	Maximum number of links for this object. (Note 5) **UNIX value:** pcfglinkmax **MVS value:** 1
29	maxname	1	Maximum filename size for this object. (Note 5) **UNIX value:** pcfgnamemax **MVS value:** 256
30	maxread	1	Maximum read size for this object. (Note 5) **UNIX value:** 65536**MVS value:** 65536
31	maxwrite	1	Maximum write size for this object. (Note 5) **UNIX value:** 65536 **MVS value:** 65536
32	mimetype	0	MIME body type/subtype for this object. **UNIX value:** undefined **MVS value:** undefined
33	mode	1	Support of UNIX-style mode and permission bits. (Note 5) **UNIX value:** at_mode **MVS value:** 666 or 777
34	no_trunc	1	Return error rather than truncate the filename if name is greater than *maxname*. **UNIX value:** pcfgnotrunc **MVS value:** 1
35	numlinks	1	Number of hard links to this object. (Note 5) **UNIX value:**at_nlink **MVS value:** file = 1, directory = 2
36	owner	1	The owner's string name. (Note 9) **UNIX value:** undefined. **MVS value:** undefined..
37	owner_group	1	The owner's group string name. (Note 9) **UNIX value:** undefined. **MVS value:** undefined.
38	quota_avail_hard	0	Server rejects a write request and provides the additional disk space that can be allocated to this file or directory. **UNIX value:** undefined. **MVS value:** undefined.
39	quota_avail_soft	0	Server reasonably warns on a write request and provides the additional disk space that can be allocated to this file or directory. **UNIX value:** undefined. **MVS value:** undefined.
40	quota_used	0	Amount of disk space used by this file or directory. **UNIX value:** undefined. **MVS value:** undefined.
41	rawdev	UNIX: 1 MVS: 0	Raw device identifier (UNIX device major/minor node information). **UNIX value:** at_major, at_minor **MVS value:** undefined.
42	space_avail	1	Disk space available to this user. (Notes 3, 5) **UNIX value:** fs_freespace, fs_blocksize **MVS value:** 80,000,000
43	space_free	1	Filesystem free disk space. (Notes 3, 6) **UNIX value:** fs_freespace, fs_blocksize **MVS value:** 8,000,000,000
44	space_total	1	Filesystem total disk space. (Notes 3, 5) **UNIX value:** fs_usedspace, fs_blocksize **MVS value:** 10,000,000,000
45	space_used	1	Allocated bytes for this file object. (Notes 3, 6) **UNIX value:** at_blocksh, at_blocks, at_blksize **MVS value:** see notes.
46	system	0	System file with respect to the Windows API. **UNIX value:** undefined. **MVS value:** undefined.
47	time_access	1	Last read access satisfied by server for this object. (Notes 4, 5, 6) **UNIX value:** at_atime64 **MVS value:** see notes.
48	time_access_set	1	Setattr operation to set the time of last access for this object. (Notes 4, 5, 6) **UNIX value:** at_atimechg, at_atimeTOD **MVS value:** support is limited, see notes.

Table 86. NFS Version 4 Attributes (continued)

49	time_backup	0	Last backup time for this object. **UNIX value:** undefined. **MVS value:** undefined.
50	time_create	UNIX: 1 MVS: 0	Object time of creation. (Notes 4, 6) **UNIX value:** at_createtime64 **MVS value:** undefined.
51	time_delta	1	Smallest useful server time granularity. (Notes 4, 5) **UNIX value:** fs_time_delta_sec, fs_time_delta_ns **MVS value:** 1 second.
52	time_metadata	UNIX: 1 MVS: 0	Time of last meta-data modification to the object. (Note 4) **UNIX value:** at_ctime64,at_ctimemsec **MVS value:** undefined.
53	time_modify	1	Time of last modification to the object. (Notes 4, 6) **UNIX value:** at_mtime64 **MVS value:** see notes.
54	time_modify_set	1	Setattr operation to set the time of last modification for this object. (Notes 4, 5, 6) **UNIX value:** at_mtimechg, at_mtimeTOD **MVS value:** see notes.
55	mounted_on_fileid	UNIX: 1 MVS: 0	Like *fileid*, but if the target filehandle is a file system root the *fileid* of the underlying file system directory is returned. **UNIX value:** undefined. **MVS value:** same as *fileid*.

Notes:

1. Column indicates z/OS NFS server V4 **supp_attr** attribute setting, where 0 = no support, 1 = supported values. The NFS V4 **supp_attr** is a bitmap of attributes requested, returned, or being set in an NFS client request.
2. UNIX value may contain the UNIX system provided macro and field name. For additional information on UNIX values see *z/OS UNIX System Services File System Interface Reference*.
3. See *z/OS DFSMS Using Data Sets* for additional information on MVS file size limits. See Appendix A, “File size value for MVS data sets,” on page 449 for additional information on file sizes.
4. For z/OS conventional MVS data sets refer to handling of time stamps in Appendix B, “Time stamps for MVS data sets,” on page 453.
5. The constant or generated values are being provided by the NFS server for performance or correct operation.
6. For z/OS conventional MVS data sets, values represent those in the data set control block DSCB, Catalog locate, PDSE directory, HSM MIC cell, and SMS DATACLASS.
7. The *fileid* of z/OS conventional MVS data sets is generated from data set name and member name using **crc32** checksum algorithms.
8. Returned by z/OS UNIX System Services as a result of the NFS server registration request **v_reg()** during server initialization.
9. The z/OS NFS server supports name@domain owner and owner_group strings in this release. These strings are the string name of the owner of file system object and the string name of the group ownership of file system object. The strings are returned and accepted for setattr, verify, nverify, open, create, getattr, and readdir.

Appendix D. NSM (statd) protocol

The NSM (statd) protocol defines network status monitor (NSM) functions for NFS. In z/OS V1R7, the network status monitor, along with the network lock manager (NLM) was integrated into the z/OS NFS address space to improve performance and other functions. This integration changed the statd protocol implementations on the z/OS NFS server.

Using supported NSM (statd) procedures

The NSM protocol defines six RPC procedures which implement the network status manager. With the integration of NSM into the NFS server address space, procedures 0 through 5 act as null procedures and return no results if invoked externally. Procedure 6, however, is fully implemented. The procedures are as follows:

Procedure 0: do nothing
Procedure 0 (NULL) does nothing

Procedure 1: SM_STAT
Dummy call, always return STAT_FAIL

Procedure 2: SM_MON
Monitor a client host

Procedure 3: SM_UNMON
Unmonitor a client host

Procedure 4: SM_UNMON_ALL
Unmonitor all client hosts

Procedure 5: SM_SIMU_CRASH
Simulate a crash

Procedure 6: SM_NOTIFY
NFS server notifies clients that server is restarting, so clients need to reclaim any locks they previously had on server files. This procedure is fully implemented by the z/OS NFS server.

Note: NSM is only active if the NFS server is started with the NLM attribute set, not with NONLM.

© Copyright IBM Corp. 1991, 2010

Appendix E. NFS system server sample attribute table

You can use the contents of the Figure 44 on page 466 attribute table file as a NFS server sample. The attributes table can be found as GFSAPATT in the NFSSAMP library.

© Copyright IBM Corp. 1991, 2010

```
#######################################################################
#                                                                     #
#  Z/OS Network File System Server Sample Attribute Table        @L6C #
#                                                                     #
#     PROPRIETARY STATEMENT=                                          #
#      LICENSED MATERIALS - PROPERTY OF IBM                           #
#      THIS MODULE IS "RESTRICTED MATERIALS OF IBM"                   #
#      5694-A01                                                       #
#      COPYRIGHT IBM CORPORATION 1991, 2010                     @P0IC #
#      SEE IBM COPYRIGHT INSTRUCTIONS                                 #
#     END PROPRIETARY STATEMENT                                       #
#     Copyright IBM Corp. 1991, 2009                            @LBBC #
#     Copyright SUN Microsystems, Inc &                               #
#     Electronic Data Systems Corp. 1988, 1989                        #
#                                                                     #
#######################################################################
#                                                                     #
# change activities:                                                  #
# 1.  5/10/91 - Release it for MVS/DFP V3                             #
# 2.  1/30/92 - Updates for VSCR                                      #
# 3.  8/06/92 - Change mintimeout default                             #
# 4.  8/31/92 - Add PCNFSD                                            #
# 5.  9/20/92 - R2 updates                                            #
# $L3X=NFS,HDZ11NP,931230,SJPLJST: Change NFSTASKS default and   @L3XA#
#                                  add XLAT keyword              @L3XA#
# $L3L=KA90033,HDZ11NP,940405,SJPLJST: Add RESTIMEOUT keyword    @L3LA#
#                                                                     #
# $L33=NFS,HDZ11NP,940613,SJPLTEM:  Add SMF keyword              @L33A#
# $01=OW12199,HDZ11NP,950323,SJPLTEM: Add HFS keywork             @01A#
# $P1=KA00045,HDZ11SP,960111,SJPLTEM: Updates for DFSMS 1.3       @P1A#
# $P2=KA00107,HDZ11SM,960415,SJPLTEM: Remove MODEL attribute      @P2A#
# $L59=NFS,HDZ11TS,970226,SJPLBPF: File Ext. Support             @L59A#
# $P3=KAB0033,HDZ11TS,970701,SJPLPKL: Add # comment char after    @P3A#
#                                        xlat keyword             @P3A#
# $P4=KAB0114,HDZ11TS,971030,SJPLTEM: Chg DFSMS/MVS->OS/390       @P4A#
# $L53=NFS,HDZ11TS,971031,SJPLBPF: WebNFS Support                @L53A#
# $P5=KAB0379,HDZ11TS,980512,SJPLBPF: Default changes             @P5A#
# $L5D=NFS,HDZ11TS,980821,SJPLBPF: NC Support            OW34846=@L5DA#
# $L5X=NFS,HDZ11TS,980820,SJPLTEM: Filename delimiter    OW34846=@L5XA#
# $LA1=OW38745,HDZ11TS,981209,SJPLRMS: nfstasks(n,m,o)           @LA1A#
# $02=OW40268,HDZ11TS,990727,SJPLRMS: Comment out nfstasks flag   @02A#
# $03=OW42036,HDZ11TS,991213,SJPLBPF: New readdirtimeout keyword  @03A#
# $04=OW43829,HDZ11TS,000410,SJPLBPF: Lower readdirtimeout limit  @04A#
# $LA7=OW46949,HDZ11TS,000921,IBSKEK: TEXT/BINARY on a single    @LA7A#
#                                     mount point                @LA7A#
# $P6=KAD0016,HDZ11TS,001106,SJPLBPF: Allow up to 100 hfs tasks   @P6A#
# $05=OW48939,HDZ11TS,010415,SJPLJST: rddr cookie verifier        @05A#
# $LC1=OW49104,HDZ11TS,010115,IBSNIV: File Tagging Support       @LC1A#
# $06=OW51358,HDZ11TS,010921,IBSPIV : New hfsfbtimeout keyword    @06A#
# $07=OW54351,HDZ11TS,020422,IBSKVV : New upcase and              @07A#
#                                     mixcase keywords            @07A#
# $08=OW55830,HDZ11TS,020819,IBSKVV :SMF activate at NFSS startup @08A#
# $L6=NFS,HDZ11US,030405,SJPLMB:      Changed OS/390 to Z/OS      @L6A#
# $L66=NFS,HDZ11US,030303,IBSMVB: NFSS 878 abend handling        @L66A#
# $LCE=OW55734,HDZ11TS,020701,IBSNIV: File Tagging Support       @LCEA#
# $L74=NFS,HDZ11VS,031015,SJPLJST: NFS ver 4 protocol support    @L74A#
# $09=OA03523,HDZ11TS,030515,IBSVKR: Extend RETRIEVE attr to HFS  @09A#
# $0A=OA05684,HDZ11TS,031208,IBSNIV: New remount/noremount keyword@0AA#
# $L76=NFS,HDZ11VS,040119,IBSDYP: NFS Server DHCP Support        @L76A#
# $L74=NFS,HDZ11VS,040322,SJPLSLH: Added MVSMNT PRocessing Attr @L74BA#
# $L74=NFS,HDZ11VS,040805,SJPLMB:  Added DENYRW/NODENYRW Attrs @L74M5A#
# $L7E=NFS,HDZ11VS,041214,SJPLMB:                                @L7EA#
#          Legal Rqmt: Change "OPEN EDITION" to "z/OS UNIX"      @L7EA#
```

Figure 44. NFS system server sample attribute table (Part 1 of 21)

```
# $P07=KAJ0262,HDZ11VS,050204,SJPLMB:                            @P07A#
#          1. Add missing comment delimiters                     @P07A#
#          2. Move MVSMNT to Processing Attributes Section       @P07A#
#          3. Remove nohfs option.  It is no longer supported.   @P07A#
# $0B=OA08867,HDZ11US,040929,IBSKYL: Correct typo for @L66A       @0BA#
# $P08=KAJ0243,HDZ11VS,050331,SJPLSCA: 64bit fileid support      @P08A#
# $0C=OA12850,HDZ118N,050811,SJPLSLH: MVSMNT and SAF text added   @0CA#
# $0D=OA14044,HDZ118N,051208,IBSNIV: Correct nfstasks(n, ,o)      @0DA#
# $L74=NFS,HDZ11VS,041020,SJPLKU:                               @L74AA#
#             Added mvssec(),hfssec() and pubsec().             @L74AA#
# $L81=NFS,HDZ118N,050908,IBSDYP: Exports file netgroup support  @L81A#
# $L82=NFS,HDZ118N,050808,SJPLJFA: Stringprep Support            @L82A#
# $0E=OA12994,HDZ118N,060106,SJPLRAS:                             @0EA#
#             Added nordrcache                                    @0EA#
# $P09=K9N0086,HDZ119N,060807,SJPLMB:                            @P09A#
#          1. Added NFS v4 attr/read/write timeout vs lease      @P09A#
#             time restriction note.                             @P09A#
# $P0A=K9N0099,HDZ119N,060823,SJPLMB:                            @P0AA#
#          1. Changed nfstasks note wording slightly.            @P0AA#
# $P0B=K9N0317,HDZ1AN0,070315,SJPLRAS:                           @P0BA#
#          1. Changed comment for SETOWNERNOBODY attribute.      @P0BA#
# $0F=OA20232,HDZ1AN0,070321,IBSNIV:                              @0FA#
#          1. New second parameter of writetimeout attribute added@0FA#
# $LAG=NFS,HDZ119N,070410,IBSDYP:                                @LAGA#
#          1. Added NFSv4domain attribute for NFSv4 name mapping @LAGA#
# $P0C=KN00022,HDZ1AN0,070615,IBSDYP:                            @P0CA#
#          1. Changed comment for NFSV4DOMAIN attribute.         @P0CA#
# $P0D=KN00025,HDZ1AN0,070704,SJPLJST: # TCP/UDP tasks spec      @P0DA#
# $LAH=NFS,HDZ1AN0,070831,SJPLMB:                                @LAHA#
#          1. Added ZNFSClient attribute for z/OS NFS Client     @LAHA#
# $P0E=KN00144,HDZ1AN0,071024,IBSDYP: multi-domain support for   @P0EA#
#                                     NFSv4 name mapping         @P0EA#
#-------------------- Release V1r11 -----------------------------@LCQC#
# $LBB=NFS,HDZ1B1N,080404,SJPLMB:                                @LBBA#
#          1. Added MVSPREFIX and IMPPREFIX site attributes for  @LBBA#
#             the extended path prefix support.                  @LBBA#
# $LBK=__NFS__,HDZ1B1N,080414,IBSVAE:                            @LBKA#
#          1. Delay Detection                                    @LBKA#
# $P0F=KBN0052,HDZ1B1N,080814,IBSVAE:                            @P0FA#
#          1. Correct wording in description of DlyDTimeout      @P0FA#
# $LBC=__NFS__,HDZ1B1N,080818,SJPLJST:                           @LBCA#
#     1. Add DELEGATION/NODELEGATION Site Attributes             @LBCA#
# $010=OA24375,HDZ1B1N,080605,IBSVKR:                            @010A#
#          1. h3_rddr enhancement                                @010A#
#-------------------- Release V1r12 -----------------------------@LCQA#
# $LCQ=__NFS__,HDZ1C1N,090420,IBSAYG:                            @LCQA#
#          1. Cache monitoring & reporting - Bufhigh attribute   @LCQA#
#             now contains 2 values.                             @LCQA#
# $LCO=__NFS__,HDZ1C1N,090407,IBSVKR :                           @LCOA#
#     1. Add new operands in the site attribute: SMF             @LCOA#
# $011=OA29577,HDZ1C1N,090720,SJPLMM:                            @011A#
#      1.  Change MAXWINDOW from 256 to 2048 so zNFSS can        @011A#
#          cache up to 64MB out_of_sequence 32KB_rpc_write.      @011A#
# $012=OA27743,HDZ1C1N,090522,IBSVKR:                            @012A#
#          1. New site attribute for GID inheritance             @012A#
# $013=OA24594,HDZ1C1N,080521,IBSNIV:                            @013A#
#      1. The overflow attribute support.                        @013A#
# $014=OA31051,HDZ1C1N,091111,IBSNIV:                            @014A#
#      1. AE for OA24594: Add pound sign to fiche flag.          @014A#
# $015=OA30333,HDZ1C1N,091221,IBSSAN:                            @015A#
#      1. Changed comment for NFSV4DOMAIN attribute.             @015A#
# $P0G=KCN0160,HDZ1C1N,100325,SJPLRAS:                           @P0GA#
```

Figure 44. NFS system server sample attribute table (Part 2 of 21)

```
#      1. bufhigh - restore Server may adjust initial value on   @P0GA#
#         startup. And change default to 32M from 64M            @P0GA#
# $P0H=KCN0164,HDZ1C1N,100402,SJPLRAS:                           @P0HA#
#      1. bufhigh - restore comment delimiter in front of fiche  @P0HA#
# $P0I=KCN0145,HDZ1C1N,100407,SJPLRAS:                           @P0IA#
#      -  General Comment Cleanup                                @P0IA#
#      1. Comment update: nfstasks attribute                     @P0IA#
#      2. Comment update: mintasks attribute                     @P0IA#
#      3. Fix Prolog comment                                     @P0IA#
#      4. Comment update: MVSMNT                                 @P0IA#
#      5. Comment update: Added commented out sidefile(dsname)   @P0IA#
#      6. Convert lowercase BUFHIGH to bufhigh                   @P0IA#
#      7. Comment update: RECFM - change "S" to "M"              @P0IA#
# $016=OA31545,HDZ1C1N,100311,IBSNIV:                            @016A#
#      1. The overflow attribute support for V4.                 @016A#
#          (included in R1.12 roll-up APAR OA32835)              @016A#
#                                                                     #
#######################################################################
#
# This is a prototype site defaults attribute file for the
#  Z/OS Network File System Server Sample Attribute Table         @L6C#
#
# '#' character starts a comment.  Comments can appear anywhere.
# White space is ignored when parsing the file.

# Default values are illustrated in the examples in this file

#
# Keywords are not case sensitive. 'BLKS' is the same as 'blks' is
# the same as 'BlKs'.

# All time values are in seconds.

#######################################################################
# The following are known as data set creation attributes.            #
#######################################################################

# SPACE specifies the amount of primary and secondary space allocated
# for a new data set.  The syntax is:
#
#       SPACE(PRIMARY,SECONDARY)
#
# The secondary field is optional (if omitted, the default is taken).
#
# Dimension of allocation is BLKS, TRKS, or CYLS
# RECS is a synonym for BLKS.

space(100,10), blks

# RLSE specifies that unused space should be released from the data
# set the first time that a new data set is closed.  For slow clients,
# with long pauses between writes, the RLSE attribute will cause space
# to be released from the primary extent prematurely.  Subsequent
# writes will cause secondary space to be allocated.

norlse

# The record format, or RECFM, defines part of the layout of a data
# set: how the records are physically layed out on disk.
#
```

Figure 44. NFS system server sample attribute table (Part 3 of 21)

```
# Valid RECFM characters are:
#
#       V  - Variable Length Records (LRECL defines maximum size of
#              any record)
#       F  - Fixed Length Records (LRECL defines the actual length of
#              all records)
#       U  - Undefined Length Records
#
# modified by:
#
#       B  - Records are Blocked (BLKSIZE defines the size of the block)
#       S  - Spanned for variable length records
#            Standard format for fixed length records
#       M  - Machine Control Codes
#       A  - ANSI Control Codes
#
# "A" and "M" are mutually exclusive                               @P0I
# "V", "F", and "U" are mutually exclusive
# "S" is not allowed for DSNTYPE(PDS) and DSNTYPE(LIBRARY)
# (refer to DSNTYPE later in this section.)
#
# The BLKSIZE is the size, in bytes, of a physical block on disk.
# BLKSIZE(0) allows the system to choose an optimized block size.
#
# LRECL defines the size, in bytes, of a logical record in the data set.

recfm(vb), blksize(0), lrecl(8196)

# The data set organization can be one of:
#
#       PS              - Physical Sequential
#       DA              - Direct Access
#       INDEXED         - VSAM KSDS data set
#       NONINDEXED      - VSAM ESDS data set
#       NUMBERED        - VSAM RRDS data set
#
# PS is a good organization for NFS usage, and NONINDEXED is the
# corresponding good VSAM data set for NFS (e.g. with AIX client in
# BINARY mode) usage.

dsorg(ps)

# DSNTYPE specifies whether a PDS or a PDSE is to be created when
# the make directory workstation command is issued.
#
# Valid DSNTYPEs are:
#
#       PDS     - Create a Partitioned Data Set.
#       LIBRARY - Create a Partitioned Data Set Extended.

dsntype(pds)

# Number of Directory blocks for PDS allocation

dir(27)

# The MGMTCLAS specifies the management class associated with the
# file creation.
#
# The syntax is:
#
#   mgmtclas(mgmt_class_name)
```

Figure 44. NFS system server sample attribute table (Part 4 of 21)

```
# The VOLUME (or VOL) attribute enables you to specify the volume
# on which to create the specified data set.
#
# The syntax is:
#
#    volume(volser)

# The UNIT attribute enables you to specify the unit on which to
# create the specified data set.
#
# The syntax is:
#
#    unit(unit_name)

# The following attributes are used to control VSAM data set
# creation. They are used only if the DSORG parameter defines
# the data set to be type INDEXED, NONINDEXED or NUMBERED.
#
# Refer to appropriate IBM MVS documentation for a more
# complete description of these and other data set creation
# attributes.

# The KEYS(LENGTH,OFFSET) attribute enables you to define the key
# length and offset for a VSAM INDEXED (KSDS) data set. It is used
# only if DSORG is INDEXED.
#
# Valid range for LENGTH is from 1 to 255.
# Valid range for OFFSET is from 0 to 32760.

keys(64,0)

# The RECORDSIZE(AVERAGE,MAXIMUM) attribute enables a user to define
# the average and maximum record sizes for a VSAM data set.
# These two values must be equal for NUMBERED (RRDS) data sets.
#
# Valid range is from 1 to 32760.

recordsize(512,4K)

# The SPANNED and NONSPANNED attributes define whether VSAM
# records will span control intervals. This option does not affect
# non-VSAM variable length record data sets. Use the 'S' option
# with the RECFM attribute for non-VSAM data sets.

nonspanned

# The SHAREOPTIONS attribute defines the cross region and cross
# system file sharing allowed for a VSAM data set.
#
# Valid range for each argument is from 1 to 4.

shareoptions(1,3)

########################################################################
# The following are known as processing attributes.                    #
########################################################################

# There are three timeout types: attributes, reads and writes.
# The various timeout values are used by the system to determine
# when to close and deallocate an inactive data set after the last
# "attribute", "read", or "write" operation.
# The WRITETIMEOUT is usually kept short, because WRITE
```

Figure 44. NFS system server sample attribute table (Part 5 of 21)

```
# operations result in exclusive locking, and you'll want to release
# the data set.  For slow clients, with long pauses between writes,
# you'll want to increase the WRITETIMEOUT value.
#
# Valid range is from "m" to "n"; where "m" is the argument of
# MINTIMEOUT(m) and "n" is the argument of MAXTIMEOUT(n), unless
# NOMAXTIMEOUT is specified.  In that case, "n" is 32767.
#
# xxxxTIMEOUT(n) indicates to deallocate the data set n seconds after
# the "xxxx" operation;
# NOxxxxTIMEOUT indicates not to deallocate the data set after
# the "xxxx" operation;
# Where "xxxx" can be "ATTR", "READ", or "WRITE".
#
# e.g. WRITETIMEOUT(1) indicates to deallocate the data set 1 second
#      after a write.
#      NOWRITETIMEOUT indicates not to deallocate the data set after
#      a write.
#      READTIMEOUT(90) indicates to deallocate the data set 90 seconds
#      after a read if no further activity against it.
#
# NOTE:  When using the NFS version 4 protocol, these timeout     @P09A
#        values should be set to a value less than or equal to    @P09A
#        the lease time.  Otherwise, it is possible for           @P09A
#        performance problems to occur when attempting to access  @P09A
#        MVS data sets.                                           @P09A
# Also, WRITETIMEOUT(n,o) for a data set can be specified.        @0FA
# In this case, "o" indicates the number of seconds that NFS      @0FA
# Server will wait for data to arrive to complete a partial       @0FA
# record before closing the data set.                             @0FA
# Valid range for "o" is from "n" to "255 * n".                   @0FA
# The default value for "o" is "4 * n", if not specified.         @0FA
# It is preferable to set it as a multiple of "n", otherwise it   @0FA
# will be rounded down to the nearest multiple of "n".            @0FA

attrtimeout(120), readtimeout(90), writetimeout(30,120)          #@0FC

# Processing may be TEXT or BINARY
#
# BINARY is good for using MVS as a disk farm for PCs and AIX machines
# and offers better performance.
#
# TEXT should be specified if it is necessary to share data sets
# containing textual data with other MVS applications.

binary

#                                                                 @LA7A
# MAPPED should be specified when a mixed set of data types are   @LA7A
# to be processed on a single mount point.  The determination     @LA7A
# of whether the data is to be processed as text or binary        @LA7A
# depends on the rules established in the specified side file.    @LA7A
# If a file extension cannot be mapped to text or binary, then    @LA7A
# the data will be processed according to what has been specified @LA7A
# as binary or text at the mount level, and finally, the site     @LA7A
# level.  If binary or text is specified at the file level,       @LA7A
# the specification overrides the MAPPED specification.           @LA7A
#                                                                 @LA7A
# The syntax is:                                                  @LA7A
#                                                                 @LA7A
# mapped                                                         #@LA7A
#                                                                 @LA7A
```

Figure 44. NFS system server sample attribute table (Part 6 of 21)

```
# The end of line terminators are:
#
#       CR, CRLF, LF, LFCR, or NOEOL.
#
# They define the conversion of records to end-of-line terminators in TEXT
# mode.
# LF should be used for AIX clients;
# CRLF should be used for PC clients.
# (set through client mounts appropriately).  When TEXT mode is
# specified, LF is the default.

LF

# BLANKSTRIP or NOBLANKSTRIP affects the processing of trailing
# blanks when reading and writing to Fixed Record data sets with
# text processing enabled.

blankstrip

# The MAPLEADDOT attribute turns on mapping of a file name starting
# with a leading "." from a client to a legal leading "$" for a
# MVS data set name.  NOMAPLEADDOT turns off this mapping.

mapleaddot

# The MAPLOWER attribute tells the server to map file names to
# uppercase when received from the client in an NFS request and
# to translate from uppercase to lowercase when returned to the client.
# Keywords are not case sensitive and are unaffected by this option.
#
# The NOMAPLOWER attribute tells the server NOT to do any translation.
# i.e. the server is neither to map uppercase when received from
# the client nor to translate to lowercase when returned to the client.
# All the entries in the EXPORTS file are case sensitive.
# The client MOUNT request must specify the MVS qualifier with
# the correct case to successfully match the EXPORTS file entry.

maplower

# RETRIEVE tells the server to recall a migrated data set on read/write
# access. NORETRIEVE will force the return of "Device not available"
# error for migrated files.
# RETRIEVE can be coded in the following ways:
#  RETRIEVE         - nowait for recall.                           @09C
#  RETRIEVE(WAIT)   - wait for recall.
#  RETRIEVE(NOWAIT) - nowait for recall.
# For a quiesced HFS file system:                                  @09A
#  RETRIEVE(WAIT) will suspend a request until the file system     @09A
#                 will be unquiesced.                              @09A
#  Others value will force the return of IO error.                 @09A
# RETRIEVE is the default

retrieve

# The FASTFILESIZE attribute tells the server to calculate approximate
# file sizes from available catalog information and disk geometries.
# This approximate size may cause problems with client applications
# since the size is probably inaccurate. The NOFASTFILESIZE may
# result in decreased performance because the server may read a data
# set, applying the defined processing attributes, to determine the
# exact size of the data set as viewed by an NFS client.

nofastfilesize
```

Figure 44. NFS system server sample attribute table (Part 7 of 21)

```
# The SETOWNERROOT keyword tells the server to set the user ID in the
# attributes returned to a client for a specified file to 'root' when
# the client is logged on as superuser.  SETOWNERNOBODY tells the
# server to set the user ID in the attributes to 'nobody' (65534).@P0BC

setownerroot

# You can have the execute bit for plain files on or off by mount
# point. Turn this option on if you plan to store executables
# or shell scripts on the MVS system on a mount by mount basis.
# It should probably always be off in the site file.
#
# EXECUTEBITON will turn on the execute bits (user, group
# and other) for a mount point's files.

executebitoff

# If the installation intends to customize the translation table, @L3XA
# a new DD card, NFSXLAT, is required in the NFSS startup proc.   @L3XA
#                                                                 @L3XA
# //NFSXLAT       DD      DSN=dataset_name,DSP=SHR                @L3XA
#                                                                 @L3XA
# Where dataset_name is the name of PDS or PDSE whose             @L3XA
# members are the translation tables.                             @L3XA
#                                                                 @L3XA
# The XLAT(member) keyword tells the server which member          @L3XA
# the server is to use as the installation default translation    @L3XA
# table. 'member' is the name of a translation table which        @L3XA
# resides in a PDS or PDSE dataset.                               @L3XA
#                                                                 @L3XA
# The syntax is:                                                  @L3XA
#                                                                 @L3XA
# xlat(member)                                                    #@P3C

# FILEEXTMAP or NOFILEEXTMAP affects the file extension mapping   @L59A
# capability.  FILEEXTMAP turns on file extension mapping and     @L59A
# NOFILEEXTMAP turns it off.  This option can be specified at     @L59A
# the file command level.  The default is NOFILEEXTMAP.           @L59A

nofileextmap                                                     #@L59A

# SIDEFILE(dsname) specifies the name of the data set that        @L59A
# contains the rules for file extension mapping purposes.         @L59A
# If a side file name is specified in the attributes data set     @L59A
# then it is the default side file for this NFS server.           @L59A
# A user can also specify another side file name during a MOUNT   @L59A
# operation to be used along with the default.  The mapping rules @L59A
# will first be searched in the side file specified during MOUNT  @L59A
# and then in the default.  To allow file extension mapping a     @L59A
# side file name must be specified either as a default or in the  @L59A
# MOUNT command.  dsname is a fully-qualified MVS data set name   @L59A
# without quotation marks.  SIDEFILE is only specifiable at the   @L59A
# MOUNT level.  See GFSAPMAP for sample mapping side file and     @L59A
# syntax.                                                         @L59A

# sidefile(dsname)                                               #@P0IA

#
# TAG specifies the newly created files should receive a file tag.@LCEA
# NOTAG specifies the newly created file should be untagged.      @LCEA
# The default is NOTAG.                                           @LCEA

  notag                                                          #@LCEA
```

Figure 44. NFS system server sample attribute table (Part 8 of 21)

```
# CLN_CCSID(n) specifies the Coded Character Set Identifier(CCSID)@LC1A
# for the remote mounted file system (NFS client) when text is    @LC1A
# being translated.                                               @LC1A
# The default (if specified) is 819 (ISO 8859-1 ASCII).           @LC1A

  cln_ccsid(819)                                                 #@LCEC

# SRV_CCSID(n) specifies the Coded Character Set Identifier(CCSID)@LC1A
# for the local file system (Z/OS NFS Server) when                @L6C
# a new file is being created.                                    @LC1A
# The default (if specified) is 1047 (Latin Open System EBCDIC).  @LC1A

  srv_ccsid(1047)                                                #@LCEC

# CONVSERV(technique) - 'technique' specifies technique-search-order
# which Unicode Service will use for specified srv_ccsid(n) and   @LCEA
# cln_ccsid(n) code pages.                                        @LCEA
# 'technique' consists of up 5 technique-characters corresponding @LCEA
# to the 5 available techniques: R, E, C, L and M.                @LCEA
# The default is CONVSERV(LRE).                                   @LCEA

  convserv(LRE)                                                  #@LCEA

# MVSMNT specifies that in NFS version 4, a lookup will be        @P07M
# treated as if a mount procedure were given for the LOOKUP       @P07M
# result object.                                                  @P07M
#                                                                 @P07M
# For any LOOKUPs that do not specify MVSMNT, any Processing      @P07M
# Attributess that may have been provided will be merged          @P07M
# with any that were in effect for the LOOKUP parent directory.   @P07M
#                                                                 @P07M
# For LOOKUPs that *DO* specify MVSMNT, any other Processing      @P07M
# Attributes provided will be merged with the site defaults.      @P07M
# MVSMNT cannot be specified for any LOOKUP where the parent      @P07M
# directory was navigated to by an mount procedure or a result    @P07M
# of an object that was already LOOKUP'ed with MVSMNT.  This      @P07M
# is to ensure that only a client system mount specifies MVSMNT.  @P07M
# With MVSMNT specified, z/OS NFS server site security attributes  @0CA
# saf and safexp are used to control access to z/OS UNIX           @0CA
# file systems.                                                    @0CA
#                                                                 @P07M
# There is no default.  MVSMNT should not be specified in the     @P0IA
# attributes data set.  The mvsmnt attribute should only be       @P0IA
# specified on NFS Version 4 mount commands.                      @P0IA
#                                                                 @P0IA

#                                                                  @0EA
# NORDRCACHE specifies that the server should not                  @0EA
# stale the Legacy (MVS conventional data) directory cache if an   @0EA
# addition is made to the directory.  This causes the next         @0EA
# READDIR operation to access the directory                        @0EA
# information from the Physical File System (PFS) rather than the  @0EA
# directory cache.                                                 @0EA
#                                                                  @0EA
# The NORDRCACHE attribute does not apply to z/OS UNIX files.      @0EA
#                                                                  @0EA
# NOTE: When NORDRCACHE is not specified, the addition of an entry @0EA
# to the LEGACY directory cache will not be visible to client      @0EA
# until the next readdir cache timeout or a remove from that       @0EA
# directory.                                                       @0EA
# When NORDRCACHE is specified, the addition will be visible to    @0EA
# the client by the subsequent READDIR whether the readdir         @0EA
```

Figure 44. NFS system server sample attribute table (Part 9 of 21)

```
# cache timeout has expired or not. This may impact                 @0EA
# performance because the directory list must be read from the      @0EA
# Physical File System after any addition to the directory.         @0EA
#                                                                   @OEA
# The syntax is:                                                    @0EA
#                                                                   @OEA
# NORDRCACHE                                                       #@OEA

# The RDRVERF attribute tells the server to do cookie verifier      @0EA
# checking for NFS version 3 readdir and readdirplus requests,      @0EA
# and for NFS version 4 readdir requests.                           @0EA
# NOTE: RDRVERF provides consistency in the listing of a           @010A
# directory's content and as a result may impact performance.      @010A
#                                                                   @0EA
# The NORDRVERF attribute tells the server not to do cookie         @0EA
# verifier checking for NFS version 3 readdir and readdirplus       @0EA
# requests, and for NFS version 4 readdir requests.                 @0EA
# NOTE: NORDRVERF does not provide consistency in the listing of a@010A
# directory's content and may cause duplicate or omit entries      @010A
# when the directory is changing during the listing.               @010A
# The default is NORDRVERF.                                         @0EA

nordrverf                                                          #@0EA

# ZNFSCLIENT specifies that the NFS Client is z/OS.                @LAHA
#                                                                  @LAHA
# This value is used for compatibility purposes only.  For         @LAHA
# example, z/OS Unix System Services has separate File and         @LAHA
# Directory Default Access Control Lists (ACLs).  Other platforms @LAHA
# a common Default ACL which is used for both.  This flag          @LAHA
# allows the z/OS NFS Server to customize its response to the      @LAHA
# NFS Client accordingly.                                          @LAHA
#                                                                  @LAHA
# NOTE: This value should never be specified by the customer.      @LAHA
# this attribute is automatically appended to the procssing        @LAHA
# attribute string by the z/OS NFS Client when it detects that     @LAHA
# it is sending the mount request (or the last LOOKUP for an       @LAHA
# NFS version 4 mount) to a z/OS NFS Server.                       @LAHA
# If this attribute is specified on a non-z/OS Client Mount        @LAHA
# request, some requests may not function properly.                @LAHA
#                                                                  @LAHA
# ZNFSCLIENT  DO NOT SPECIFY!  The attribute is listed here for    @LAHA
#                              documentation purposes only.        @LAHA


########################################################################
# The following are known as site attributes.                          #
########################################################################

# The following are attributes specifiable ONLY in the site
# file (this file).
#
# Some of these values control internal structures and processing
# within the NFS server. Tuning of these values to improve performance
# should be done incrementally and tested.

# SECURITY attribute control the level of security checking.
# The format of the security keyword is security(mvs,hfs,public)  @L5DA
# where:                                                           @L5DA
# mvs - security option for mvs data access                        @L5DA
# hfs - security option for HFS data access                        @L5DA
# public - security option for data access with the public         @L5DA
#          filehandle                                              @L5DA
```

Figure 44. NFS system server sample attribute table (Part 10 of 21)

```
 # The first positional parameter is required and the other two    @L5DA
# are optional.  When the optional parameters are not specified   @L5DA
# they are assigned the same security as the first parameter.     @L5DA
# Four options can be chosen from.  They are:
#  NONE    - No security checking is performed.
#  SAF     - SAF checking is performed in line.
#  EXPORTS - EXPORTS file is used to check security.
#  SAFEXP  - Both SAF and EXPORTS file checks are performed.
#
#
# Defaults are SAFEXP for all data accesses.                      @L5DA

security(safexp,safexp,safexp)                                  # @L5DC

# PCNFSD tells the server to start PCNFS server.
# NOPCNFSD tells the server not to start PCNFS server.
# If not specified PCNFSD, default is NOPCNFSD.

nopcnfsd

# LEADSWITCH tells the server to return '/' as the first character
#            in each export entry.
# NOLEADSWITCH tells the server not to return '/' as the first character
#            in each export entry.
# If not specified NOLEADSWITCH, default is LEADSWITCH.

leadswitch

# MINTIMEOUT and MAXTIMEOUT set allowable values for
# ATTRTIMEOUT, READTIMEOUT, and WRITETIMEOUT. Specify NOMAXTIMEOUT
# to allow NOATTRTIMEOUT, NOREADTIMEOUT and NOWRITETIMEOUT specification
# by clients.
#
# Valid range is from 1 to 32767.

mintimeout(1)
nomaxtimeout

# The logout time is the time limit on inactivity for a given user
# on a machine. When the limit is reached, the user is automatically
# logged out. The user must then do another "mvslogin" to restart
# the session. The time value is specified in seconds.
#
# You would probably want to set LOGOUT to the TSO timeout value that
# is defined at your site.
#
# MAXTIMEOUT does not affect the LOGOUT site attribute.
#

logout(1800)                    # 30 minutes (30 * 60)

# The readdirtimeout is a new timeout attribute to control the   @03A
# timeout of the readdir cache used by MVS conventional data     @03A
# sets.  The timeout value controls how long before the readdir  @03A
# results saved in cache are discarded.                          @03A
#                                                                @03A
# Valid range is from 1 to 32,767 seconds.                       @04C
#                                                                @03A
# n can go as low as 1 second  but to avoid the possibility of   @04C
# client hanging (because of network delays and staled cache),   @04C
# n is not recommended to be lower than 5 seconds.               @04A
# n may need to be increased if the network is slow and the      @03A
# accessed directory has a lot of entries.                       @03A
```

Figure 44. NFS system server sample attribute table (Part 11 of 21)

```
# The default readdirtimeout is 30 seconds.                      @03A

readdirtimeout(30)                                           #   @03A

# The NFSTASKS(n,m,o,t,u)                                         @P0DC
#   defines the number of NFS tasks (or threads) to spawn.        @P0DC
#                                                                 @LA1A
# If NFSTASKS(n,m) is specified, then the following is true:      @LA1A
#                                                                 @LA1A
# 'n' is the number of subtasks which handle the asynchronous I/O @L3BC
# operations or short blocking operations (the maximum number of  @LA1C
# concurrent NFSS requests) in the z/OS MVS data path.            @P0IC
# 'm' is the number of subtasks which handle the long blocking    @L3BC
# operations (the maximum number of concurrent NFSS recall and    @L3BC
# HFS requests.). Increase this value if your server supports     @L3BC
# many active recall operations or HFS clients.                   @P0IC
#
# Valid range for 'n' is from 4 to 99.                            @P0IC
# Valid range for 'm' is from 4 to 100.                           @P0IC
# The sum of 'n' plus 'm' must be less than or equal to 100.       @0DC
#                                                                  @0DA
# NOTE: Based on system resources available below the 16Mb line,   @0DA
#       the maximum 'n' value may not be achievable.  The          @0DA
#       precise maximum value will be system configuration         @0DA
#       dependent.  If an 878 or 80A Abend is experienced during  @P0AC
#       NFSS startup, use a smaller value for 'n'.                @P0AC
#
# If NFSTASKS(n,m,o) or NFSTASKS(n,m,o,t,u) is specified,         @P0DC
# then the following is true:                                     @P0DA
#                                                                 @LA1A
# 'n' is the number of subtasks which handle the asynchronous I/O @LA1A
# operations or short blocking operations (the maximum number of  @LA1A
# concurrent NFSS requests) in the MVS data path.                 @P0IC
# 'm' is the number of subtasks which handle HFS requests.        @LA1A
# Increase this value if your server supports lots of active      @LA1A
# HFS clients.                                                    @LA1A
# 'o' is the number of subtasks which handle the long blocking    @LA1A
# operations (the maximum number of concurrent NFSS recall        @LA1A
# requests) in the z/OS MVS data path.                            @P0GC
# Increase this value if your server supports many active recall  @P0IC
# operations.                                                     @P0IC
# 't' is the number of transport subtasks which handle TCP        @P0DA
#     network requests.                                           @P0DA
# 'u' is the number of transport subtasks which handle UDP        @P0DA
#     network requests.                                           @P0DA
#
# Valid range for 'n' is from 4 to 99.                            @P0IC
# Valid range for 'm' is from 4 to 100.                           @P0IC
# Valid range for 'o' is from 1 to 99.                             @0DC
# Valid range for 't' is from 4 to 32.                            @P0DA
# Valid range for 'u' is from 4 to 32.                            @P0DA
# The sum of 'n' plus 'o' must be less than or equal to 100.       @0DC
#                                                                  @0DA
# NOTE: Based on system resources available below the 16Mb line,   @0DA
#       the maximum 'n' + 'o' value may not be achievable.  The    @0DA
#       precise maximum value will be system configuration         @0DA
#       dependent.  If an 878 or 80A Abend is experienced during  @P0AC
#       NFSS startup, use a smaller value for 'n' + 'o'.          @P0AC

nfstasks(8,16,8,4,4)                                            # @P0DC

#                                                                 @L66A
# The MINTASKS(n,m,o) defines the minimum number of NFS tasks     @L66A
```

Figure 44. NFS system server sample attribute table (Part 12 of 21)

```
# or threads) allowed to run. Tasks may be terminated for reasons @L66A
# such as 878 or 80A abends.                                      @L66A
#
# Note: Options t and u in NFSTASKS(n,m,o,t,u) do not have        @P0DA
#       MINTASKS counterparts because MINTASKS only applies to    @P0DA
#       the request processing tasks, not the transport tasks.    @P0DA
#
# 'n' is the minimum number of subtasks which handle              @L66A
#     the asynchronous I/O operations or short blocking           @L66A
#     operations. If the number of active 'short' tasks becomes   @L66A
#     less than 'n' the shutdown process of NFS server will start.@L66A
# 'm' is the minimum number of subtasks which handle HFS          @L66A
#     requests. If the number of active HFS tasks becomes less    @L66A
#     than 'm' the shutdown process of NFS server will start.     @L66A
# 'o' is the minimum number of subtasks which handle long         @L66A
#     blocking operations. If the number of active legacy long    @L66A
#     service tasks becomes less than 'o' the shutdown process    @L66A
#     of NFS server will start.                                   @L66A
# If 'n','m' or 'o' are greater than the corresponding values     @L66A
# in NFSTASKS, they are assigned to half of the NFSTASKS value.   @P0IC
# If 'n','m' or 'o' are not specified they are assigned default   @P0IC
# values of 4,4,1 respectively.                                   @P0IC
# Valid range for 'n' is from 4 to 99.                            @P0IC
# Valid range for 'm' is from 4 to 100.                           @P0IC
# Valid range for 'o' is from 1 to 99.                            @0DC
# The sum of 'n' plus 'o' must be less than or equal to 100.      @0DC
#                                                                 @0DA
# NOTE: Based on system resources available below the 16Mb line,  @0DA
#       the maximum 'n' + 'o' value may not be achievable.  The   @0DA
#       precise maximum value will be system configuration        @0DA
#       dependent.  If an 878 or 80A Abend is experienced during  @P0AC
#       NFSS startup, use a smaller value for 'n' + 'o'.          @P0AC

mintasks(4,8,4)                                                  #@L66A
                                                                 #@P07C

# The RESTIMEOUT(n,m) defines resource cleanup timer.             @L3LA
# 'n' is the resource retention period for mounts and
# associated resources which will be removed if the mounts
# have been inactive more than 'n' number of hours.               @L3LA
# 'm' is the time of day to do the cleanup work
# for mounts and associated resources which
# have been inactive more than 'n' number of hours.
# The time of day is specified as a 24 hour local time value
# The starting time is at least 24 hours from NFSS start-up.
#                                                                 @L3LA
# NFSS will appear slow during this cleanup activity.             @L3LA
# Cleanup will be executed under main task, thus preventing       @L3LA
# any additional work from executing.                             @L3LA
# The value of RESTIMEOUT should be set to a value                @L3LA
# such that this cleanup activity will occur while                @L3LA
# NFSS is lightly loaded.                                         @L3LA
#                                                                 @L3LA
# Valid range for 'n' is from 1 to 720.                           @L3LA
# If n is set to 0, NFSS will not remove any mount points.        @L3LA
# Note: NFSS keeps all the information for the                    @L3LA
# inactive mount points, thus creating long chains                @L3LA
# to be searched.                                                 @L3LA
#                                                                 @L3LA
# Valid range for 'm' is from 0 to 23.                            @L3LA

restimeout(48,0)                                                 #@L3LA
```

Figure 44. NFS system server sample attribute table (Part 13 of 21)

```
# The CACHEWINDOW attribute limits the number of windows to be used
# for each data set in caching client block writes which are received
# out of order.
# The size of each window is determined by the packet size.
# The suggested value is some small multiple of
# the number of BIODs running on an NFS client.
#
# Valid range is from 1 to 2048.                                  @011C

cachewindow(112)                                                 # @P5C

# The HFSPREFIX (or HFS) attribute specifies a z/OS UNIX file     @LBBC
# system prefix to be imbedded in the mount directory path name.  @LBBC
# The default value of the z/OS UNIX file system prefix is /hfs.  @LBBC
# Mount requests received by the Network File System beginning    @LBBC
# with this prefix value are identified as mount requests for     @LBBC
# z/OS UNIX.  The prefix value is not part of the path name.      @LBBC
#                                                                 @01A
# Note: The file system must be mounted locally by z/OS UNIX.     @LBBC
#       Otherwise, the client mount fails.                        @LBBC
# hfsprefix(prefix)  or  hfs(prefix)                              @LBBC

hfsprefix(/hfs)                                                  #@LBBC

# The MVSPREFIX attribute specifies an MVS data set prefix        @LBBA
# to be imbedded in the mount directory path name.                @LBBA
# The default value of the MVS data set prefix is /mvs.           @LBBA
# Mount requests received by the Network File System beginning    @LBBA
# with this prefix value are identified as mount requests for     @LBBA
# MVS data sets.  The prefix value is not part of the path name.  @LBBA
#                                                                 @LBBA
# mvsprefix(prefix)                                               @LBBA

mvsprefix(/mvs)                                                  #@LBBA

# The IMPPREFIX attribute specifies how mount directory path      @LBBA
# names which do not have either an HFSPREFIX or an MVSPREFIX     @LBBA
# are to be interpreted.  If no prefix is present in the path     @LBBA
# name, the path is to be interpreted according to the IMPPREFIX  @LBBA
# setting:                                                        @LBBA
#    HFS     - assume it is a z/OS UNIX file system.              @LBBA
#    MVS     - assume it is an MVS data set.                      @LBBA
#    HFS,MVS - try z/OS UNIX file system first.  If that fails    @LBBA
#              or is not found, try MVS data set.                 @LBBA
#    MVS,HFS - try MVS data set first.  If that fails or is not   @LBBA
#              found, try z/OS UNIX file system.                  @LBBA
# The default value is MVS, which matches pre-z/OS V1R11          @LBBA
# functionality.                                                  @LBBA
#                                                                 @LBBA
# Note: For the MVS,HFS setting, the MVS data set must exist      @LBBA
#       if the path is referring to an MVS data set.  Prior to    @LBBA
#       this attribute being added, it was not necessary for the  @LBBA
#       data set to pre-exist.  However, now, if it does not      @LBBA
#       exist, the 2nd option (HFS) will be used.                 @LBBA

impprefix(mvs)                                                   #@LBBA

# fileidsize site attribute:                                      @P08C
#                                                                 @P08C
# The fileidsize attribute tells NFS how to handle the size       @P08C
# of fileids.  Fileids can be recognized either as 32-bit         @P08C
# values or 64-bit values.                                        @P08C
#                                                                 @P08C
```

Figure 44. NFS system server sample attribute table (Part 14 of 21)

```
# Client platforms often copy the returned NFS object fileid to   @P08C
# client internal structure fields such as a UNIX inode.   Older  @P08C
# 32-bit applications and operating systems may not support NFS   @P08C
# fileids larger than 32 bits resulting in value too large errors @P08C
# or unexpected client behaviors.                                 @P08C
#                                                                 @P08C
# IBM recommends upgrading older applications and operating       @P08C
# systems to an environment which supports the latest supported   @P08C
# NFS Protocol Version offering the largest supported fileidsize. @P08C
# If upgrading your environment is not possible, the fileidsize   @P08C
# may be set to a lower value to support older applications if    @P08C
# accessed data can be properly recovered or refreshed.           @P08C
#                                                                 @P08C
# Valid values are 32 or 64.                                      @P08C
#                                                                 @P08C
#     NFS Version 2 maximum value in effect is (32) bits          @P08C
#     NFS Version 3 maximum value in effect is (32) bits          @P08C
#     NFS Version 4 maximum value in effect is (64) bits          @P08C
#                                                                 @P08C
# If NFS Version 2 or Version 3 is in effect, the maximum value   @P08C
# of 32 is used.  When the specified fileidsize value             @P08C
# is larger than the maximum value for the NFS Version in effect, @P08C
# the proper maximum value will be used for generating            @P08C
# the NFS fileid attribute.                                       @P08C
#                                                                 @P08C
# Warning:                                                        @P08C
#                                                                 @P08C
# The fileidsize site attribute should be set to the maximum      @P08C
# value to generate the maximum size in unique ids for NFS        @P08C
# objects within a file system.  Setting the value lower than     @P08C
# the allowed maximum (e.g. NFS Version 4) may result in the      @P08C
# same NFS fileid to be returned to the client for                @P08C
# different server objects.  As a result, reducing the            @P08C
# fileidsize site attribute for client                            @P08C
# compatibility may result in data corruption.                    @P08C
#                                                                 @P08C
# The default is fileidsize(64)                                   @P08C
                                                                 #@P08C
fileidsize(64)                                                   #@P08C

# The LOGICALCHE attribute sets the high water mark for all the
# cache windows combined(in bytes).
#
# Valid range is from 1 to 128MB

logicalcache(16M)                                               # @P5C

# BUFHIGH(XX,YY)                                                  @LCQC
# The BUFHIGH attribute sets the high water mark (in bytes)       @LCQC
# of data buffers before a buffer reclamation takes place         @LCQC
# (it's the first parameter XX) and the watermark in percent of   @LCQC
# XX for printing a data buffer utilization alert message         @LCQC
# (it's the second parameter YY).                                 @LCQC
# A higher number of XX means more caching and probably better    @LCQC
# read performance.                                               @LCQC
# (If XX + LOGICALCACHE is larger than the available storage      @LCQC
#  in the extended private area (implied by the REGION parameter  @LCQC
#  coded in your PROC) at startup, the NFS server will            @LCQC
#  shutdown immediately.)                                         @LCQC
#
# At z/OS Server startup, the actual value (xx) specified with    @P0GA
# bufhigh may be adjusted by the z/OS NFS Server internally       @P0GA
# depending on the available region size and other z/OS NFS       @P0GA
```

Figure 44. NFS system server sample attribute table (Part 15 of 21)

```
# Server memory requirements to enable the z/OS NFS Server to    @P0GA
# execute properly.                                               @P0GA
#
# The valid value of YY is from 50 to 90 percents or 0(zero)      @LCQC
# to turn off the data buffer utilization reporting mechanism.    @LCQC
# Any invalid value of YY is converted to 80 percent.             @LCQC
# The default value of YY is 80 percent.                          @LCQC
# If Bufhigh contains only the first parameter (comma and YY are  @LCQC
# ommited) the default value of YY is used.                       @LCQC

bufhigh(32M,80)                                                #  @P0IC

# On reaching the BUFHIGH high water mark, a percentage of buffers
# is reclaimed for reuse. This is the PERCENTSTEAL. A higher value
# means a reclaim operation is performed less often, but that
# the cached buffers will be significantly trimmed on each reclaim.
# This can result in a poor read performance, because readahead
# buffers may be stolen.
# Lower values result in more frequent reclaim operations, but the
# cached buffers are not significantly trimmed on each reclaim.
#
# Valid range is from 1 to 99.

percentsteal(20)

# The READAHEADMAX value defines the maximum read ahead (in bytes)
# during read processing.
# This reduces the amount of synchronous physical I/O required for
# NFS read requests in sequential processing.  It also reduces context
# switching overhead on NFS read requests by allowing more read
# requests to be satisfied directly from the main task.
#
# The number is usually set to 2 to 4 times the common block size
# used for file access (which is recommended at 8K for AIX file
# activity).
#
# Valid range is from 1 to 128K

readaheadmax(16K)

# The MAXRDFORSZLEFT attribute defines the number of physical block
# buffers to be remained after a read-for-size operation.
# These buffers are remained to satisfy potential subsequent
# NFS read requests against the same file.
# The buffers remained are subject to trimming during buffer
# steal operations.
#
# Valid range is from 1 to 1024.

maxrdforszleft(32)

# SMF(level,switch) attribute controls the level of smf support  #@08A
# and defines whether to start SMF records collection            #@08A
# at the NFSS startup or not.                                    #@08A
# The following 'level' options are available:                   #@LCOC
# NONE - No smf records are to be produced.                      #@08A
# ALL  - All smf records are to be produced.                     #@LCOA
# USERFILE - Both user session and file usage smf records are    #@LCOM
#           to be produced.                                      #@LCOM
# List of levels that are delimited by commas, can be specified. #@LCOA
#        At least one of  the levels: ( USER | FILE | AUDIT )    #@LCOA
#        must be specified for a list, and the remaining levels  #@LCOA
#        are optional. Where:                                    #@LCOA
```

Figure 44. NFS system server sample attribute table (Part 16 of 21)

```
# USER - User session smf records are to be produced.            #@08A
# FILE - File usage smf records are to be produced.              #@08A
# AUDIT- File creation, file removing and file renaming          #@LCOA
#        smf records are to be produced.                         #@LCOA
# Default level is NONE.                                         #@LCOC
# Switch has two options:                                        #@08A
# ON       - activates SMF records collection at the NFSS        #@08A
#            startup                                             #@08A
# OFF      - activation of SMF records collection can be done    #@08A
#            manually by issuing the operator command MODIFY.    #@08A
# Switch is optional parameter. Its default value is OFF.        #@08A
#                                                                #@08A
# The syntax of the SMF attribute is                             #@08A
# SMF(NONE|ALL|USERFILE|subtype_list,ON|OFF)                     #@LCOC
#    where subtype_list is list of 1 - 3 levels:(USER|FILE|AUDIT)#@LCOA
#    separeted by commas.                                        #@LCOA
# For example,                                                   #@08A
# SMF(USER,ON)                                                   #@08A
# SMF(AUDIT,FILE,OFF)                                            #@LCOA
# You can use the short form of the SMF attribute:               #@08A
# SMF(level)                                                     #@08A
# In that case the value of the switch is OFF                    #@08A
smf(none,off)                                                    #@08A

# SFMAX(n) where n specifies the maximum size (in kilobytes) of   @L59A
# allocated storage to hold all of the side files.  n is an       @L59A
# integer from 0 to 2000 (2MB).  The default value is 0 and it    @L59A
# also signifies that no mapping is allowed on this NFS server.   @L59A
# If SFMAX=0, the specification of sidefile in the attributes     @L59A
# data set will cause the server to shut down and the             @L59A
# specification of sidefile in any subsequent MOUNT commands will @L59A
# cause the mount to fail as mapping is not allowed on this NFS   @L59A
# server.  If the amount of storage specified can not be obtained @L59A
# during server initialization then the server will shut down     @L59A
# immediately.                                                    @L59A
#                                                                 @L59A
# The default is SFMAX(0).                                        @L59A

sfmax(0)                                                         #@L59A

# PUBLIC(legacy_path,hfs_path) specifies the legacy path          @L53A
# (MVS conventional data) and/or HFS path that is associated with @L53A
# the public file handle for WebNFS access.  The first path, if   @L53A
# specified, is the legacy path.  The second path is the HFS      @L53A
# path.  MVS and HFS prefix values are optional on these path     @LBBC
# name values since they are positional and thus self-defining.   @LBBA
# However, if a prefix is specified, it must match the active     @LBBA
# prefix value to be recognized as such.  Otherwise, it will      @LBBA
# be assumed to be a qualifier in the path name.  The PUBLIC      @LBBA
# keyword must be specified after the HFSPREFIX (or HFS) and      @LBBM
# MVSPREFIX keywords in this site attribute table.                @LBBM
# If the first path is not there, a comma must precede            @LBBC
# the second path.  If the PUBLIC keyword is specified then one   @L53A
# of the paths must be specified.                                 @LBBC
# A LOOKUP request with the public file handle will determine     @L53A
# which of the two paths it is referring to by the pathname that  @L53A
# it comes in with.  An absolute pathname will tell the server    @L53A
# which of the paths it is referring to by a match on one of the  @L53A
# paths specified.  A LOOKUP request with a relative pathname     @L53A
# will be taken to be an HFS request if HFS is active (i.e. an    @L53A
# hfs_path has been provided); otherwise, it is treated as a      @L53A
# legacy request.                                                 @L53A
```

Figure 44. NFS system server sample attribute table (Part 17 of 21)

```
#                                                                 @L53A
# The default is no PUBLIC paths.                                 @L53A

# CHECKLIST refers to the data set pointed to by the EXPORTS DD   @L81C
# in the startup procedure to be scanned. The data set may contain@L81C
# a list of CHKLIST entries which should match a subsequent       @L81C
# mount point or be the parent of a subsequent mount point to     @L5DA
# allow SAF checking to be bypassed for everything underneath     @L5DA
# that mount point.  CHECKLIST is only valid if SAF checking is   @L5DA
# the security option for the particular data access; otherwise,  @L5DA
# it is ignored even if specified.  See GFSAPEXP in NFSSAMP       @L81C
# library for sample CHKLIST entries description.                 @L81C
# NOCHECKLIST will cause the server not to look at the CHKLIST    @L81C
# information in the EXPORTS data set even if it is there. This   @L81C
# is the default.                                                 @L81C

nochecklist                                                     # @L5DA

# FN_DELIMITER specifies a character to be used instead of a      @L5XA
# comma to delimit the file name from the accompanying attributes.@L5XA
# This will allow those sites which have UNIX data sets containing@L5XA
# commas to copy and store their data on the Z/OS NFS Server.     @L6C
#                                                                 @L5XA
# An example of this would be:                                    @L5XA
#   The attribute data set contains fn_delimiter(;)               @L5XA
#    this changes the default delimiter character from a comma    @L5XA
#    to a semicolon.                                              @L5XA
#                                                                 @L5XA
#   so instead entering ... vi "data_set_name,text,lf"            @L5XA
#                                                                 @L5XA
#   you would enter.........vi "comma,in-name;text,lf"            @L5XA
#                                                                 @L5XA
# This is a new SITE attribute and it's default is a comma        @L5XA
#                                                                 @L5XA
fn_delimiter(,)                                                  #@L5XA

# The hfsfbtimeout is a new timeout attribute to control the      @06A
# timeout of the HFS vnode token used by NFS server.              @06A
# The timeout value controls how long before vnode tokens         @06A
# saved in file blocks are released.                              @06A
#                                                                 @06A
# Valid range is from 1 to 32,767 seconds.                        @06A
#                                                                 @06A
# n can go as low as 1 second  but to avoid the possibility of    @06A
# client hanging (because of network delays),                     @06A
# n is not recommended to be lower than 5 seconds.                @06A
# n may need to be increased if the network is slow and the       @06A
# accessed directory has a lot of entries.                        @06A
# The default hfsfbtimeout is 60 seconds.                         @06A
#                                                                 @06A
hfsfbtimeout(60)                                                 #@06A

# The UPCASE attribute will convert the messages to uppercase.    @07A
# The MIXCASE attribute will display the messages without         @07A
# any modification (as they are).                                 @07A
# The default is upcase.                                          @07A
#                                                                 @07A
upcase                                                           #@07A

# NOREC878 specifies that recovery processing of 878 or 80A      @L66A
# abends will be turned OFF. That is, if this type of abend      @L66A
# occurs, server will shutdown without recovery.                 @L66A
```

Figure 44. NFS system server sample attribute table (Part 18 of 21)

```
# It can be used for Debug only.                                 @L66A
# REC878 specifies that recovery processing of 878 or 80A        @L66A
# abends will be turned ON, and affected tasks will attempt to   @L66A
# recover.                                                       @L66A
# This is default value.                                         @L66A
rec878                                                          #@L66A

# NLM tells the server to start Lockd/Statd.                     @L74A
# NONLM tells the server not to start Lockd/Statd.               @L74A
# If not specified NLM, default is NONLM.                        @L74A

nonlm                                                           #@L74A

# DELEGATION tells the server to enable delegation support.      @LBCA
# NODELEGATION tells the server not to enable delegation support.@LBCA
# If neither is specified, the default is NODELEGATION.          @LBCA

# nodelegation                                                  #@LBCA

# Stringprep Support:
# Stringprep is a standard by which similar characters are       @L82A
# mapped to a common character to improve string compares. NFS   @L82A
# calls Z/OS Coversion Services stringprep services when         @L82A
# receiving network strings such as filenames.                   @L82A
# STRINGPREP tells the server to invoke Stringprep support.      @L82A
# NOSTRINGPREP tells the server to disable Stringprep support.   @L82A
# Default:                                                       @L82A
#    For compatibility with NFS version 4 support in Z/OS V1R7   @L82A
#    stringprep is disabled by default: NOSTRINGPREP             @L82A
#                                                                @L82A
# stringprep                                                    #@L82A
# nostringprep                                                  #@L82A

nostringprep                                                    #@L82A

# Leasetime(n)  specifies the lease time for NFSv4 access.  The  @L74A
# lease time is the length of time for which a client is         @L74A
# irrevocably granted a lock.  At the end of a lease period the  @L74A
# lock may be revoked if the lease has not been extended.        @L74A
# The lock must be revoked if a conflicting lock has been        @L74A
# granted after the lease interval.  The lease time is also      @L74A
# used to specify the grace period duration.  The grace period   @L74A
# is the length of time provided for NFSv4 and NLM clients to    @L74A
# reclaim locks following a server crash and restart.            @L74A
#                                                                @L74A
# n is from 5 seconds to 300 seconds.                            @L74A
# The default value is 120 seconds.                              @L74A
# n must be smaller than logout value if logout is not 0.        @L74A
#                                                                @L74A
# NOTE:  When using the NFS version 4 protocol, the lease time   @P09A
#        value should be set to a value larger than or equal to  @P09A
#        the attr/read/write timeout values.  Otherwise, it is   @P09A
#        possible for performance problems to occur when         @P09A
#        attempting to access MVS data sets.                     @P09A

leasetime(120)                                                  #@L74A

# The REMOUNT attribute enables the NFS Server to process NFS     @0AA
# requests after the NFS Server is restarted even though the HFS  @0AA
# file system was remounted with a new HFS file system number     @0AA
# (device number) after its last usage.                           @0AA
# Use of the REMOUNT attribute causes the NFS Server to           @0AA
```

Figure 44. NFS system server sample attribute table (Part 19 of 21)

```
# automatically access a remounted HFS file system even though it @0AA
# may have been changed prior to remounting.                      @0AA
# Any active client mounts are reestablished.                     @0AA
# The NOREMOUNT attribute enables the NFS Server to fail NFS      @0AA
# requests (with return value NFSERR_STALE) if the HFS file system@0AA
# was remounted with a new HFS file system number (device number) @0AA
# after its last usage.                                           @0AA
# The default is NOREMOUNT.                                       @0AA
# This site attribute applies only to HFS file systems.           @0AA
#                                                                 @0AA
noremount                                                        #@0AA

# DHCP specifies that NFS Clients using Dynamic IP addresses     @L76A
# are supported.  To function properly, this support requies     @L76A
# that the Clients have static Host Names and that they          @L76A
# dynamically update the DNS server with their IP address        @L76A
# changes.                                                       @L76A
# NODHCP specifies that only NFS Clients using Static IP         @L76A
# addresses are supported.                                       @L76A
# The default is NODHCP.                                         @L76A
#                                                                @L76A
nodhcp                                                          #@L76A

# DENYRW specifies that the NFS Version 4 OPEN operation,      @L74M5A
# and the NLM Protocol NLM_Share procedure, should support     @L74M5A
# the Windows Share_Deny value.                                @L74M5A
# NODENYRW specifies that the NFS Version 4 OPEN operation,    @L74M5A
# and the NLM Protocol NLM_Share procedure, should ignore      @L74M5A
# the Windows Share_Deny value for z/OS UNIX file systems (the @L7EC
# value will be treated as if DENY_NONE was specified).        @L74M5A
# The default is DENYRW.                                       @L74M5A
#                                                              @P07C
DENYRW                                                        #@P07C

# mvssec( , , , ) specifies the permissible network            @L74AA
# transmission security for MVS data.                          @L74AA
# The valid flavors are krb5,krb5i,krb5p,sys. All or none of   @L74AA
# them can be specified.                                       @L74AA
# The default value is to allow all pseudo flavors for MVS     @L74AA
# Data sets.                                                   @L74AA
# If mvssec is specified, then only the specified flavors      @L74AA
# will be allowed for MVS Data sets.                           @L74AA
#                                                              @L74AA
mvssec(krb5,krb5i,krb5p,sys)                                  #@L74AA

# hfssec( , , , ) specifies the permissible network            @L74AA
# transmission security for HFS data.                          @L74AA
# The valid flavors are krb5,krb5i,krb5p,sys. All or none of   @L74AA
# them can be specified.                                       @L74AA
# The default value is to allow all pseudo flavors for MVS     @L74AA
# Data sets.                                                   @L74AA
# If hfssec is specified, then only the specified flavors      @L74AA
# will be allowed for HFS Data sets.                           @L74AA
#                                                              @L74AA
hfssec(krb5,krb5i,krb5p,sys)                                  #@L74AA

# pubsec( , , , ) specifies the permissible network            @L74AA
# transmission security for PUBLIC data.                       @L74AA
# The valid flavors are krb5,krb5i,krb5p,sys. All or none of   @L74AA
# them can be specified.                                       @L74AA
# The default value is to allow all pseudo flavors for MVS     @L74AA
# Data sets.                                                   @L74AA
# If pubsec is specified, then only the specified flavors      @L74AA
```

Figure 44. NFS system server sample attribute table (Part 20 of 21)

```
# will be allowed for PUBLIC Data sets.                        @L74AA
#                                                              @L74AA
pubsec(krb5,krb5i,krb5p,sys)                                  #@L74AA

# NFSV4DOMAIN(NFSv4_default_domain) specifies the "pseudo" NFSv4  @015A
# domain for the NFS v4 protocol (NFSv4) name mapping. The        @015A
# "pseudo" NFSv4 domain allows various NFSv4 Clients from various @015A
# network domain to seamlessly access the server provided that    @015A
# these NFSv4 Clients are also configured with the same           @015A
# "NFSv4_default_domain".                                         @015A
# When the NFSV4DOMAIN is omitted, the participating NFSv4        @015A
# Clients' domain must match the Server's network domain for      @015A
# the proper NFS V4 protocol name mapping.                        @015A
#NFSV4DOMAIN(NFSv4_default_domain)                               #@LAGA

#DlyDTimeout(10) Specifies the minimum Delay Detection time value @LBKA
# in sec before the Delay Detection mechanism observes a delay in @LBKA
# an external call/function and prints message GFSA1030E on       @LBKA
# console.                                                        @LBKA
# The maximum value for DlyDTimeout is 60 seconds.                @LBKA
# The minimum value for DlyDTimeout is 5 seconds.                 @LBKA
# Any value of DlyDtimeout from 1 to 4 seconds is                 @P0FC
# converted to 5 seconds.                                         @LBKA
# The Default value is 10 seconds.                                @LBKA
# If DlyDTimeout is set to 0 the DelayDetection mechanism         @LBKA
#    is turned OFF                                                @LBKA
#DlyDTimeout(10)                                                 #@LBKA

# setgid(POSIX | ZOSUNIX) attribute controls GID inheritance for  @012A
# new UNIX objects.                                               @012A
# POSIX - z/OS NFS server                                         @012A
# uses POSIX rules in GID inheritance for new z/OS UNIX objects.  @012A
# If the S_ISGID bit of the parent directory is on,               @012A
# then the new GID is set to the GID of the parent directory.     @012A
# Otherwise, it is set from the GID of the process.               @012A
# A new directory inherits the S_ISGID bit value                  @012A
# of the parent directory.                                        @012A
# ZOSUNIX - z/OS NFS server                                       @012A
# provides the compatibility with z/OS UNIX.                      @012A
# When the RACF profile FILE.GROUPOWNER.SETGID in the UNIXPRIV    @012A
# class is set, z/OS NFS server                                   @012A
# uses POSIX rules, as stated above. Otherwise, a                 @012A
# new GID is always set to the GID of the parent directory,       @012A
# and for a new directory, the S_ISGID bit is always set off.     @012A
# NOTE: Some NFS clients (e.g. SUN, AIX) force GID setting after  @012A
# object creation and prevent compatibility with                  @012A
# z/OS UNIX even though the setgid(ZOSUNIX) attribute is set.     @012A
#                                                                 @012A
# The default is POSIX.                                           @012A
 setgid(POSIX)                                                   #@012A

# The 'overflow' processing attribute for MVS data path turns on  @013A
# x37 detection support for PS/PDS MVS data sets for NFS v2/v3/v4 @016C
# WRITE operations. It allows ENOSPACE errors to be reported to   @013A
# NFS Client in a timely manner and to avoid situations           @013A
# when the z/OS NFS Server closes a data set on timeout expiration@013A
# basis with x37 abend which can not be propagated to the NFS     @013A
# Client. This option may also be activated on the MOUNT level.   @013A
# Default mode is no overflow detection.                          @013A
# Note: 1. PDSE and VSAM data sets are not supported.             @016C
#                                                                 @013A
# overflow                                                       #@014C
```

Figure 44. NFS system server sample attribute table (Part 21 of 21)

Appendix F. Sample exports data set

Figure 45 contains an example exports data set. The exports data set can be found as GFSAPEXP in the NFSSAMP library.

```
#######################################################################
#                                                                     #
#  z/OS   Network File System Server Sample EXPORTS                   #
#                                                                     #
#     PROPRIETARY STATEMENT=                                          #
#      LICENSED MATERIALS - PROPERTY OF IBM                           #
#      THIS MODULE IS "RESTRICTED MATERIALS OF IBM"                   #
#      5647-A01                                                       #
#      COPYRIGHT IBM CORPORATION 1991, 2007                           #
#      SEE IBM COPYRIGHT INSTRUCTIONS                                 #
#     END PROPRIETARY STATEMENT                                       #
#     Copyright IBM Corp. 1991, 2007                                  #
#     Copyright SUN Microsystems, Inc &                               #
#     Electronic Data Systems Corp. 1988, 1989                        #
#                                                                     #
#######################################################################
#  @(#)exports.cntl 3.2 89/04/20 SMI EDS
#
# This file contains EXPORT and CHKLIST entries.
# This file is used for processing EXPORT entries when Server
# security option is set to EXPORT/SAFEXP, and CHKLIST entries
# when Server security option is SAF/SAFEXP modes.
# i.e. being  in security(SAFEXP) mode both EXPORT and CHKLIST
# entries (directory suffixes) are to be processed.
# This file is not used for SECURITY(NONE).
#
# This data set contains entries for directories that may be
# exported to Network File System Clients. It is used by the
# server to determine which data sets and prefixes may be
# accessed by a client, and to write protect data sets on the
# server provided that the SECURITY site attribute is set
# to either SECURITY(EXPORTS) or SECURITY(SAFEXP).  This file
# is not used for SECURITY(NONE).
#
# This data set contains also suffixes of directories that are
# to be exempt from SAF checking even though SAF or SAFEXP is
# specified as the security option. The directory suffixes are
# only used when SAF or SAFEXP is specified for the particular
# data type (i.e. MVS data, HFS data, data accessed using the
# public file handle) AND CHECKLIST option is specified as the
# site attribute. The directory suffixes specified here forms
# file/directory names which HAVE TO MATCH a subsequent mount
# point OR be the parent of a subsequent mount point to allow
# SAF checking to be bypassed for everything underneath that
# mount point.
#
# The mvsnfs.cntl(exports) data set is read during server startup
# processing. Subsequent changes to this data set will not take
# effect until the server is restarted or the EXPORTFS command
# is issued.  However, changes to the data set do NOT affect the
# mounts processed before the server was restarted or before
# EXPORTFS was issued.
#
```

Figure 45. Sample exports data set

© Copyright IBM Corp. 1991, 2010

```
# Errors found in the file are sent to the system log, and the server
# continues processing for minor errors such as undefined host
# names.  Termination is brought about if the EXPORTS data set
# cannot be read or if a syntax error exists.  (In the case of the
# EXPORTFS command, these errors will cause the command to be
# ignored and processing will continue with the original exports
# file in effect.)

# Statement Syntax
#
# Entries can be up to 4096 characters long. Lines can be continued
# by placing a '+' or '\' at the end of the line. A '#' anywhere in the
# data set indicates a comment that extends to the end of the line.
# No spaces may appear in the keywords; however, leading blanks
# are ignored in new or continuation lines.
#
# Wildcard symbols support
#
# The following wildcard symbols can be used to create regular
# expressions within directories in the EXPORTS file.
#
# Symbol          Explanation
#
#  *       Represents any length sequence (can be zero) of any
#          valid characters.
#
# Example  /u/ab*             matches /u/abcde or /u/abfg
#          IBMUSER.HO*.NAME   matches IBMUSER.HOME.NAME or
#                                     IBMUSER.HOLD.NAME
#
#  ?       Represents any (but only one) valid character.
#
# Example  /u/user?           matches /u/user1 or /u/user2
#                             but not /u/user12
#
# [.-.] or
# [...]    Is used to specify one symbol from the region of
#          characters.
#          Characters are regulated in EBCDIC alphabetical order.
#          Open bracket, [, has to be hex value x'AD'
#          Close bracket, ], has to be hex value x'BD'
#
#          NOTE: Hex values x'AD' and x'BD' may represent
#                different characters depending on the
#                code page defined for your installation
#
# Example 1: /u/[a-c]de       matches /u/ade or /u/bde or /u/cde
#            /u/[abd]fg       matches /u/bfg or /u/dfg
#                             but not /u/cfg
# Example 2: Given the set of similar resource names
#
#    /user/abcdef1  /user/abddef1
#
#    /user/abcdef2  /user/abedef2
#
#    /user/abcdef3  /user/abfdef3
#
#    /user/abcdef4
#
```

```
# with wildcard symbols support it is possible to export all
# of these names by only one row in the EXPORTS file:
#
#    /user/ab[c-f]def?
#
# Be aware that with this line other datasets, for example
# /user/abfdef4, are also exported.
#
# MVS system symbols support
#
# The NFS server supports both types of MVS system symbols:
# static and dynamic.
# System-defined static system symbols already have their names
# defined to the system.
# An installation defines a default list of symbols that can be
# adjusted by the system administrator.
#
# Some examples of static MVS system symbols are:
# &SYSCLONE, &SYSNAME, &SYSPLEX, &SYSR1.
# Actual values for static symbols are constant for a running
# system.
#
# In contrast, dynamic MVS system symbols can change their
# values at any time.
# The NFS server will substitute the actual values for the
# the dynamic symbols at NFS start up and on the EXPORTFS NFS
# operator command.
# These values will be fixed in the NFS exports list until
# the next EXPORTFS command execution even if the value of any
# dynamic symbol changes.
# Some examples of dynamic MVS system symbols are:
# &SYYMMDD, &DATE, &SHHMMSS, &YR4.
#
# When used in the EXPORTS file, MVS system symbols
# (static or dynamic) should be preceded by an "&" sign and
# closed with a ".".
# Also it should not be empty (i.e. "&." is not allowed).
# The name of the MVS system symbol should not exceed 80
# characters.
#
#   Examples            /u/&&SYSNAME.
#                       IBMUSER.&&DATE..REPORT
#
# Limitations on the usage of Wildcard symbols and MVS system
# symbols in directories are described under directory_suffix.
#
#
# Entries for a directory are specified by a line of the following
# syntax:
#
#    directory   -ro
#    directory directory_suffix  -ro
#    directory   -rw=clients
#    directory directory_suffix  -rw=clients
#    directory   -access=clients
#    directory directory_suffix  -access=clients
#    directory_suffix
#    directory   -sec=sys|krb5|krb5i|krb5p
#
# where:
#
# directory = Prefix or file name
# directory_suffix = <nosaf>
#                    <hosts=clients,nosaf>
#                    <hosts=list_of_items,nosaf>
#       where item is prefix or file name with optional client
#       list:
#                    hfs_subdir_or_mvs_prefix
#                    hfs_subdir_or_mvs_prefix,hosts=clients
#
#       Clients in the "hosts=" option may be specified as
#       hostname or IP address.
#       Only hosts from the list can have access to the mounted
#       data in SAF or SAFEXP mode without MVSLOGIN.
#       Wildcard symbols and MVS system symbols within a
#       directory must not be used because the checklist entry
#       provides for the exact data path to be excluded from
#       SAF checking.
#
# -ro       = Export the directory as read only. If not specified,
#             the directory is exported read/write to every one.
#
# -rw       = Directory exported read/write to specified clients,
#             and read only to everyone else.  Separate clients
#             specs by '|' (vertical bar).
#
# -access   = Give access only to clients listed. Separate client
#             specs by '|' (vertical bar). This can be further
#             qualified with the "ro" keyword or with an "rw"
#             list.
#
# -sec      = Specifies the Kerberos authentication level that
#             clients must have to access individual files and
#             data sets on the z/OS NFS server.
#
# The keywords "ro" and "rw" are mutually exclusive.
#
# Directory suffix examples.
# 1. Example showing bypass of the SAF checking of multiple
#    sub-directories for the SAME directory.
# Given that the directory tree and the exports list are as
# follows:
#  Directory tree:           Exports:
#  /user1/.                  /hfs/user1<docs,test/release2,nosaf>
#         temp
#         docs.
#             /guides
#         test/.
#              release1
#              release2
#
#  SAF checking will be bypassed for mount point and everything
#  underneath that mount point, i.e. users do not need to do
#  MVSLOGIN for the files/directories underneath the mount point:
#  /hfs/user1/docs
#  /hfs/user1/docs/guides
#  /hfs/user1/test/release2
#
#  SAF checking will be enforced for mount point:
#  /hfs/user1
#  /hfs/user1/temp
#  /hfs/user1/test/release1
# 2. Example for SAFEXP and SAF security.
# Given that security of SAFEXP or SAF is specified and the
# exports list is as follows:
#  Exports:
#  WEBNFS.PDS <hosts=client1|client2,nosaf>
#  USER1hosts=<PUBLIC,NOSAF>
#  /hfs/user1 <tmp,hosts=client1|client2,NOSAF>
#  /hfs/u/public<nosaf>
#  /hfs/bin
#
#  For SAF security only exports entries with NOSAF option will
#  be taken into consideration. Such entries will be used only to
#  bypass SAF checking and to limit access to the mount point
#  with the list of hosts.
#
#  The following mount requests for SAFEXP security will fail
#  because it is not in the exports list:
#    WEBNFS
#    /hfs/u
#
#  The following mount requests for SAFEXP and SAF security will
#  succeed and SAF checking will be bypassed for mount point and
#  everything underneath that mount point (i.e. users do not need
#  to do MVSLOGIN to access mount point and everything underneath
#  that mount point):
#    WEBNFS.PDS
#    USER1.PUBLIC.PDS
#    /hfs/user1/tmp/dir1
#    /hfs/u/public
#  But access to WEBNFS.PDS and /hfs/user1/tmp/dir1 will be
#  allowed only for client1 and client2 and will require the use
#  of MVSLOGIN by users on all other hosts.
#  Access to USER1.PUBLIC.PDS, /hfs/u/public will not be limited
#  (will be possible for any hosts).
#
#  The following mount requests for SAFEXP security will succeed
#  but SAF checking will be enforced:
#    USER1.SOMEPDS
#    /hfs/user1
#    /hfs/bin
#  Access to the mount point will be possible for any hosts.
#
#  The following mount requests for SAF security will succeed
#  (no exports checking) but SAF checking will be enforced:
#    WEBNFS
#    USER1.SOMEPDS
#    /hfs/u
#    /hfs/bin
#    /hfs/dir_which_is_absent_in_exports_list_but_exist
#  Access to the mount point will be possible for any hosts.
#
# 3. Example showing usage of a directory suffix with nothing
#    before it.
# Given that security of SAFEXP or SAF is specified and the
# exports list and the PUBLIC z/OS NFS site attribute are as
# follows:
#  Exports:
#  <WEBNFS.PDS,hosts=client1|client2,nosaf>
#  </hfs/user1/tmp,hosts=client1|client2,NOSAF>
#  <nosaf>
#  <hosts=client1|client2,nosaf>
#  PUBLIC:
#  PUBLIC(WEBNFS.PDS,/hfs/user1/tmp)
#  SAF checking will be bypassed for PUBLIC mount points and
#  everything underneath these PUBLIC mount points:
#  WEBNFS.PDS
#  /hfs/user1/tmp
#  But access to WEBNFS.PDS and /hfs/user1/tmp will be allowed
#  only for client1 and client2 and will require the use of
#  MVSLOGIN by users on all other hosts.
#
#  The entries of the exports list will be ignored but will cause
#  syntax error:
#  <nosaf>
#  <hosts=client1|client2,nosaf>
#
# Clients in the client list must be separated with '|' and may
# contain a client suffix. Client suffix can be used to give the
# root user of the client root access to the exported directory.
# Client suffix only applies to Security EXPORTS mode. It is
# ignored in SAF and SAFEXP Security modes.
# Client suffix has syntax:
#
# <root>
#
# Client and its suffix combines as:
#
# client<root>   NFSS will honor client UID = 0 (trusted user).
#                NFSS will give client's user with UID=0
#                (trusted user) superuser access on the directory
#                while other users on that client will get normal
#                access to the directory the same as if the
#                client suffix (i.e. <root>  ) was not specified
#                for that client.
#
# Clients in the client list can be specified in a number of ways
# (depending on DHCP z/OS NFS site attribute):
#
# single hostname
#             You may specify a hostname regardless of DHCP Z/OS
#             NFS site attribute. It may be specified without the
#             domain part.
#             You may specify client suffix. Example:
#             host1<root>
#             host1.cs.foo.edu<root>
#
# netgroup name
#             Netgroup name (from the local /etc/netgroup file)
#             may be given as @groupname. You may specify a
#             netgroup name regardless of DHCP z/OS NFS site
#             attribute.
#             You may NOT specify client suffix (i.e. <root>)
#             with netgroup name.
# single IP address
#             You may specify a client by an IPv4 or IPv6 address.
#             If DHCP z/OS NFS site attribute is "dhcp" such
#             clients will be ignored. If NFS Server starts up in
#             IPv4 mode all clients specified with IPv6 addresses
#             will be ignored. If NFSS starts up in IPv6 mode all
#             IPv4 addresses of clients will be translated to
#             IPv4-mapped addresses.
#             You may specify client suffix (i.e. <root>).
#
# IP network template
#             You can export directory to all hosts on an IPv4
#             and IPv6 (sub-) network simultaneously. If DHCP
#             attribute is "dhcp" such IP network template will
#             be ignored.
#             You may NOT specify client suffix (i.e. <root>).
#             For IPv4, a network template is specified by an
#             IPv4 address and netmask pair as address/netmask
#             where a netmask must be specified in dotted-decimal
#             format, or as a contiguous mask length (for example,
#             either '/255.255.252.0' or '/22' appended to the
#             network base address result in identical subnetworks
#             with 10 bits of host). If NFS Server starts up in
#             IPv6 mode IPv4 network template will be converted
#             to a template based on IPv4-mapped addresses.
#             For IPv6, a network template is specified by an
#             ipv6-address/prefix-length (for example, the node
#             address 12AB:0:0:CD30:123:4567:89AB:CDEF and its
#             subnet number 12AB:0:0:CD30::/60 can be abbreviated
#             as 12AB:0:0:CD30:123:4567:89AB:CDEF/60). If NFS
#             Server starts up in IPv4 mode IPv6 network template
#             will be ignored.
#
# hostname template
#             Machine names may contain the wildcard characters
#             '*' and '?'. This can be used to make the exports
#             file more compact; for instance, *.cs.foo.edu
#             matches all hosts in the domain cs.foo.edu. However,
#             these wildcard characters do not match the dots in
#             a domain name, so the above pattern does not include
#             hosts such as a.b.cs.foo.edu.
#             For hostname template the domain part is mandatory.
#             If DHCP z/OS NFS site attribute is "nodhcp" such
#             entries will be ignored.
#             You may NOT specify client suffix (i.e. <root>).
#
# If no options are specified, the default value
# allows any client read/write access to the given directory.
#
#
# If 'maplower' is specified in the default attributes, all
# entries are translated to upper case. High level qualifiers
# specified in the client mount request are translated to
# upper case.
#
# If 'nomaplower' is specified in the default attributes,
# attention must be given to the case of entries. High level
# qualifiers specified in the client mount request are not
# translated to upper case.
```

```
mvsnfs                   -ro
userid0.mixds            -rw=fsrs001|fslab004<root>|9.43.249.211|\
              9.43.249.206<root>|fe80::204:acff:fee4:b7b6|\
              fe80::a00:20ff:feb9:4a65<root>|fssun??.cs.foo.edu|\
              *.de.foo.net|10.20.324.151/255.255.255.240|\
              10.33.324.151/28|fe80::b04:ac20:feff:a776/121|@hpnetgrp
userid1<nfs,nosaf>       -rw=fsrs001
userid2.nfs.pds <nosaf>  -access=fsrs001|fssun03<root>
userid3.nfs.ps           -sec=krb5|krb5i|krb5p|sys
userid4.nfs.another.pds  -access=fslab004
/hfs/u/newproducts       -access=fsrs001,ro
```

Appendix G. Sample startup procedures

The following text contains sample startup procedures that you can use to start the z/OS NFS server and client.

Note: Although both the following sample startup procedures include the SYSTCPD DD statement, SYSTCPD is not required for either the NFS server or the NFS client, unless a custom configuration is required for NFS. To see how the configuration is determined when the SYSTCPD DD card is not specified in the NFS startup procedures, see the documented search order for the resolver configuration files in the section "Search orders used in the z/OS UNIX environment" in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

Sample z/OS NFS server startup procedures

Figure 46 on page 496 contains a sample MVSNFS z/OS NFS server startup procedure. This procedure can be found as GFSAPROC in the NFSSAMP library.

© Copyright IBM Corp. 1991, 2010

```
//MVSNFS  PROC MODULE=GFSAMAIN,PARMS='INFO',
//           SYSNFS=SYS1,SYSLE=SYS1,NFSPRFX=MVSNFS              @002C
//*          SYSNFS=SYS1,SYSLE=SYS1,NFSPRFX=MVSNFS,             @002A
//*          TCPIP=TCPIP                                        @002A
//********************************************************************
//*                                                                  *
//*  Z/OS NETWORK FILE SYSTEM SERVER START UP PROC(HDZ1AN0)    @LA0C*
//*                                                                  *
//*PROPRIETARY V3 STATEMENT                                          *
//*LICENSED MATERIALS - PROPERTY OF IBM                              *
//*"RESTRICTED MATERIALS OF IBM"                                     *
//*5694-A01                                                          *
//*    COPYRIGHT 1989, 2008 IBM CORP.                          @LA0C*
//*END PROPRIETARY V3 STATEMENT                                      *
//*                                                                  *
//*  CHANGE HISTORY                                                  *
//*                                                                  *
//*  $01=OY55396,HDP3FS1,920616,SJPLHYC : ENHANCED ERROR LOG    @01A *
//*  $L12=NFS,JDZ1170,920925,SJPLHYC    : ADD SYSMDUMP DD      @L12A *
//*  $L3L=KA90033,HDZ11NP,940117,SJPLHYC: MODIFY FOR DFSMS 1.2 @L3LA *
//*  $L3G=NFS,HDZ11NP,940420,SJPLJST    : ADD NFSXLAT DD       @L3GA *
//*  $P1=KA901XX,HDZ11NP,940805,SJPLJST : CHANGE &SYS1          @P1A *
//*  $P2=KA00045,HDZ11SM,960111,SJPLTEM : MODIFY FOR DFSMS 1.3 @P02A *
//*  $P3=KA00093,HDZ11SM,960327,SJPLTEM : ADD HDZ11SM STEPLIB  @P03A *
//*  $P4=KA00101,HDZ11SM,960410,SJPLTEM : SWITCH TO HDZ11SM    @P04A *
//*  $P5=KAB0114,HDZ11TS,971030,SJPLTEM : CHG DFSMS/MVS->OS/390@P05A *
//*  $P6=KAB0316,HDZ11TS,980325,SJPLBPF : ADD CHKLIST DD       @P06A *
//*  $L6=NFS,HDZ11US,030405,SJPLMB : CHG OS/390 TO Z/OS        @L6A  *
//*  $L65=NFS,HDZ11US,030214,IBSKVV:     UPGRADE TO SP4 MACRO  @L65A *
//*  $L781=NFS,HDZ11VS,041207,SJPLMB : CHANGE NFSLIB TO NFSLIBE@L781A*
//*  $L74=NFS,HDZ11VS,050308,SJPLSLH : ADD LOCKING DB DD DEF   @L74FA*
//*  $P07=KAJ0355,HDZ11VS,050414,SJPLMB: UPDATE TO HDZ11VS     @P07A *
//*  $L80=NFS,HDZ118N,050512,SJPLMB: UPDATE RELEASE TO HDZ118N @L80A *
//*  $L81=NFS,HDZ118N,050908,IBSDYP: REMOVE CHKLIST DD CARD    @L81A *
//*  $L90=NFS,HDZ119N,060131,SJPLMB: UPDATE RELEASE TO HDZ119N @L90A *
//*  $LA0=NFS,HDZ1AN0,070110,SJPLMB: UPDATE RELEASE TO HDZ1AN0 @LA0A *
//*  $P08=KN00140,HDZ1AN0,071022,SJPLMB: REMOVE SYSTCPD        @P08A *
//*  $002=OA25770,HDZ1AN0,080709,SJPLMTM:                      @002A *
//*       1. REMOVE TCPIP PARM FROM PROC STMT                  @002A *
//*                                                                  *
//********************************************************************
//GFSAMAIN EXEC PGM=&MODULE,
//             PARM='&PARMS',
//             REGION=0M,
//             TIME=1440
//*
//*
//* STEPLIB CONSISTS OF NFSLIBE AND LANGUAGE ENVIRONMENT LIBRARIES
//*         CHANGE THE NAMES AS APPROPRIATE FOR YOUR INSTALLATION
//*  NOTE: EACH OF THESE LIBRARIES MUST BE AUTHORIZED AND
//*        LANGUAGE ENVIRONMENT V1R5M0 OR HIGHER IS REQUIRED
//*
//* &TCPIP..TCPIP.DATA IS TCP/IP DATA FILE
//* &SYSNFS..NFSLIBE IS Z/OS NETWORK FILE SYSTEM SERVER       @L781C
//*                    TARGET LIBRARY
//* &SYSLE..SCEERUN IS LANGUAGE ENVIRONMENT RUNTIME LIBRARY
//*
//* SYSTCPD EXPLICITLY IDENTIFIES WHICH DATA SET IS TO BE     @P08A
//* USED TO OBTAIN THE PARAMETERS DEFINED BY TCPIP.DATA       @P08A
//* WHEN NO GLOBALTCPIPDATA STATEMENT IS CONFIGURED.          @P08A
//* SEE THE IP CONFIGURATION GUIDE FOR INFORMATION ON         @P08A
//* THE TCPIP.DATA SEARCH ORDER.                              @P08A
//* THE DATA SET CAN BE ANY SEQUENTIAL DATA SET OR A MEMBER   @P08A
//* OF A PARTITIONED DATA SET (PDS).                          @P08A
//*                                                           @002A
```

Figure 46. Sample z/OS NFS server startup procedures

```
//* NOTE: THE TCPIP PARAMETER ON THE PROC STATEMENT IS         @002A
//*       REQUIRED WHEN USING THE SYSTCPD DD STATEMENT.        @002A
//*                                                            @002A
//*
//*SYSTCPD  DD   DISP=SHR,DSN=&TCPIP..TCPIP.DATA(TCPDATA)      @P08C
//*
//STEPLIB  DD   DISP=SHR,DSN=&SYSNFS..NFSLIBE                 @L781C
//        DD   DISP=SHR,DSN=&SYSLE..SCEERUN                     @P1C
//*                                                             @L3LD
//SYSPRINT DD   SYSOUT=*
//*********************************************************************
//*                                                                   *
//* THE SYSMDUMP DATASET NEEDS TO BE PRE-ALLOCATED TO THE      @L12A *
//* DCB SPECIFICATIONS REQUIRED BY SYSMDUMP                     @L12A *
//*                                                                   *
//*********************************************************************
//SYSMDUMP DD   DISP=SHR,DSN=&NFSPRFX..SYSMDUMP                 @L12A
//OUTPUT   DD   SYSOUT=*
//SYSERR   DD   SYSOUT=*
//*
//*********************************************************************
//*                                                                   *
//* NFSLOG1 AND NFSLOG2 DD'S ARE ADDED.                          @01A *
//* THE TWO LOG DATA SETS NEED TO BE PRE-ALLOCATED               @01A *
//*  AS SEQUENTIAL FILES.                                        @01A *
//* THE DCB WILL BE SET TO (VB,LRECL=137,BLKSIZE=6144)           @01A *
//* ONCE MVSNFS IS STARTED, REGARDLESS OF THE ALLOCATION DCB.   @01A *
//* INITIALLY, WE RECOMMEND SPACE=(CYL,(2,5),RLSE)               @01A *
//*                                                                   *
//*********************************************************************
//NFSLOG1  DD   DISP=SHR,DSN=&NFSPRFX..LOG1                      @01A
//NFSLOG2  DD   DISP=SHR,DSN=&NFSPRFX..LOG2                      @01A
//*
//*  &NFSPRFX..CNTL IS A FB LRECL 80 DATASET (PDS OR PDSE)       @L3LC
//*  SAMPLE NFSATTR CAN BE FOUND IN GFSAPATT IN &NFSPRFX.NFSSAMP @L3GC
//*  SAMPLE EXPORTS CAN BE FOUND IN GFSAPEXP IN &NFSPRFX.NFSSAMP @L3GC
//*****NOTE:   AS OF Z/OS V1R8, THE EXPORTS FILE IS COMPOSED OF  @L81A
//*                 BOTH EXPORT AND CHKLIST ENTRIES               @L81A
//*
//NFSATTR  DD   DISP=SHR,DSN=&NFSPRFX..CNTL(NFSATTR)
//EXPORTS  DD   DISP=SHR,DSN=&NFSPRFX..CNTL(EXPORTS)
//*********************************************************************
//*                                                                   *
//*       FHDBASE AND FHDBASE2 ARE                                @L3LC
//*       THE MOUNT FILE HANDLE DATABASES.                        @L3LC
//*       THEY NEED TO BE PREALLOCATED                            @L3LC
//*       AS EMPTY VSAM KSDS DATA SETS.                           @L3LC
//*       THEY WILL BE USED ALTERNATELY.                          @L3LC
//*       SAMPLE JCL CAN BE FOUND IN &NFSPRFX.NFSSAMP(GFSAMHDJ). @L3GC
//*                                                                   *
//*********************************************************************
//FHDBASE  DD   DISP=SHR,DSN=&NFSPRFX..FHDBASE                    @L3LC
//FHDBASE2 DD   DISP=SHR,DSN=&NFSPRFX..FHDBASE2                   @L3LA
```

```
//*********************************************************************
//*                                                                   *
//*       LDBASE AND LDBASE2 ARE                                @L74FA
//*       THE LOCKING DATABASES.                                @L74FA
//*       THEY NEED TO BE PREALLOCATED                          @L74FA
//*       AS EMPTY VSAM KSDS DATA SETS.                         @L74FA
//*       THEY WILL BE USED ALTERNATELY.                        @L74FA
//*       SAMPLE JCL CAN BE FOUND IN &NFSPRFX.NFSSAMP(GFSAMHDJ).@L74FA
//*                                                                   *
//*********************************************************************
//LDBASE  DD   DISP=SHR,DSN=&NFSPRFX..LDBASE                    @L74FA
//LDBASE2 DD   DISP=SHR,DSN=&NFSPRFX..LDBASE2                   @L74FA
//*********************************************************************
//*                                                                   *
//* IF THE INSTALLATION INTENDS TO CUSTOMIZE THE TRANSLATION     @L3GA
//* TABLE, A NEW DD CARD, NFSXLAT, IS REQUIRED IN THE PROC.      @L3GA
//* THE TRANSLATION TABLE DATASET NEEDS TO BE PRE-ALLOCATED      @L3GA
//*  AS PDS OR PDSE DATASET. THE FORMAT OF THE DATASET HAS TO    @L3GA
//*  CONFORM TO THE TRANSLATION TABLE FORMAT SUPPORTED BY        @L3GA
//*  TCP/IP FOR Z/OS.                                            @L6C
//*                                                                   *
//*********************************************************************
//*NFSXLAT  DD   DISP=SHR,DSN=&NFSPRFX..XLAT                    @L3GA
//*
//*********************************************************************
//* IF IT IS NECESSARY TO USE MVSSNAP() C-FUNCTION               @L65A
//* A NFSSNAP DD STATEMENT IS REQUIRED.                          @L65A
//* THE NFSSNAP DATA SET NEEDS TO BE PRE-ALLOCATED               @L65A
//* THE FORMAT OF THE DATA SET IS:                               @L65A
//* DSORG=PS,                                                    @L65A
//* RECFM=VBA,                                                   @L65A
//* BLKSIZE=1632,                                                @L65A
//* LRECL=125                                                    @L65A
//*
//* UNCOMMENT THE NFSSNAP DD STATEMENT IN CASE OF USING MVSSNAP()@L65A
//*
//*********************************************************************
//*NFSSNAP  DD  DISP=MOD,DSN=&NFSPRFX.NFSSNAP                    @L65A
//NFSSNAP   DD  DUMMY                                            @L65A
```

Sample z/OS NFS client startup procedures

Figure 47 contains a sample MVSNFSC z/OS NFS client startup procedure. This procedure can be found as GFSCPROC in the NFSSAMP library.

```
//MVSNFSC PROC SYSNFS=SYS1,SYSLE=SYS1,NFSPRFX=MVSCLNT            @002C
//*MVSNFSC PROC SYSNFS=SYS1,SYSLE=SYS1,NFSPRFX=MVSCLNT,          @002A
//*          TCPIP=TCPIP                                         @002A
//**********************************************************************
//*                                                                    *
//*  Z/OS NETWORK FILE SYSTEM CLIENT START UP PROC(HDZ1AN0)    @LA0C *
//*                                                                    *
//*  CHANGE HISTORY                                                    *
//*                                                                    *
//*  $L41=NFSC,HDZ11SE,960120,SJPLJST: NFS Client support              *
//*  $KAB0020=NFSC,HDZ11TC,970425,SJPLPKL: Update FMID                 *
//*  $OW52984=NFSC,HDZ11TC,020322,IBSPIV:                              *
//*     1. replace DD with SYSOUT=*  (SYSPRINT, SYSOUT, SYSERR and     *
//*                            OUTPUT) to use DSN specification    *
//*     2. add SYSIN and CEEDUMP DD's to reduce LE activity            *
//*  $L6=NFSC,HDZ11UC,030405,SJPLMB:  Change OS/390 to Z/OS            *
//*  $L781=NFS,HDZ11VC,041207,SJPLMB : Change NFSLIB to NFSLIBE@L781A*
//*  $P01=KAJ0356,HDZ11VC,050414,SJPLMB : Changed to HDZ11VC    @P01A*
//*  $L80=NFS,HDZ118N,050512,SJPLMB: Update release to HDZ118N  @L80A*
//*  $L90=NFS,HDZ119N,060201,SJPLMB: Update release to HDZ119N  @L90A*
//*  $LA0=NFS,HDZ1AN0,070110,SJPLMB: Update release to HDZ1AN0  @LA0A*
//*  $P02=KN00140,HDZ1AN0,071022,SJPLMB: REMOVE SYSTCPD         @P02A*
//*  $002=OA25770,HDZ1AN0,080709,SJPLMTM:                       @002A*
//*      1. REMOVE TCPIP PARM FROM PROC STMT                    @002A*
//*                                                                    *
//**********************************************************************
//MVSCLNT  EXEC PGM=BPXVCLNY,
//             REGION=0M,
//             TIME=1440
//*
//*     STEPLIB CONSISTS OF NFSLIBE AND LANGUAGE ENVIRONMENT LIBRARIES
//*             CHANGE THE NAMES AS APPROPRIATE FOR YOUR INSTALLATION
//*       NOTE: EACH OF THESE LIBRARIES MUST BE AUTHORIZED
//*
//* &TCPIP..TCPIP.DATA IS TCP/IP DATA FILE
//* &SYSNFS..NFSLIBE IS Z/OS NETWORK FILE SYSTEM CLIENT        @L781C
//*                     TARGET LIBRARY
//* &SYSLE..SCEERUN IS LANGUAGE ENVIRONMENT RUNTIME LIBRARY
//*
//* SYSTCPD EXPLICITLY IDENTIFIES WHICH DATA SET IS TO BE      @P02A
//* USED TO OBTAIN THE PARAMETERS DEFINED BY TCPIP.DATA        @P02A
//* WHEN NO GLOBALTCPIPDATA STATEMENT IS CONFIGURED.           @P02A
//* SEE THE IP CONFIGURATION GUIDE FOR INFORMATION ON          @P02A
//* THE TCPIP.DATA SEARCH ORDER.                               @P02A
//* THE DATA SET CAN BE ANY SEQUENTIAL DATA SET OR A MEMBER    @P02A
//* OF A PARTITIONED DATA SET (PDS).                           @P02A
//*                                                            @002A
//* NOTE: THE TCPIP PARAMETER ON THE PROC STATEMENT IS         @002A
//*       REQUIRED WHEN USING THE SYSTCPD DD STATEMENT.        @002A
//*                                                            @002A
//*
//*SYSTCPD  DD   DISP=SHR,DSN=&TCPIP..TCPIP.DATA(TCPDATA)      @P02C
//*
//STEPLIB  DD   DISP=SHR,DSN=&SYSNFS..NFSLIBE                  @L781C
//         DD   DISP=SHR,DSN=&SYSLE..SCEERUN
//*
//**********************************************************************
//*                                                                    *
//* THE SYSPRINT DD NEEDS TO BE PRE-ALLOCATED                          *
//*  AS A SEQUENTIAL FILE.                                             *
//* THE DCB WILL BE SET TO (VB,LRECL=137,BLKSIZE=6144)                 *
//*                                                                    *
//**********************************************************************
//SYSPRINT DD   DISP=SHR,DSN=&NFSPRFX..SYSP
```

Figure 47. Sample z/OS NFS client startup procedures

```
//********************************************************************
//*                                                                  *
//* THE SYSMDUMP DATASET NEEDS TO BE PRE-ALLOCATED TO THE            *
//* DCB SPECIFICATIONS REQUIRED BY SYSMDUMP                          *
//*                                                                  *
//********************************************************************
//SYSMDUMP DD   DISP=SHR,DSN=&NFSPRFX..SYSMDUMP
//********************************************************************
//*                                                                  *
//* THE SYSOUT, SYSERR AND OUTPUT DD'S NEED TO BE PRE-ALLOCATED      *
//*  AS A SEQUENTIAL FILES.                                          *
//* THE DCB WILL BE SET TO (VB,LRECL=137,BLKSIZE=6144)               *
//*                                                                  *
//********************************************************************
//SYSOUT   DD   DISP=SHR,DSN=&NFSPRFX..SYSP
//SYSERR   DD   DISP=SHR,DSN=&NFSPRFX..SYSP
//OUTPUT   DD   DISP=SHR,DSN=&NFSPRFX..SYSP
//SYSIN    DD   DUMMY
//CEEDUMP  DD   DUMMY
//*
//********************************************************************
//*                                                                  *
//* NFSCMSG1 AND NFSCMSG2 DD'S ARE                                   *
//* THE TWO CLIENT LOG DATA SETS NEED TO BE PRE-ALLOCATED            *
//*  AS SEQUENTIAL FILES.                                            *
//* THE DCB WILL BE SET TO (VB,LRECL=137,BLKSIZE=6144)               *
//* ONCE MVSCLNT IS STARTED, REGARDLESS OF THE ALLOCATION DCB.       *
//* INITIALLY, WE RECOMMEND SPACE=(CYL,(2,5),RLSE)                   *
//*                                                                  *
//********************************************************************
//NFSCMSG1 DD   DISP=SHR,DSN=&NFSPRFX..LOG1
//NFSCMSG2 DD   DISP=SHR,DSN=&NFSPRFX..LOG2
```

Appendix H. Retrieval of source code for client enabling commands

Figure 48 shows how you can retrieve the necessary source code to install the client enabling commands on any platform except an AIX, UNIX, or z/OS client. See "Installing the client enabling commands" on page 198 for information on retrieving the necessary source code to install the client enabling commands on AIX or UNIX workstation. The z/OS NFS client source code is installed when the z/OS NFS client and TCP/IP are installed. In this example, **mvshost1** is the name of the z/OS host and **smith** is a z/OS user ID:

```
C> md client         (to create a directory to contain clientcode)
C> cd client
C> ftp mvshost1
Connected to mvshost1
220-FTPSERVE at MVSHOST1, 01:44:24 on 12/02/07
220 Connection closes if idle for more than 5 minutes.
Name (mvshost1:nobody):  smith
<Press ENTER key>
331 Send password please.
Password: password

230 smith is logged on.
ftp> get 'mvsnfs.nfstarb(gfsawaxd)' gfsawaxd.c
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWAXD)
250 Transfer completed successfully.
local: gfsawaxd.c remote: 'mvsnfs.nfstarb(gfsawaxd)'
19516 bytes received in 0.14 seconds (139.4 Kbytes/s)
ftp> get 'mvsnfs.nfstarb(gfsawclt)' gfsawclt.c
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWCLT)
250 Transfer completed successfully.
local: gfsawclt.c remote: 'mvsnfs.nfstarb(gfsawclt)'
17584 bytes received in 0.19 seconds (92.36 Kbytes/s)
ftp> get 'mvsnfs.nfstarb(gfsawlin)' gfsawlin.c
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWLIN)
250 Transfer completed successfully.
local: gfsawlin.c remote: 'mvsnfs.nfstarb(gfsawlin)'
52808 bytes received in 0.25 seconds (211.23 Kbytes/s)
ftp> get 'mvsnfs.nfstarb(gfsawlou)' gfsawlou.c
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWLOU)
250 Transfer completed successfully.
local: gfsawlou.c remote: 'mvsnfs.nfstarb(gfsawlou)'
22468 bytes received in 0.14 seconds (160.49 Kbytes/s)
ftp> get 'mvsnfs.nfstarb(gfsawmcl)' gfsawmcl.c
```

Figure 48. Retrieving source code for client enabling commands (Part 1 of 2)

© Copyright IBM Corp. 1991, 2010

```
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWMCL)
250 Transfer completed successfully.
local: gfsawmcl.c remote: 'mvsnfs.nfstarb(gfsawmcl)'
17302 bytes received in 0.12 seconds (144.18 Kbytes/s)
ftp> get 'mvsnfs.nfstarb(gfsawmou)' gfsawmou.c
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWMOU)
250 Transfer completed successfully.
local: gfsawmou.c remote: 'mvsnfs.nfstarb(gfsawmou)'
29110 bytes received in 0.17 seconds (171.24 Kbytes/s)
ftp> get 'mvsnfs.nfstrab(gfsawrp6)' gfsawrp6.h
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWRP6)
250 Transfer completed successfully.
local: gfsawrp6.h remote: 'mvsnfs.nfstarb(gfsawrp6)'
27880 bytes received in 0.16 seconds (174.25 Kbytes/s)
ftp> get 'mvsnfs.nfstrab(gfsawrs6)' gfsawrs6.h
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWRS6)
250 Transfer completed successfully.
local: gfsawrs6.h remote: 'mvsnfs.nfstarb(gfsawrs6)'
204262 bytes received in 0.94 seconds (216.84 Kbytes/s)
ftp> get 'mvsnfs.nfstarb(gfsawsha)' gfsawsha.c
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWSHA)
250 Transfer completed successfully.
local: gfsawsha.c remote: 'mvsnfs.nfstarb(gfsawsha)'
141122 bytes received in 0.61 seconds (230.97 Kbytes/s)
ftp> get 'mvsnfs.nfstarb(gfsawmnt)' gfsawmnt.h
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWMNT)
250 Transfer completed successfully.
local: gfsawmnt.h remote: 'mvsnfs.nfstarb(gfsawmnt)'
10414 bytes received in 0.09 seconds (115.71 Kbytes/s)
ftp> get 'mvsnfs.nfstarb(gfsawsho)' gfsawsho.h
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWSHO)
250 Transfer completed successfully.
local: gfsawsho.h remote: 'mvsnfs.nfstarb(gfsawsho)'
7052 bytes received in 0.07 seconds (100.74 Kbytes/s)
(Note: Make sure that the tabs key in the gfsawjcl makefile will not
 be translated into a space key for the following step.)
ftp> get 'mvsnfs.nfstarb(gfsawjcl)' makefile
200 Port request OK.
125 Sending data set MVSNFS.NFSTARB(GFSAWJCL)
250 Transfer completed successfully.
local: makefile remote: 'mvsnfs.nfstarb(gfsawjcl)'
8364 bytes received in 0.07 seconds (119.49 Kbytes/s)
ftp> quit
221 Quit command received. Goodbye.
C>
C> (Port commands source code to be compiled in your client
    workstation.)
C> (Set up the path to the "client" directory for the users to use
    the executable code of mvslogin, mvslogout,
    and showattr commands.)
C>  mvslogin mvshost1 smith         (MVSNFS must be operational
Password required                          on host side.)
GFSA973A  Enter MVS password:
GFSA955I  smith logged in ok.
C>
```

Figure 48. Retrieving source code for client enabling commands (Part 2 of 2)

Appendix I. PCNFSD protocol

The PCNFSD protocol enables clients to access z/OS files without issuing the **mvslogin** command. Check with your workstation administrator to see if your clients have PCNFSD support.

Accessing data with PCNFSD

NFS creates a unique UID and GID for each client, and the UIDs and GIDs are consistent within a session. For example, the server might return the UID and GID set (100,100) for one user, and (101,100) for the next user.

If the client maintains the UID and GID for the mount point base, you should use the PCNFSD requests rather than the **mvslogin** client enabling command. However, if the client does not maintain the UID and GID for the mount point base, you should use the mvslogin client enabling commands rather than PCNFSD requests.

If you issue the **mount** command, which issues a PCNFSD request (as it is implemented on some platforms), to do the authentication, you should not use the mvslogin client enabling command.

If the client's z/OS user ID, which was authenticated by a PCNFSD request issued by a mount command, is logged off by the server due to timeout or server restart, then the client needs to issue the **umount** and mount commands to log on again.

See Part 1, "Using z/OS Network File System," on page 1 for information about how clients use PCNFSD.

Accessing z/OS UNIX files

For z/OS UNIX file access, the user must first have a TSO/E user ID defined to RACF with a z/OS UNIX segment defining the UID and GID. When the PCNFSD authenticate request is received, the Network File System uses the UID and GID associated with the user's RACF profile z/OS UNIX segment, to return in the client's UID and GID credentials. It is recommended that the RACF installation preserve a unique network UID and GID that each user is associated with throughout the network. If there is no z/OS UNIX segment for a user, the client's UID and GID credentials are assigned as presented in the following paragraphs.

If **security(saf)** or **security(safexp)** is specified in the site attributes and the client user does not have a valid z/OS UNIX segment, a system assigned UID and GID is returned in the client user credentials.

If **security(none)** or **security(exports)** is specified in the site attributes, a system assigned UID and GID is returned in the client user credentials.

Starting the PCNFSD server

Specify **pcnfsd** or **nopcnfsd** in the attributes data set to control whether to start the PCNFSD server.

© Copyright IBM Corp. 1991, 2010

Using supported PCNFSD protocols

You can use either version 1 or 2 of the PCNFSD protocol with the Network File System.

Version 1 of the PCNFSD protocol

The PCNFSD Version 1 protocol is specified in *Developer's Specification Protocols for X/Open PC Interworking (PC) NFS*. The z/OS Network File System supports procedures 0 and 1 of the version 1 PCNFSD protocol.

Procedure 0: do nothing

Procedure 0 (NULL) does nothing.

Procedure 1: perform user authentication

The input parameters for this procedure are the z/OS user ID and password. The output parameters are the return code, UID and GID. If the z/OS user ID and password are verified successfully (return code=0), the corresponding UID and GID values are returned.

The caller receives one of these return codes:	• **Return code = 0** - Authentication successful • **Return code = 2** - Authentication failed

Version 2 of the PCNFSD protocol

z/OS NFS supports procedures 0, 1, 12, and 13 of the PCNFSD Version 2 protocol.

Procedure 0: do nothing

Procedure 0 (NULL) does nothing.

Procedure 1: give descriptions

Procedure 1 describes how the NFS supports each procedure of the PCNFSD version 2 Protocol.

The input parameters are the version information and comments from the client platform. The NFS ignores the information in the input parameters.

The output parameters are the version information, comments, number of procedures in the PCNFSD version 2 protocol, and a list of descriptions about how the NFS supports each procedure in this PCNFSD version 2 protocol.

There are three values to represent how the server supports the procedures:	• **-1** - The procedure is not supported • **100** - The procedure gets quick service from the server • **2000** - The procedure takes more time to complete the service from the server

Procedure 12: perform mapping

Procedure 12 performs mapping the UID to user name, GID to group name, or group name to GID depending on the request identifier.

If the mapping is successful, the procedure returns successfully. If the names (or IDs) do not exist, or the caller does not have permission to do the mapping, the procedure returns an error.

Procedure 13: perform user authentication

The input parameters are local host name, encoded user name, encoded password, and comments from the requestor. NFS ignores the local host name and comments from the requestor, and decodes the user name and password in the input parameters. NFS does authentication based on the user name and password.

The output parameters are:

There are only two possible return codes that can be sent back to the caller:	• **Return code = 0** - Authentication successful • **Return code = 2** - Authentication failed
UID and GID	If user name and password are verified successfully (return code = 0), the corresponding UID and GID values are returned.
Number of alternate GIDs	Because NFS does not support alternate GIDs, this field is set to zero.
Pointer to a list of the alternate GIDs	Because NFS does not support alternate GIDs, this field is set to zero.
Logon Home Directory	Because NFS does not support logon home directory, this field is zero.
Default umask	Because NFS does not support default umask, this field is set to zero.
Comments sent to the requestor	If the return code is 2, this comment field contains the reason why the request failed.

Appendix J. SMF C and assembler header macros

The following text contains copies of the SMF C and Assembler header macros for your reference. The SMF C header macro is GFSASSMF and the Assembler header macro is GFSAUSMF. Both header macros can be found in the SYS1.NFSMAC library.

SMF C header macro GFSASSMF

Figure 49 contains information about the SMF C header macro GFSASSMF. The GFSASSMF macro can be found in the SYS1.NFSMAC macro library.

```
/*********************************************************************/ 00050000
/*                                                                   */ 00100000
/*  $MAC(GFSASSMF) COMP(5695DF121): NFSS XI                          */ 00150000
/*                                                                   */ 00200000
/*  MACRO NAME: GFSASSMF.H (SMF.H)                                   */ 00250000
/*                                                                   */ 00300000
/*  DESCRIPTION:  Contains C Language Mapping of SMF type 42         */ 00350000
/*                and the subtype 7, 8 and 26 records.         @LCOC */ 00400000
/*                                                                   */ 00450000
/*  STATUS:  Version 1   Release 1                                   */ 00500000
/*                                                                   */ 00550000
/*  COPYRIGHT:                                                       */ 00600000
/*PROPRIETARY V3 STATEMENT                                           */ 00650000
/*LICENSED MATERIALS - PROPERTY OF IBM                               */ 00700000
/*"RESTRICTED MATERIALS OF IBM"                                      */ 00750000
/*5694-A01                                                           */ 00800000
/*COPYRIGHT 1996,2010  IBM CORP.                               @LCOC */ 00850000
/*END PROPRIETARY V3 STATEMENT                                       */ 00900000
/*                                                                   */ 00950000
/*  FUNCTION: This header file contains the mapping for SMF          */ 01000000
/*            records in the following format:                       */ 01050000
/*                                                                   */ 01100000
/*            ---------------------------------------                */ 01150000
/*            | Type 42 Header                      |                */ 01200000
/*            ---------------------------------------                */ 01250000
/*            | Subtype 7 or 8 or 26 Header         |          @LCOC */ 01300000
/*            ---------------------------------------                */ 01350000
/*            | Product Section                     |                */ 01400000
/*            ---------------------------------------                */ 01450000
/*            | Subtype 7 or 8 or 26 Data           |          @LCOC */ 01500000
/*            ---------------------------------------                */ 01550000
/*            | Client Section                      |                */ 01600000
/*            ---------------------------------------                */ 01650000
/*                                                                   */ 01700000
/*  Note: 1) To obtain address of the subtype 7,8 or 26 header @LCOC */ 01750000
/*           add SMF42LEN to the address of the type 42 header       */ 01800000
/*        2) To obtain the address of the product section            */ 01850000
/*           add SMF42OPS to the address of the type 42 header       */ 01900000
```

Figure 49. SMF C header macro GFSASSMF

```
/*        3) a) To obtain the address of the subtype 7 data portion  */ 01950000
/*              add SMF42NFO to the address of the type 42 header    */ 02000000
/*           b) To obtain the address of the subtype 8 data portion  */ 02050000
/*              add SMF42NUO to the address of the type 42 header    */ 02100000
/*           c) To obtain the address of the subtype 26 data    @LCOA*/ 02116600
/*              add SMF42ADO to the address of the type 42 header    */ 02133300
/*        4) a) To obtain the address of the subtype 7 client        */ 02150000
/*              section add SMF42FLO to the address of the           */ 02200000
/*              current subtype 7 data portion. There can be more    */ 02250000
/*              than one subtype 7 record in a type 42 record.       */ 02300000
/*              There will always be a subtype 7 and client pair.    */ 02350000
/*           b) To obtain the address of the subtype 8 client        */ 02400000
/*              section add SMF42UCO to the address of the           */ 02450000
/*              type 42 header.                                      */ 02500000
/*           c) To obtain the address of the subtype 26 client  @LCOA*/ 02508300
/*              section add SMF42ALO to the address of the           */ 02522200
/*              type 42 header.                                      */ 02536100
/*                                                                   */ 02550000
/*                                                                   */ 02600000
/*  NOTES:                                                           */ 02650000
/*    DEPENDENCIES:  All changes made to this macro must be          */ 02700000
/*                   reflected in GFSAUSMF. If any of the header     */ 02750000
/*                   sections change, they will also have to be      */ 02800000
/*                   reflected in IGWSMF                             */ 02850000
/*                                                                   */ 02900000
/*  MODULE TYPE:   C header/include file                             */ 02950000
/*    PROCESSOR:   IBM ADC/370 compiler                              */ 03000000
/*    LIBRARY:     NFSLB                                             */ 03050000
/*    CC OPTIONS:  None.                                             */ 03100000
/*    ATTRIBUTES:  None.                                             */ 03150000
/*                                                                   */ 03200000
/*    LINKAGE: #include "GFSASsmf.h"                                 */ 03250000
/*                                                                   */ 03300000
/*                                                                   */ 03350000
/*  EXTERNAL REFERENCES:  (external to this header file)             */ 03400000
/*    DATA AREAS:  None                                              */ 03450000
/*    INCLUDE FILES:  None                                           */ 03500000
/*                                                                   */ 03550000
/* TYPEDEFS DEFINED:                                                 */ 03600000
/*       smf42                                                       */ 03650000
/*       smf427h                                                     */ 03700000
/*       smf428h                                                     */ 03750000
/*        smf4226h                                             @LCOA */ 03775000
/*        smf42prd                                                   */ 03800000
/*        smf42s7                                                    */ 03850000
/*        smf42cs                                                    */ 03900000
/*        smf42s8                                                    */ 03950000
/*        smf42s26                                             @LCOA */ 03975000
/*                                                                   */ 04000000
/* STRUCTs DEFINED:                                                  */ 04050000
/*    None                                                           */ 04100000
/*                                                                   */ 04150000
/* MACROs DEFINED:                                                   */ 04200000
/*    None                                                           */ 04250000
/*                                                                   */ 04300000
/*  CHANGE ACTIVITY:                                                 */ 04350000
/* $L33=NFS,HDZ11NP,940425,SJPLTEM: New macro for DFSMS 1.2/NFSS@L33A*/ 04400000
/* $P1=KA90085,HDZ11NP,940623,SJPLTEM: Add File sys & type values@P1A*/ 04450000
/* $P2=KA90127,HDZ11NP,940726,SJPLTEM: Remove HFS block counters @P2A*/ 04500000
/*                                     Convert to 2 name lengths @P2A*/ 04550000
/* $P3=KA90166,HDZ11NP,940825,SJPLTEM: Add hfs pathname support  @P3A*/ 04600000
/* $P4=KA90179,HDZ11NP,940909,SJPLTEM: Define HFS SMF42FSN & FDN @P4A*/ 04650000
/* $LA5=NFS,HDZ11TS,000521,SJPLJST: AIO C Compiler V2.6 support @LA5A*/ 04700000
/* $L75=NFS,HDZ11VS,031209,IBSVKR:  IPv6 support                @L75A*/ 04750000
/* $01=OA10174,HDZ11US,050311,IBSGYG: for IGWSMF compatibility as@01A*/ 04800000
/*                                    IGWSMF contains SMF42 header01A*/ 04850000
/* $LCO=__NFS__,HDZ1C1N,090407,IBSIPI :                         @LCOA*/ 04866600
/*     1. Add SMF type 42 subtype 26 recording                  @LCOA*/ 04883200
/* $P05=KCN0097,HDZ1C1N,091103,IBSVKR:                          @P05A*/ 04888800
/*     1. Delete SMF42ATN field (New Data Set Type)             @P05A*/ 04894400
/*********************************************************************/ 04900000
                                                                        04950000
/*---------------------------------------------*/                       05000000
/* Header for SMF record type 42  as in IGWSMF */              /*@01C*/ 05050000
/*---------------------------------------------*/                       05100000
                                                                        05150000
typedef _Packed struct smf42 {                                          05200000
short int      smf42rcl;    /* Record Length                       */  05250000
short int      smf42sgd;    /* Segment Descriptor (RDW) -- 0 if     */  05300000
                            /* record is not spanned                */  05350000
unsigned char  smf42flg;    /* System indicator flags               */  05400000
char           smf42rty;    /* Record type: 42 (X'2A')              */  05450000
int            smf42tme;    /* Time in hundredths of a second       */  05500000
unsigned char  smf42dte[4]; /* Date record written (by SMF)         */  05550000
unsigned char  smf42sid[4]; /* System identification (by SMF)       */  05600000
unsigned char  smf42ssi[4]; /* Subsystem Id                         */  05650000
short int      smf42sty;    /* Record subtype                       */  05700000
short int      smf42nt;     /* Number of triplets (optional)        */  05750000
short int      _filler1;    /* Reserved (optional)                  */  05800000
                                                                        05850000
/*---------------------------------------------*/                       05900000
/* Product section triplet                     */                       05950000
/*---------------------------------------------*/                       06000000
                                                                        06050000
int            smf42ops;    /* Offset to product section            */  06100000
short int      smf42lps;    /* Length of product section            */  06150000
short int      smf42nps;    /* Number of product sections           */  06200000
                                                                        06250000
```

```
/*---------------------------------------------*/                       06300000
/* Header must end on word boundary            */                       06350000
/*---------------------------------------------*/                       06400000
/* unsigned char  smf42end;     1st data section triplet            */  06450000
} SMF42;                                                                06500000
                                                                        06550000
                                                                        06600000
/* Values for field "smf42flg"                                      */  06650000
#define smf42fsi  0x80 /* When set=subsystem id follows system id   */  06700000
#define smf42fsu  0x40 /* When set = subtypes are used              */  06750000
#define smf42fxa  0x04 /* When set = MVS/XA (SMF enters)            */  06800000
#define smf42fs2  0x02 /* When set = VS2 (SMF enters)               */  06850000
#define smf42fs1  0x01 /* When set = VS1 (SMF enters)               */  06900000
                                                                        06950000
#define smf42_len 0x28                                                  07000000
                                                                        07050000
                                                                        07100000
/*---------------------------------------------------------------*/     07150000
/* SMF42 subtype 7 header section                                */     07200000
/*  (file timeout statistics)                                    */     07250000
/*---------------------------------------------------------------*/     07300000
                                                                        07350000
typedef struct smf427h {                                                07400000
 int             smf42nfo; /*Offset NFSS file timeout stats section */  07450000
 short int       smf42nfl; /*Length NFSS file timeout stats section */  07500000
 short int       smf42nfn; /*Number NFSS file timeout stats section */  07550000
} SMF427H;                                                              07600000
                                                                        07650000
                                                                        07700000
 /*---------------------------------------------------------------*/    07750000
 /* SMF42 subtype 8 header section                                */    07800000
 /*  (user logout statistics )                                    */    07850000
 /*---------------------------------------------------------------*/    07900000
                                                                        07950000
typedef struct smf428h {                                                08000000
 int             smf42nuo;/* Offset NFSS user session stats section */  08050000
 short int       smf42nul;/* Length NFSS user session stats section */  08100000
 short int       smf42nun;/* Number NFSS user session stats section */  08150000
} SMF428H;                                                              08200000
                                                                        08203800
                                                                        08207600
 /*---------------------------------------------------------------*/    08211400
 /* SMF42 subtype 26 header section                              @LCOA*/08215200
 /*  (delete/rename/create statistics)                           @LCOA*/08219000
 /*---------------------------------------------------------------*/    08222800
                                                                        08226600
typedef struct smf4226h {                                      /*@LCOA*/08230400
 int         smf42ado;/* Offset NFSS audit stats section         @LCOA*/08234200
 short int   smf42adl;/* Length NFSS audit stats section         @LCOA*/08238000
 short int   smf42adn;/* Number NFSS audit stats section         @LCOA*/08241800
} SMF4226H;                                                    /*@LCOA*/08245600
                                                                        08250000
                                                                        08300000
/*---------------------------------------------*/                       08350000
/* Product Section                             */                       08400000
/*---------------------------------------------*/                       08450000
                                                                        08500000
typedef struct smf42prd {                                               08550000
 char            smf42pdl[8];  /* Product Level                     */  08600000
 char            smf42pdn[10]; /* Product Name                      */  08650000
 char            smf42psv;     /* Subtype Version Number       @L75A*/  08700000
 char            filler[21];   /* Reserved                     @L75C*/  08750000
} SMF42PRD;                                                             08800000
                                                                        08850000
                                                                        08900000
/*----------------------------------------------- */                    08950000
/* SMF type 42 subtype 7 file timeout Statistics  */                    09000000
/*----------------------------------------------- */                    09050000
                                                                        09100000
typedef _Packed struct smf42s7 {                              /*@P2C*/  09150000
 int             smf42flo;/*Offset to client identification section.*/  09200000
 short int       smf42fll;/*Length of client identification section.*/  09250000
 char            smf42ffs; /* File system type indicator.           */  09300000
                           /* 01 = z/OS UNIX                   @LCOC*/  09350000
                           /* 02 = MVS                          @P1A*/  09400000
 char            smf42fty; /* File type as defined in NFS protocol. */  09450000
                           /*  0 = Non-file                     @P1A*/  09500000
                           /*  1 = Regular file                 @P1A*/  09550000
                           /*  2 = Directory                    @P1A*/  09600000
                           /*  3 = Block special device         @P1A*/  09650000
                           /*  4 = Character special device     @P1A*/  09700000
                           /*  5 = Symbolic link                @P1A*/  09750000
 char            smf42ftm; /* MVS data set type.                    */  09800000
                           /*  0 = unknown MVS file type        @P1A*/  09850000
                           /*  1 = Sequential (BSAM) file       @P1A*/  09900000
                           /*  2 = Partitioned (BPAM)           @P1A*/  09950000
                           /*  3 = Direct Access file           @P1A*/  10000000
                           /*  4 = ISAM is not supported        @P1A*/  10050000
                           /*  5 = Virtual Sequential Access    @P1A*/  10100000
                           /*  6 = VSAM Entry Sequenced         @P1A*/  10150000
                           /*  7 = VSAM Relative Record         @P1A*/  10200000
                           /*  8 = VSAM Keyed access            @P1A*/  10250000
                           /*  9 = dummy index level file block @P1A*/  10300000
                           /* 10 = z/OS UNIX file type         @LCOC*/  10350000
 char            s7fill1[3]; /* Reserved                            */  10400000
 int             smf42fsn; /*File Serial Number,z/OS UNIX INODE@LCOC*/  10450000
 int             smf42fdn; /*Unique device number, z/OS UNIX   @LCOC*/  10500000
                           /*  file system number              @LCOC*/  10550000
 int             smf42fir; /*  Number of I/O blocks read.           */  10600000
 int             smf42fiw; /*  Number of I/O blocks written.        */  10650000
 int             s7fill2;  /*  Reserved                             */  10700000
 u_int64         smf42fbr; /*  Number of bytes read from file. @LA5C*/  10750000
 u_int64         smf42fbw; /*  Number of bytes written to file.@LA5C*/  10800000
 short int       smf42fnl; /* Length of file name.                  */  10850000
 char            s7fill3[6];/* Reserved - Dword boudary         @P2A*/  10900000
```

```
/*-----------------------------------------------*/
/* This is followed by the file name. It is      */
/* a 256 byte character string.            5L75C*/
/*                                         @L75D*/
/* See the S7RECV0 and S7RECV2 types below. @L75C*/
/*                                         @L75D*/
/*-----------------------------------------------*/
} SMF42S7;


/*-----------------------------------------------*/
/* Client Identification Section. Version 0 @L75C*/
/*-----------------------------------------------*/

typedef struct smf42cs0 {                                   /*@L75C*/
 char           smf42cri[8];  /* RACF user ID.                    */
 char           smf42crg[8]; /* RACF group name.                  */
 char           smf42can[8]; /* Account Number.                   */
 int            smf42cui;    /* User ID at client host (UNIX style)*/
 int            smf42cgi;   /* Group ID at client host (UNIX style)*/
 int            smf42cip;    /* IP address of client host.        */
 short  int     smf42chl;    /* Length of client host name        */
 char           smf42chn[256]; /* Client host name.          @L75C*/
} SMF42CS0;                                                 /*@L75C*/


/*-----------------------------------------------*/
/* Client Identification Section. Version 2 @L75A*/
/*-----------------------------------------------*/

typedef struct smf42cs2 {                                    /*@L75A*/
 char           smf42cri[8];   /* RACF user ID.               @L75A*/
 char           smf42crg[8];   /* RACF group name.            @L75A*/
 char           smf42can[8];   /* Account Number.             @L75A*/
 int            smf42cui;      /* User ID at client host      @L75A*/
 int            smf42cgi;      /* Group ID at client host     @L75A*/
 char           smf42cip[16];  /* IPv6 address of client host. @L75A*/
 short  int     smf42chl;      /* Length of client host name   @L75A*/
 char           smf42chn[256]; /* Client host name.           @L75A*/
} SMF42CS2;                                                  /*@L75A*/


/*---------------------------------------------------------- */
/* SMF type 42 subtype 8 user session completion Statistics */
/*---------------------------------------------------------- */

typedef struct smf42s8 {
 int            smf42uco;/* Offset client identification section  */
 short int      smf42ucl; /*Length client identification section  */
 short int      smf42res2; /* Reserved.                           */
 u_int64        smf42ust; /*Session start time (in STCK format)@LA5C*/
 u_int64        smf42uet; /*Session end time (in STCK format)@LA5C*/
 int            smf42uel; /* Session elapsed time (milliseconds)  */
 u_long         smf42unr; /* Number of RPC requests processed in  */
                          /* this session                         */
 int            smf42ute; /* Total elapsed time of all RPC        */
                          /* requests processes in this session.  */
 int            smf42uat; /* Total active time of all RPC         */
                          /* requests processes in this session.  */
 u_int64         smf42urn; /*Number of bytes read in from the)@LA5C*/
                           /* network in this session              */
 u_int64         smf42uwn; /*Number of bytes written out to the@LA5C*/
                           /* network in this session              */
 u_int64         smf42urf; /*Number of bytes read from files on@LA5C*/
                           /* this session                         */
 u_int64         smf42uwf; /*Number of bytes written to files in@LA5C*/
                           /* this session                         */

} SMF42S8;

/*---------------------------------------------------------*/
/* SMF type 42 subtype 26 delete/rename/create Statistics       @LCOA*/
/*---------------------------------------------------------*/

typedef _Packed struct smf42s26                              /*@LCOA*/
{
 int             smf42alo;     /*Offset to client ident section */
 short int       smf42all;     /*Length of client ident section */
 char            smf42aov;     /*NFS protocol version (2,3 or 4)*/
 char            smf42aop;     /*NFS operation/procedure number */
 char            smf42afs;     /* File system type              */
                               /* 01 = UNIX                     */
                               /* 02 = MVS                      */
 char      s26fill1[3];
 union
 {
   struct
   {
     char        smf42atm;     /* Data Set type                */
                               /* 0 = unknown MVS file type    */
                               /* 1 = Sequential (BSAM) file   */
                               /* 2 = Partitioned (BPAM) file  */
                               /* 3 = Direct Access file       */
                               /* 4 = ISAM is not supported    */
                               /* 5 = Virtual Sequential Access*/
                               /* 6 = VSAM Entry Sequenced     */
                               /* 7 = VSAM Relative Record     */
                               /* 8 = VSAM Keyed access        */
                               /* 9 = dummy index level file block*/
     char        s26fill2;
     char        smf42avl[6];  /* Volume name 6 chars */
     int         smf42anl;     /* Data set name length*/
                                                             /*@P05D*/
     char        s26fill3[2];                                /*@P05C*/
     char        smf42avn[6];  /* New Volume name 6 chars */
     int         smf42ann;     /* New Data set name length*/
   } mvs;

   struct
   {
     char        smf42aty;     /* File object type          */
                               /* 1 = Directory             */
                               /* 2 = Character special file*/
                               /* 3 = Regular file          */
                               /* 4 = Named pipe (FIFO) file*/
                               /* 5 = Symbolic link         */
                               /* 6 = Block special file    */
                               /* 7 = Socket file           */
     char        s26fill4[3];                                           14385000
     int         smf42asd;     /* File system device number */          14385700
     int         smf42asl;     /* File system name length */            14386400
     char        smf42asn[44]; /* File system name 1-44 chars */        14387100
                               /* it is not NULL terminated */          14387800
     int         smf42afi;     /* File inode attribute   */             14388500
     char        smf42afa[16]; /* File auditid attribute */             14389200
     int         smf42adi;     /* Directory inode attribute */          14389900
     char        smf42ada[16]; /* Directory auditid attribute */        14390600
     int         smf42afl;     /* File name length */                   14391300
     int         smf42ani;     /* New directory inode    */             14392000
     char        smf42ana[16]; /* New directory audited  */             14392700
     int         smf42afn;     /* New File name length   */             14393400
   } unix;                                                              14394100
 } s26a;                                                                14394800
 char smf42anm[1];             /* File name (1-255 chars)  */           14395500
                               /* and New file name, if exist */        14396200
                               /* It are not NULL terminated */         14396900
} SMF42S26;                                                             14397600
                                                                        14398300
/*********************************************************************/ 14400000
/* the following typedefs are used internally to construct and   @P2A*/ 14450000
/* free SMF type 42 subtype 7 data chaining elements.            @P2A*/ 14500000
/* elements.                                                     @P2A*/ 14550000
/*                                                               @P2A*/ 14600000
                                                                        14650000
 #define SMF_SHORT_FNAME 256     /* short file name <= 256           */ 14700000
 #define SMF_LONG_FNAME 1023     /* long file name > 256             */ 14750000
 #define SMF_MAX_RECLEN 32390    /* maximum SMF record length        */ 14800000
/* Length of file usage stat., plus short file name, plus client data*/ 14850000
 #define SMF_SHORT_DATALEN      \                                       14900000
    (sizeof(SMF42S7)+SMF_SHORT_FNAME+(GLOBAL->flags.IPv6 ?    /*@L75C*/\14950000
     sizeof(SMF42CS2) : sizeof(SMF42CS0)))                    /*@L75A*/ 15000000
 #define SMF_LONG_DATALEN      \                                        15050000
    (sizeof(SMF42S7)+SMF_LONG_FNAME+(GLOBAL->flags.IPv6 ?     /*@L75C*/\15100000
     sizeof(SMF42CS2) : sizeof(SMF42CS0)))                    /*@L75A*/ 15150000
                                                                        15200000
/* S7ELEM is gotten whenever a file times out, to collect statistics    15250000
   on the file.  It is then chained to the list from NFSGLOBAL, and     15300000
   the chain is processed later to write SMF dataset usage record       15350000
   subtype 7.  This is done for files with file name <= 256.         */ 15400000
 typedef _Packed struct s7elem {         /* element to be chained@P2A*/ 15450000
 _Packed struct s7elem  *next;           /* next element in list @P2A*/ 15500000
         char   *filler;                 /* account for C forcing@P2A*/ 15550000
                                         /* doubleword alignment @P2A*/ 15600000
         SMF42S7 smf42s7;                /* data portion         @P2A*/ 15650000
         char    smfilen[SMF_SHORT_FNAME]; /* 256 byte file name @P2A*/ 15700000
         SMF42CS2 smf42cs;               /* client section V2   @L75C*/ 15750000
  } S7ELEM;                              /*                      @P2A*/ 15800000
                                                                        15850000
/* SMF file usage record. Version 0                            @L75C*/  15900000
typedef _Packed struct s7recV0    {     /* type 42 subtype 7   @L75C*/  15950000
        SMF42    smf42;                 /* type 42 header       @P2A*/  16000000
        SMF427H  smf427h;               /* subtype 7 header     @P2A*/  16050000
        SMF42PRD smf42prd;              /* product section      @P2A*/  16100000
        SMF42S7  smf42s7;               /* data portion         @P2A*/  16150000
        char     smfilen[SMF_SHORT_FNAME]; /* 256 byte path name @P2A*/ 16200000
        SMF42CS0 smf42cs0;              /* client section      @L75C*/  16250000
 } S7RECV0;                             /*                     @L75C*/  16300000
                                                                        16350000
```

```
/* SMF file usage record. Version 2                            @L75A*/  16400000
typedef _Packed struct s7recV2 {        /* type 42 subtype 7   @L75A*/  16450000
        SMF42    smf42;                 /* type 42 header      @L75A*/  16500000
        SMF427H  smf427h;               /* subtype 7 header    @L75A*/  16550000
        SMF42PRD smf42prd;              /* product section     @L75A*/  16600000
        SMF42S7  smf42s7;               /* data portion        @L75A*/  16650000
        char     smfilen[SMF_SHORT_FNAME];/* 256 byte path name@L75A*/  16700000
        SMF42CS2 smf42cs2;              /* client section      @L75A*/  16750000
 } S7RECV2;                             /*                     @L75A*/  16800000
                                                                        16850000
/* SMF user session statistics record - subtype 8. Version 0   @L75C*/  16900000
typedef _Packed struct s8recV0   {      /* type 42 subtype 8   @L75C*/  16950000
   SMF42    smf42;                      /* type 42 header       @P2A*/  17000000
   SMF428H  smf428h;                    /* subtype 8 header     @P2A*/  17050000
   SMF42PRD smf42prd;                   /* product section      @P2A*/  17100000
   SMF42S8  smf42s8;                    /* data portion         @P2A*/  17150000
   SMF42CS0 smf42cs0;                   /* client section      @L75C*/  17200000
} S8RECV0;                                                   /*@L75C*/  17250000
                                                                        17300000
/* SMF user session statistics record - subtype 8. Version 2   @L75A*/  17350000
typedef _Packed struct s8recV2   {      /* type 42 subtype 8   @L75A*/  17400000
   SMF42    smf42;                      /* type 42 header      @L75A*/  17450000
   SMF428H  smf428h;                    /* subtype 8 header    @L75A*/  17500000
   SMF42PRD smf42prd;                   /* product section     @L75A*/  17550000
   SMF42S8  smf42s8;                    /* data portion        @L75A*/  17600000
   SMF42CS2 smf42cs2;                   /* client section      @L75A*/  17650000
} S8RECV2;                                                   /*@L75A*/  17700000
                                                                        17750000
/* SMF delete/rename record                                   /*@LCOA*/ 17800000
typedef _Packed struct s26recV2         /* type 42 subtype 26   @LCOA*/ 17833300
{                                       /*                      @LCOA*/ 17866600
        SMF42    smf42;                 /* type 42 header       @LCOA*/ 17900000
        SMF4226H smf4226h;              /* subtype 26 header    @LCOA*/ 17950000
        SMF42PRD smf42prd;              /* product section      @LCOA*/ 18000000
        SMF42S26 smf42s26;              /* data portion         @LCOA*/ 18050000
        SMF42CS2 smf42cs2;              /* client section       @LCOA*/ 18100000
} S26RECV2;                                                   /*@LCOA*/ 18150000
                                                                        18200000
```

SMF assembler header macro GFSAUSMF

Figure 50 contains information about the SMF Assembler header macro GFSAUSMF. The GFSAUSMF macro can be found in the SYS1.NFSMAC library.

```
*/********************************************************************/ 00050000
*/*                                                                  */ 00100000
*/* $MAC(GFSAUSMF) COMP(5695DF121): NFSS XI                          */ 00150000
*/*                                                                  */ 00200000
*/* MACRO NAME: GFSAUSMF.ASM                                         */ 00250000
*/*                                                                  */ 00300000
*/* DESCRIPTION:  Contains ASM Language Mapping of SMF type 42       */ 00350000
*/*               and the subtype 7, 8 and 26 records.         @LCOC */ 00400000
*/*                                                                  */ 00450000
*/* STATUS:  Version 1   Release 1                                   */ 00500000
*/*                                                                  */ 00550000
*/* COPYRIGHT:                                                       */ 00600000
*/*PROPRIETARY V3 STATEMENT                                          */ 00650000
*/*LICENSED MATERIALS - PROPERTY OF IBM                              */ 00700000
*/*"RESTRICTED MATERIALS OF IBM"                                     */ 00750000
*/*5694-A01                                                         */ 00800000
*/*COPYRIGHT 1996,2010  IBM CORP.                               @LCOC */ 00850000
*/*END PROPRIETARY V3 STATEMENT                                      */ 00900000
*/*                                                                  */ 00950000
```

Figure 50. SMF assembler header macro GFSAUSMF

```
*/* FUNCTION: Contains ASM Language Mapping of SMF type 42           */ 01000000
*/*           subtypes 7, 8 and 26 records in the following    @LCOC */ 01050000
*/*           format:                                                */ 01100000
*/*                                                           2@02D  */ 01150000
*/*            ---------------------------------------               */ 01200000
*/*            | Subtype 7 or 8 or 26 Header         |         @LCOC */ 01250000
*/*            ---------------------------------------               */ 01300000
*/*            | Product Section                     |               */ 01350000
*/*            ---------------------------------------               */ 01400000
*/*            | Subtype 7 or 8 or 26 Data           |         @LCOC */ 01450000
*/*            ---------------------------------------               */ 01500000
*/*            | Client Section                      |               */ 01550000
*/*            ---------------------------------------               */ 01600000
*/*                                                                  */ 01650000
*/*  Note: 1) To obtain address of the subtype 7,8 or 26 header @LCOC*/ 01700000
*/*           add SMF42LEN to the address of the type 42 header      */ 01750000
*/*        2) To obtain the address of the product section           */ 01800000
*/*           add SMF42OPS to the address of the type 42 header      */ 01850000
*/*        3) a) To obtain the address of the subtype 7 data portion */ 01900000
*/*              add SMF42NFO to the address of the type 42 header   */ 01950000
*/*           b) To obtain the address of the subtype 8 data portion */ 02000000
*/*              add SMF42NUO to the address of the type 42 header   */ 02050000
*/*           c) To obtain the address of the subtype 26 data   @LCOA*/ 02066600
*/*              add SMF42ADO to the addr of the type 42 header @LCOA*/ 02083300
*/*        4) a) To obtain the address of the subtype 7 client       */ 02100000
*/*              section add SMF42FLO to the address of the          */ 02150000
*/*              current subtype 7 data portion. There can be more   */ 02200000
*/*              than one subtype 7 record in a type 42 record.      */ 02250000
*/*              There will always be a subtype 7 and client pair.   */ 02300000
*/*           b) To obtain the address of the subtype 8 client       */ 02350000
*/*              section add SMF42UCO to the address of the          */ 02400000
*/*              type 42 header.                                     */ 02450000
*/*           c) To obtain the address of the subtype 26 client @LCOA*/ 02458300
*/*              section add SMF42ALO to the address of the     @LCOA*/ 02472200
*/*              type 42 header.                                @LCOA*/ 02486100
*/*                                                                  */ 02500000
*/*                                                                  */ 02550000
*/* NOTES:                                                           */ 02600000
*/*   DEPENDENCIES:  All changes made to this macro must be          */ 02650000
*/*                  reflected in GFSASSMF.                      @02C*/ 02700000
*/*                  All changes made into IGWSMF SMF42 header   @02A*/ 02750000
*/*                  should be reflected in GFSASSMF             @02A*/ 02800000
*/*   USAGE:         To get SMF42 header definitions if needed   @02A*/ 02850000
*/*                  IGWSMF should be used and precede GFSAUSMF: @02A*/ 02900000
*/*                  IGWSMF           | GFSASSMF covers the both @02A*/ 02950000
*/*                  GFSAUSMF         | macros definitions       @02A*/ 03000000
*/*                                                                  */ 03050000
*/* MACRO:                                                           */ 03100000
*/*   PROCESSOR:   High Level Assembler                              */ 03150000
*/*   LIBRARY:     NFSLB                                             */ 03200000
*/*   ATTRIBUTES:  Include                                           */ 03250000
*/*                                                                  */ 03300000
*/*                                                                  */ 03350000
*/* EXTERNAL REFERENCES:  (external to this header file)             */ 03400000
*/*   DATA AREAS:     None                                           */ 03450000
*/*   CONTROL BLOCKS: None                                           */ 03500000
*/*   MACROS:         None                                           */ 03550000
*/*                                                                  */ 03600000
*/* CHANGE ACTIVITY:                                                 */ 03650000
*/*                                                                  */ 03700000
*/*$L33=NFS,HDZ11NP,940310,SJPLTEM: New macro for DFSMS 1.2/NFSS@L33A*/ 03750000
*/*$P1=KA90085,HDZ11NP,940623,SJPLTEM: Add File sys & type values@P1A*/ 03800000
*/*$P2=KA90127,HDZ11NP,940726,SJPLTEM: Remove HFS block counters @P2A*/ 03850000
*/*$P3=KA90179,HDZ11NP,940909,SJPLTEM: Define HFS SMF42FSN & FDN @P3A*/ 03900000
*/*$L75=NFS,HDZ11VS,031209,IBSVKR:  IPv6 support                @L75A*/ 03950000
*/*$01=OA08867,HDZ11US,040712,IBSVKR: Add Sybtype Version Number @01A*/ 04000000
*/*$02=OA10174,HDZ11US,050311,IBSGYG: for IGWSMF compatibility as@02A*/ 04050000
*/*                                   IGWSMF contains SMF42 header02A*/ 04100000
*/*$03=OA15050,HDZ118N,060130,SJPLMTM: Remove SMF42PSV reserved  @03A*/ 04112500
*/*                                    field added from IPv6     @03A*/ 04125000
*/*                                    integration (@L75).       @03A*/ 04137500
*/*$LCO=__NFS__,HDZ1C1N,090407,IBSIPI :                         @LCOA*/ 04141600
*/*    1. Add SMF type 42 subtype 26 recording                  @LCOA*/ 04145700
*/*$P04=KCN0097,HDZ1C1N,091103,IBSVKR:                          @P04A*/ 04147100
*/*    1. Delete SMF42ATN field (New Data Set Type)             @P04A*/ 04148500
*/*******************************************************************/ 04150000
         MACRO                                                          04200000
         GFSAUSMF                                                       04250000
*                                                                       04300000
*                                                                       04350000
*********************************************************************   04400000
*     Header for SMF record type 42 should be used from  IGWSMF   @02C  04450000
*********************************************************************   04500000
*                                                                       04550000
*                                                                       04600000
*  SMF42 header definition is deleted from GFSAUSMF. For reference@02A  04650000
*  the SMF42 header contains the following fields :               @02A  04700000
*                                                                       04750000
*SMF42    DSECT                                                 56@02D  04800000
*SMF42BAS DS    0D        SMF42BAS is the basing expr.                  04850000
*SMF42RCL DS    H         Record Length                                 04900000
*SMF42SGD DS    H         Segment Descriptor (RDW) -- 0 if record is    04950000
**                        not spanned                                   05000000
*SMF42FLG DS    0BL1      System indicator flags                        05050000
*SMF42FSI EQU   X'80'     When set=subsystem id follows system id       05100000
*SMF42FSU EQU   X'40'     When set = subtypes are used                  05150000
*SMF42FXA EQU   X'04'     When set = MVS/XA (SMF enters)                05200000
*SMF42FS2 EQU   X'02'     When set = VS2 (SMF enters)                   05250000
*SMF42FS1 EQU   X'01'     When set = VS1 (SMF enters)                   05300000
*         ORG   SMF42FLG+X'00000001'                                    05350000
*SMF42RTY DS    X         Record type: 42 (X'2A')                       05400000
*SMF42TME DS    FL4       Time in hundredths of a second when record    05450000
*                        was moved to SMF buffer                        05500000
*SMF42DTE DS    CL4       Date record written (by SMF)                  05550000
*SMF42SID DS    CL4       System identification (by SMF)                05600000
*SMF42SSI DS    CL4       Subsystem Id                                  05650000
*SMF42STY DS    HL2       Record subtype                                05700000
*SMF42NT  DS    HL2       Number of triplets (optional)                 05750000
*         DS    HL2       Reserved (optional)                           05800000
**                                                                      05850000
*********************************************************************   05900000
*    Product section triplet                                        *   05950000
*********************************************************************   06000000
**                                                                      06050000
*SMF42OPS DS    FL4       Offset to product section                     06100000
*SMF42LPS DS    HL2       Length of product section                     06150000
*SMF42NPS DS    HL2       Number of product sections                    06200000
**                                                                      06250000
```

```
********************************************************************    06700000
*     Product Section                                              *    06750000
********************************************************************    06800000
**                                                             2@03D    06850000
*SMF42PRD DSECT                                                         07000000
*SMF42PDL DS    CL8       Product Level                                 07050000
*SMF42PDN DS    CL10      Product Name                                  07100000
*SMF42PSV DS    CL1       Subtype Version Number                @01A    07150000
*         DS    CL21      Reserved                              @01C    07200000
**                                                                      07250000
********************************************************************    07300000
*    Product section must end on word boundary                     *    07350000
********************************************************************    07400000
**                                                                      07450000
*SMF42PEN DS    0F                                                      07500000
*SMF42PLN EQU   *-SMF42PRD Length of product section                    07550000
**                                                                      07600000
**    end of header deletion                                    56@02D  07650000
*******************************************************************     07700000
*  SMF42 subtype 7 header section                                 *     07750000
*   (file timeout statistics)                                     *     07800000
*******************************************************************     07850000
*                                                                       07900000
SMF427H  DSECT                                                          07950000
SMF42NFO DS    F         Offset to NFSS file timeout stats section      08000000
SMF42NFL DS    H         Length of NFSS file timeout stats section      08050000
SMF42NFN DS    H         Number of NFSS file timeout stats section      08100000
SMF427HE EQU   *-SMF427H Length of subtype 7 header                     08150000
*                                                                       08200000
*                                                                       08250000
********************************************************************    08300000
*   SMF42 subtype 8 header section                                 *    08350000
*    (user logout statistics )                                     *    08400000
********************************************************************    08450000
*                                                                       08500000
SMF428H  DSECT                                                          08550000
SMF42NUO DS    F         Offset to NFSS user session stats section      08600000
SMF42NUL DS    H         Length of NFSS user session stats section      08650000
SMF42NUN DS    H         Number of NFSS user session stats section      08700000
SMF428HE EQU   *-SMF428H Length of subtype 8 header                     08750000
*                                                                       08800000
*                                                                       08850000
*******************************************************************     08853800
*  SMF42 subtype 26 header section                          @LCOA *     08857600
*  audit (delete/rename/create) statistics                  @LCOA *     08861400
*******************************************************************     08865200
*                                                                       08869000
SMF4226H DSECT                                                          08872800
SMF42ADO DS    F    Offset to NFSS audit stats section          @LCOA   08876600
SMF42ADL DS    H    Length to NFSS audit stats section          @LCOA   08880400
SMF42ADN DS    H    Number to NFSS audit stats section          @LCOA   08884200
SMF4226E EQU   *-SMF4226H Length of subtype 26 header           @LCOA   08888000
*                                                                       08891800
*                                                                       08895600
********************************************************************    08900000
*     SMF type 42 subtype 7 file timeout Statistics                *    08950000
*       (file timeout statistics)                                  *    09000000
********************************************************************    09050000
*                                                                       09100000
```

```
SMF42S7  DSECT                                                          09150000
SMF42FLO DS    F         Offset to client identification section        09200000
SMF42FLL DS    H         Length of client identification section        09250000
SMF42FFS DS    X         File system type indicator                     09300000
*                            01 = z/OS UNIX                   @LCOC     09350000
*                            02 = MVS                          @P1A     09400000
SMF42FTY DS    X         File type as defined in NFS protocol           09450000
*                            00 = Non-file                     @P1A     09500000
*                            01 = Regular file                 @P1A     09550000
*                            02 = Directory                    @P1A     09600000
*                            03 = Block special device         @P1A     09650000
*                            04 = Character special device     @P1A     09700000
*                            05 = Symbolic link                @P1A     09750000
SMF42FTM DS    X         MVS data set type                              09800000
*                             0 = unknown MVS file type        @P1A     09850000
*                             1 = Sequential (BSAM) file       @P1A     09900000
*                             2 = Partitioned (BPAM)           @P1A     09950000
*                             3 = Direct Access file           @P1A     10000000
*                             4 = ISAM is not supported        @P1A     10050000
*                             5 = Virtual Sequential Access    @P1A     10100000
*                             6 = VSAM Entry Sequenced         @P1A     10150000
*                             7 = VSAM Relative Record         @P1A     10200000
*                             8 = VSAM Keyed access            @P1A     10250000
*                             9 = dummy index level file block @P1A     10300000
*                            10 = z/OS UNIX file type         @LCOC     10350000
         DS    XL3       Reserved                                       10400000
SMF42FSN DS    F         File Serial Number, z/OS UNIX INODE  @LCOC     10450000
SMF42FDN DS    F         Unique device number z/OS UNIX FSYS  @LCOC     10500000
SMF42FIR DS    F         Number of I/O blocks read                      10550000
SMF42FIW DS    F         Number of I/O blocks written                   10600000
         DS    F         Reserved                                       10650000
SMF42FBR DS    D         Number of bytes read from file                 10700000
SMF42FBW DS    D         Number of bytes written to file                10750000
SMF42FNL DS    H         Length of file name                            10800000
SMF42FCL DS    0D        C370 ends structure on doubleword boundry      10850000
SMF42FFN DS    0D        File name                        @P2C          10900000
*                                                                       10950000
* The file name is either a 256 byte character string, or a             11000000
* 1023 byte character string. SMF42FNL will contain the length          11050000
*                                                                       11100000
*                        Start of Client Section                        11150000
SMF42F7E EQU   *-SMF42S7 Length of subtype 7 data section               11200000
*                                                                       11250000
SMFSNAME EQU   256       Short file name                                11300000
SMFLNAME EQU   1023      Long file name                                 11350000
*                                                                       11400000
*                                                                       11450000
********************************************************************    11500000
*  Client Identification Section. Version 0                   @L75C*    11550000
********************************************************************    11600000
*                                                                       11650000
SMF42CS0 DSECT                                                 @L75C    11700000
SMF42CRI DS    CL8       RACF user ID                                   11750000
SMF42CRG DS    CL8       RACF group name                                11800000
SMF42CAN DS    CL8       Account Number                                 11850000
SMF42CUI DS    F         User ID at client host (UNIX style)            11900000
SMF42CGI DS    F         Group ID at client host (UNIX style)           11950000
SMF42CIP DS    F         IP address of client host                      12000000
```

```
SMF42CHL DS    H         Length of client host name                     12050000
SMF42CHN DS    CL256     Client host name                      @L75C    12100000
SMF42CSE EQU   *-SMF42CS0 Length of Client Section Version 2   @L75C    12150000
*                                                                       12200000
*                                                                       12250000
********************************************************************    12300000
*  Client Identification Section. Version 2                   @L75A*    12350000
********************************************************************    12400000
*                                                                       12450000
SMF42CS2 DSECT                                                 @L75A    12500000
S2F42CRI DS    CL8       RACF user ID                          @L75A    12550000
S2F42CRG DS    CL8       RACF group name                       @L75A    12600000
S2F42CAN DS    CL8       Account Number                        @L75A    12650000
S2F42CUI DS    F         User ID at client host                @L75A    12700000
S2F42CGI DS    F         Group ID at client host               @L75A    12750000
S2F42CIP DS    CL16      IPv6 address of client host           @L75A    12800000
S2F42CHL DS    H         Length of client host name            @L75A    12850000
S2F42CHN DS    CL256     Client host name                      @L75A    12900000
S2F42CSE EQU   *-SMF42CS2 Length of Client Section Version 2   @L75A    12950000
*                                                                       13000000
*                                                                       13050000
********************************************************************    13100000
*  SMF type 42 subtype 8 user session completion Statistics        *    13150000
*    (user logout statistics )                                     *    13200000
********************************************************************    13250000
*                                                                       13300000
SMF42S8  DSECT                                                          13350000
SMF42UCO DS    F         Offset to client identification section        13400000
SMF42UCL DS    H         Length of client identification section        13450000
         DS    H         Reserved                                       13500000
SMF42UST DS    D         Session start time (in STCK format)            13550000
SMF42UET DS    D         Session end time (in STCK format)              13600000
SMF42UEL DS    F         Session elapsed time (in milliseconds)         13650000
SMF42UNR DS    F         Number of RPC requests processed in this       13700000
*                        session                                        13750000
SMF42UTE DS    F         Total elapsed time of all RPC requests         13800000
*                        processes in this session                      13850000
SMF42UAT DS    F         Total active time of all RPC requests          13900000
*                        processes in this session                      13950000
SMF42URN DS    D         Number of bytes read in from the network in    14000000
*                        this session                                   14050000
SMF42UWN DS    D         Number of bytes written out to the network     14100000
*                        in this session. */                            14150000
SMF42URF DS    D         Number of bytes read from files on this        14200000
*                        session */                                     14250000
SMF42UWF DS    D         Number of bytes written to files in this       14300000
*                        session                                        14350000
SMF42UCS DS    0F        Start of Client Section                        14400000
SMF42S8E EQU   *-SMF42S8 Length of subtype 8 data section               14450000
*                                                                       14450800
********************************************************************    14451600
*     SMF type 42 subtype 26 create/delete/rename Statistics @LCOA *    14452400
********************************************************************    14453200
*                                                                       14454000
SMF42S26 DSECT                                                   @LCOA  14454800
SMF42ALO DS    F       Offset to client identification section   @LCOA  14455600
SMF42ALL DS    H       Length of client identification section   @LCOA  14456400
```

```
SMF42AOV DS    X       NFS protocol version (2,3 or 4)           @LCOA  14457100
SMF42AOP DS    X       NFS operation/procedure number            @LCOA  14457800
SMF42AFS DS    X       File system type z/OS UNIX = 1, MVS = 2   @LCOA  14458500
         DS    XL3     Reserved                                  @LCOA  14459300
*                                                                @LCOA  14459600
*  Legacy MVS audit data  *                                      @LCOA  14460400
*                                                                @LCOA  14461200
SMF42ATM DS    X       Data Set type                             @LCOA  14462000
*                             0 = unknown MVS file type          @LCOA  14462800
*                             1 = Sequential (BSAM) file         @LCOA  14463600
*                             2 = Partitioned (BPAM)file         @LCOA  14464400
*                             3 = Direct Access file             @LCOA  14465200
*                             4 = ISAM is not supported          @LCOA  14466000
*                             5 = Virtual Sequential Access      @LCOA  14466800
*                             6 = VSAM Entry Sequenced           @LCOA  14467600
*                             7 = VSAM Relative Record           @LCOA  14468400
*                             8 = VSAM Keyed access              @LCOA  14469200
*                             9 = dummy index level file block   @LCOA  14470000
         DS    X       Reserved                                  @LCOA  14470800
SMF42AVL DS    CL6     Volume name 6 chars                       @LCOA  14471600
SMF42ANL DS    F       Data set name length                      @LCOA  14472400
                                                                 @P04D  14473400
         DS    XL2     Reserved                                  @P04C  14474400
SMF42AVN DS    CL6     New Volume name 6 chars                   @LCOA  14475600
SMF42ANN DS    F       New Data set name length                  @LCOA  14476400
*                                                                @LCOA  14477200
*    z/OS UNIX audit data  *                                     @LCOA  14478000
*                                                                @LCOA  14478800
         ORG   SMF42ATM                                          @LCOA  14479600
SMF42ATY DS    X       File object type                          @LCOA  14480400
*                      1 = Directory                             @LCOA  14480500
*                      2 = Character special file                @LCOA  14480600
*                      3 = Regular file                          @LCOA  14480700
*                      4 = Named pipe (FIFO) file                @LCOA  14480800
*                      5 = Symbolic link                         @LCOA  14480900
*                      6 = Block special file                    @LCOA  14481000
*                      7 = Socket file                           @LCOA  14481100
         DS    XL3     Reserved                                  @LCOA  14481200
SMF42ASD DS    F       File system device number                 @LCOA  14482000
SMF42ASL DS    F       File system name length                   @LCOA  14482800
SMF42ASN DS    CL44    File system name 1-44 chars               @LCOA  14483600
SMF42AFI DS    F       File inode attribute                      @LCOA  14484400
SMF42AFA DS    CL16    File audited attribute                    @LCOA  14485200
SMF42ADI DS    F       Directory inode attribute                 @LCOA  14486000
SMF42ADA DS    CL16    Directory audited attribute               @LCOA  14486800
SMF42AFL DS    F       File name length                          @LCOA  14487600
*                                                                @LCOA  14488400
SMF42ANI DS    F       New directory inode attribute             @LCOA  14489200
SMF42ANA DS    CL16    New directory audited attribute           @LCOA  14490000
SMF42AFN DS    F       New file name length                      @LCOA  14490800
*                                                                @LCOA  14491600
SMF42ANM DS    CL1     File name and New File name, if exist     @LCOA  14492400
*                      It is not NULL terminated                 @LCOA  14493200
SMF4226E EQU   *-SMF42S26 Length of subtype 26 data section      @LCOA  14494000
         MEND                                                           14500000
```

Appendix K. Capturing diagnostic information using z/OS NFS log data sets and from other components

This topic describes how to use the z/OS NFS log data sets and how to capture diagnostic information for other components including z/OS UNIX, TCP/IP, and AIX.

Using log data sets

This section describes how to use log data sets for the z/OS NFS server and client.

Server log data sets

The z/OS Network File System server records diagnostic information (messages) in the z/OS NFS server log data sets specified in the NFSLOG1 and NFSLOG2 DD statements of the server's startup procedure.

The z/OS NFS server also records messages and debug trace information in a z/OS component trace buffer if specified. Using a component trace buffer can provide performance improvements compared to the log data sets. For details, see "Using z/OS component tracing" on page 297.

NFSLOG1 is associated with the primary log, while NFSLOG2 is associated with the secondary log. When the server is started, the primary log is used first. When the primary log is full, the logging is automatically switched or toggled to the secondary log and a console message is displayed. The last log buffer is lost when the switch takes place. When the secondary log is also full, the logging is switched back to the primary log and the original primary log content is overwritten.

The number of log records depends on the log level setting and the number of transactions that the server handles. Adjust the space allocation of your z/OS NFS server log data sets according to your installation experience to avoid frequent log switching.

Setting up the z/OS NFS server log data set

Here is an example of setting up the z/OS NFS server log data sets:

```
//NFSLOG1   DD   DISP=SHR,DSN=MVSNFS.PROCESS.LOG1
//NFSLOG2   DD   DISP=SHR,DSN=MVSNFS.PROCESS.LOG2
```

File attributes of the z/OS NFS server log data sets can be set up like this:

```
Organization       PS
Record Format      VB
Record Length      137
Block Size         6144
```

See Chapter 11, "Network File System operation," on page 211 and "Log operand" on page 235 to find out how to set the level of messages to be stored in this data set. The default message level is "info" which means that all error, attention, and informational messages are collected and stored in the z/OS NFS server log data set. Reading the message(s) in the z/OS NFS server log data set can help you identify the user correctable error or the problem error which you can report to IBM. See "Server messages" on page 319 for an explanation of the format of the messages that appear in the z/OS NFS server log data sets.

© Copyright IBM Corp. 1991, 2010

This example shows how to identify a sample problem by reading the z/OS NFS server log data set. This example shows a z/OS NFS server log data set with the message level set as “info” (informational):

```
20:25:16 GFSA305I (I) GFSAMAIN ACOPR 01 OPRPARSE: RECEIVED
   COMMAND: EXPORTFS , LENGTH = 9
20:25:16 GFSA865I (E) GFSAMAIN ANEXP 03 DOOPTION: EXPORTS:
   UNEXPECTED OPTION (PO)--EXPORTS FILE UNUSABLE.
20:25:16 GFSA866I (E) GFSAMAIN ANEXP 07 EXPORTAL: EXPORTS:
   DIRECTORY (TCP) WAS NOT EXPORTED.
20:25:16 GFSA925I (E) GFSAMAIN AXOPE 02 OPR_EXPO: ERROR WAS
   DETECTED IN THE EXPORTS FILE. EXPORT LIST NOT REBUILT
20:25:34 GFSA305I (I) GFSAMAIN ACOPR 01 OPRPARSE: RECEIVED
   COMMAND: FLUSHLOG , LENGTH = 9
   ⋮
```

The message GFSA865I in the example is an error message (indicated by the message level E).

```
20:25:16 GFSA865I (E) GFSAMAIN ANEXP 03 DOOPTION: EXPORTS:
   UNEXPECTED OPTION (PO)--EXPORTS FILE UNUSABLE.
```

The message text indicates that the option “PO” specified in the DIRECTORY statement in the exports data set is not a valid option. You could correct this error by changing “PO” to “RO” and restarting the server.

Client log data sets

The z/OS Network File System client records diagnostic information (messages and debug trace information) in the z/OS NFS client log data sets specified in the NFSCMSG1 and NFSCMSG2 DD statements of the client's startup procedure.

NFSCMSG1 is associated with the primary log, while NFSCMSG2 is associated with the secondary log. When the client is started, the primary log is used first. When the primary log is full, the logging is automatically switched or toggled to the secondary log and a console message is displayed. The last log buffer is lost when the switch takes place. When the secondary log is also full, the logging is switched back to the primary log and the original primary log content is overwritten.

The number of log records depends on the log level setting and the number of transactions that the client handles. Adjust the space allocation of your z/OS NFS client log data sets according to your installation experience to avoid frequent log switching.

Setting up the z/OS NFS client log data sets

Here is an example of setting up the z/OS NFS client log data sets:

```
//NFSCMSG1  DD   DISP=SHR,DSN=MVSNFS.CLIENT.LOG1
//NFSCMSG2  DD   DISP=SHR,DSN=MVSNFS.CLIENT.LOG2
```

File attributes of the z/OS NFS client log data sets can be set up like this:

```
Organization       PS
Record Format      VB
Record Length      137
Block Size         6144
```

Reading the message(s) in the z/OS NFS client log data set can help you identify the user correctable error or the problem error which you can report to IBM. See “Client messages” on page 381 for an explanation of the format of the messages that appear in the z/OS NFS client log data sets.

The message GFSC313I in the example is an informational message (indicated by the message level I).

```
13:00:19 GFSC313I (I) CPARS 04 P_CHECK :
   RETRANS OPTION WILL BE IGNORED AS HARD OPTION IS ON.
```

The message text indicates that the keyword, *retrans*, will be ignored because *hard* has been set on. The system programmer should check the mount parameters to make sure that *hard* should be on.

Debug trace data capture

The z/OS Network File System uses trace facilities to record debug trace diagnostic information when a problem requires additional diagnostic information beyond diagnostic messages. Assuming that debug trace diagnostics were not activated at the time of the original failure, the problem must be recreated a second time and the NFS debug trace facilities turned on to capture the diagnostic information. The server provides debug trace diagnostic information from one trace facility. The z/OS Network File System client provides debug trace diagnostic information from one trace facility.

z/OS NFS server debug trace capture

The z/OS NFS server records z/OS NFS debug trace diagnostic information to the z/OS component trace buffer. z/OS NFS error and informational messages are also recorded to the z/OS component trace buffer or to the server log data sets. When recording debug trace diagnostic information, IBM recommends that the z/OS component trace buffer be used to minimize the performance impact of collecting these extra diagnostics. To record z/OS NFS server debug trace diagnostic information in the z/OS component trace buffer, see “Using z/OS component tracing” on page 297.

To record z/OS NFS server debug trace diagnostic information in the CTRACE buffer, use the MODIFY operator command with the LOG=*msgtype* trace option. To start NFS server debug trace recording from the console, enter:

```
MODIFY mvsnfs,LOG=msgtype
```

where *mvsnfs* is the name of the z/OS NFS procedure in the PROCLIB and *msgtype* is ERROR, WARN, or INFO.

The trace records are buffered in storage. The current buffers must be flushed to disk and then the log data sets must be swapped. After swapping, the desired log data set must be copied to disk before it is overwritten by toggling back to the previous log data set by the automatic toggle mechanism.

To flush the remaining z/OS NFS buffers to disk, enter the following console command:

```
MODIFY mvsnfs,FLUSHLOG
```

where *mvsnfs* is the name of the z/OS NFS procedure in the PROCLIB.

This command enables a TSO/E user to browse all the log records that have been written by the Network File System.

To swap z/OS NFS server debug trace recording to the next log data set from the USS command line, issue:

```
MODIFY mvsnfs,SWITCHLOG
```

where *mvsnfs* is the name of the z/OS NFS procedure in the PROCLIB.

To end z/OS NFS Server debug trace recording, enter the following console command:

```
MODIFY mvsnfs,LOG=INFO
```

where *mvsnfs* is the name of the z/OS NFS procedure in the PROCLIB.

To clear (reset) the server log data sets, enter the following console command:

```
MODIFY mvsnfs,INITLOG
```

where *mvsnfs* is the name of the NFS procedure in the PROCLIB.

z/OS NFS server DEBUG trace types

The z/OS NFS server has various trace types which control the amount of trace diagnostic information recorded.

ERROR
Only writes error messages to the log data set. Some informational messages are also written to the console. See Chapter 19, "Network File System messages," on page 319for details.

WARN
Writes both error and warning messages to the NFS server log data set. Some informational messages are also written to the console. See Chapter 19, "Network File System messages," on page 319for details.

INFO Writes error, warning and informational messages to the NFS server log data set. Some informational messages are also written to the console. See Chapter 19, "Network File System messages," on page 319for details.

z/OS NFS client debug trace capture

The z/OS NFS client records z/OS NFS client debug trace diagnostic information to the z/OS component trace buffer. z/OS NFS client error and informational messages are also recorded to the client log data sets. z/OS NFS Client log data set tracing is on by default and cannot be disabled. For additional information on z/OS NFS client component tracing, see "Using z/OS component tracing" on page 297.

The nfsstat UNIX shell command can be used to manage the error and informational message recording to the client log data sets. Superuser authority is required to issue this command. There is no console command control capability for the client log data sets.

In order to see all error and informational messages, the current buffers must be flushed to disk.

To flush the remaining data in the z/OS NFS client buffers to disk, enter:

```
/usr/lpp/NFS/nfsstat -d f
```

Other nfsstat command options are available to swap the current client log data set and clear the contents of the client log data sets:

To swap z/OS NFS client error and informational message recording to the next log data set, issue:

```
/usr/lpp/NFS/nfsstat -d s
```

To clear (reset) the client log data sets from the USS command line, issue:

```
/usr/lpp/NFS/nfsstat -d c
```

Related component trace capture

The z/OS Network File System trace diagnostic information is used in parallel to other z/OS component trace diagnostic information to better understand the system activity for a problem encountered. Depending on the problem, trace diagnostic information may be needed from the z/OS components z/OS UNIX, HFS, or TCP/IP, and for an AIX client or SUN client. In addition, a z/OS dump may be required.

z/OS UNIX System Services activity trace

z/OS UNIX System Services (z/OS UNIX) diagnostic information is normally maintained in a set of cyclical buffers in memory. However, it can also be recorded to a file with a component trace. When it is maintained in memory, a dump must be taken to capture the diagnostic information for debugging purposes.

- To start the z/OS UNIX CTRACE from the console, enter:

  ```
  TRACE CT,4M,COMP=SYSOMVS
  R nn,OPTIONS=(ALL),JOBNAME=(NFS_server),END
  ```

  If you use the JOBNAME keyword on the statement, note that z/OS UNIX uses that parameter for USERID, not for job name. Specify the userid, not the jobname, for this parameter.
- Reproduce the problem in a system with minimal activity.
- Collect dump using console dump (see “z/OS dump” on page 529)
- To turn z/OS UNIX CTRACE off, issue the following console command:

  ```
  TRACE CT,OFF,COMP=SYSOMVS
  ```

z/OS hierarchical file system (HFS) physical file system activity trace

HFS physical file system diagnostic information is normally maintained in a set of cyclical buffers in memory. However, it can also be recorded to a file with a component trace. When it is maintained in memory, a dump must be taken to capture the diagnostic information for debugging purposes.

- To start the HFS CTRACE from the console, enter:

  ```
  TRACE CT,8M,COMP=SYSSMS
  R nn,JOBNAME=(OMVS),OPTIONS=(ENTRY,EXIT,EXITA,SPECIAL,CB,RRTN,COMP=(PFS)),END
  ```
- Reproduce the problem in a system with minimal activity.
- Collect dump using console dump (see“z/OS dump” on page 529)
- To turn HFS CTRACE off, issue the following console command:

  ```
  TRACE CT,OFF,COMP=SYSSMS
  ```

z/OS TCP/IP activity trace

The TCP/IP CTRACE and Packet Trace are normally maintained in a set of cyclical buffers in memory. However, they can also be recorded to a file with a component trace. When they are maintained in memory, a dump must be taken to capture the diagnostic information for debugging purposes.

- To start the TCP/IP CTRACE from the console, enter:

```
TRACE CT,ON,COMP=SYSTCPIP,SUB=(tcpipprocname)
R XX,JOBNAME=(nfsprocname,tcpipprocname)
R XX,OPTIONS=(ENGINE,PFS,SOCKET,INTERNET,TCP,UDP,IOCTL),END
```

- To start the TCP/IP packet trace from the console, enter:

  ```
  TRACE CT,ON,COMP=SYSTCPDA,SUB=(tcpipprocname)
  V TCPIP,tcpipprocname,PKTTRACE,ON,ABBREV=152,IP=xx.xx.xx.xx
  ```

  In this example, *xx.xx.xx.xx* is the client_IP_address.
- Reproduce the problem in a system with minimal activity.
- To collect a dump of TCPIP Dataspace and TCPIP address space, issue the following console command:

  ```
  DUMP COMM=('text')
  R xx,JOBNAME=(tcpipprocname,nfsprocname),DSPNAME=('tcpipprocname'.*),
  SDATA=(ALLNUC,PSA,GRSQ,SUM,CSA,LPA,LSQA,RGN,SWA,SQA,TRT),END
  ```

  See "z/OS dump" on page 529.
- To turn TCP/IP CTRACE off, issue the following console command:

  ```
  TRACE CT,OFF,COMP=SYSTCPIP,SUB=(tcpipprocname)
  V TCPIP,tcpipprocname,PKT,OFF,IP=xx.xx.xx.xx
  ```
- To stop the TCPIP Packet trace, issue the following console command:

  ```
  TRACE CT,OFF,COMP=SYSTCPDA,SUB=(tcpipprocname)
  ```

Note: The first step to collecting traces to the dataspace is to ensure that the *bufsize* in SYS1.PARMLIB member CTIEZB00 is set to at least 10 MB, which is the default value. It may need to be set higher depending on the amount of trace diagnostic information desired, but 10 MB should be used as a minimum. Using the maximum buffer size of 256 MB, you will be sure that the maximum amount of contiguous trace diagnostic information is captured. A 256 MB buffer size is recommended.

TCP/IP will need to be restarted for the change in buffer size to take affect. The trace diagnostic information will be captured using an z/OS dump command that will dump the TCP/IP dataspace named TCPIPDS1. Be aware that this method may result in lost trace diagnostic information as the possibility of wrapping is high. The dump command should be issued soon after the problem occurs.

AIX client activity trace

The AIX IP trace contains a trace of the IP activity from the AIX client:

```
rm /tmp/ibmsupt
```

Note: Make sure that there is enough space under /tmp for the IP-traces.

Systems with high TCP/IP activity may use up to 20 MB per minute.

snap -gkitn (takes 2-3 minutes and collects all necessary AIX diagnostic information).

- To start the AIX IP trace, issue the following AIX command:

  ```
  startsrc -s iptrace -a "-s -d -b
  /tmp/ibmsupt/testcase/iptrc.bin"

  cd /tmp/ibmsupt/testcase
  script cmd_log (puts all keybaord actions to file cmd_log in current directory)
  ```

```
Ping -c 5 (z/OS NFS server)  showmount -e
cd to NFS mounted z/OS filessysem
cp /etc/filesystems /tmp/ibmsupt/testcase/filesystems.out
mount
```

- Reproduce the problem in a system with minimal activity.
- To stop the AIX IP Trace, issue the following AIX command:

  ```
  stopsrc -s iptrace
  ```
- To format the Trace Report, issue the following AIX command:

  ```
  ipreport -rns /tmp/ibmsupt/testcase/iptrc.bin > /tmp/ibmsupt/testcase/iptrc.out
  ```

  Note: The formatted version of the trace will be approximately five times as large as the binary version.
- To stop script recording, issue the following console command:

  ```
  CTRL-D
  ```
- To generate snap.pax.Z file in /tmp/ibmsupt directory, issue the following console command:

  ```
  snap -c
  ```
- Send snap.pax.Z file to NFS service personnel.

SUN client activity trace

The SUN snoop trace contains a trace of the IP activity from the SUN client.

- To start the SUN snoop trace, issue the following SUN command:

  ```
  snoop -o my.trace myclient myserver
  ```
- To stop the SUN snoop Trace, issue the following SUN command:

  ```
  CTRL-C
  ```
- To format Trace Report, issue the following SUN command:

  ```
  snoop -i my.trace -v >my.trace.report
  ```

z/OS dump

A z/OS memory dump contains the current state of the machine. The actual contents of the dump will depend on the address spaces and data spaces selected. The dump will also include any component traces running at the time the dump is taken. IBM generally recommends taking a synchronous dump instead of an asynchronous dump. A synchronous dump ensures that the memory contents do not change between the time the dump request is made and the time the dump is actually taken on the machine.

To collect a console dump of NFS server, NFS client, z/OS UNIX, and TCPIP, issue the following console command:

```
DUMP COMM=(description)

R nn,JOBNAME=(OMVS,TCPproc,NFSSproc,NFSCproc,jobname*),CONT

R nn,DSPNAME=('OMVS'.*,'tcpproc'.*),CONT

R nn, SDATA=(PSA,SQA,LSQA,RGN,TRT,LPA,CSA,GRSQ,SUM,ALLNUC),CONT
```

Appendix L. GFSAMHDJ sample code for creating NFS mount handle data sets and lock data sets

The following sample code (GFSAMHDJ member of SYS1.SAMLIB) shows how to allocate mount handle data sets, copy previous mount handle data sets to the new size needed for the current release, and how to allocate lock data sets.

```
//MVSNFSA  JOB  ,
//    MSGCLASS=A,MSGLEVEL=(1,1),TIME=30,REGION=12M,CLASS=A
//*
//*  FOR MOUNT HANDLE DATABASE DEFINITION, USE:
//*    KEYS(16 0) -
//*  FOR LOCKING DATABASE DEFINITION, USE
//*    KEYS(8 0) -
//*
//***************************************************************
//*
//*
//*  CREATE A VSAM KSDS DATASET FOR THE MOUNT HANDLE DATABASE
//*
//*  THIS STEP IS ONLY REQUIRED IF THE OLD MOUNT HANDLE DATASETS
//*  DO NOT HAVE THE SAME RECORD SIZE DEFINITIONS AS THE CURRENT
//*  DEFINITIONS OR NO MOUNT HANDLE DATASETS EXIST.
//*
//*  REPLACE THE FOLLOWING FIELDS BELOW:
//*
//*     MVSNFS.FHDBASE   WITH DESIRE MOUNT HANDLE DATASET 1 NAME
//*     xxxxxx           WITH VOLUME SERIAL FOR ALLOCATION
//*
//*
//***************************************************************
//DEFINE1 EXEC PGM=IDCAMS,REGION=512K,COND=(EVEN)
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
 DEFINE CLUSTER ( NAME(MVSNFS.FHDBASE) -
            VOL(xxxxxx) -
            CYL (1 1) -
            INDEXED -
            REUSE -
            KEYS(16 0) -
            SHAREOPTIONS(1 3) -
            RECSZ(1700 2000) )
 LISTC ENT(MVSNFS.FHDBASE) ALL
/*
```

Figure 51. Sample code for creating mount handle data sets and lock data sets (Part 1 of 3)

© Copyright IBM Corp. 1991, 2010

```
//**************************************************************
//*
//*
//*  CREATE THE SECOND VSAM KSDS FOR THE MOUNT HANDLE DATABASE
//*         ON A DIFFERENT VOLUME
//*
//*  THIS STEP IS ONLY REQUIRED IF THE OLD MOUNT HANDLE DATASETS
//*  DO NOT HAVE THE SAME RECORD SIZE DEFINITIONS AS THE CURRENT
//*  DEFINITIONS OR NO MOUNT HANDLE DATASETS EXIST.
//*
//*  REPLACE THE FOLLOWING FIELDS BELOW:
//*
//*     MVSNFS.FHDBASE2  WITH DESIRE MOUNT HANDLE DATASET 2 NAME
//*     yyyyyy           WITH VOLUME SERIAL FOR ALLOCATION
//*
//*
//**************************************************************
//DEFINE2 EXEC PGM=IDCAMS,REGION=512K,COND=(EVEN)
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
 DEFINE CLUSTER ( NAME(MVSNFS.FHDBASE2) -
             VOL(yyyyyy) -
             CYL (1 1) -
             INDEXED -
             REUSE -
             KEYS(16 0) -
             SHAREOPTIONS(1 3) -
             RECSZ(1700 2000) )
 LISTC ENT(MVSNFS.FHDBASE2) ALL
/*
//**************************************************************
//*
//*                                                       @01A
//*  COPY OLD SMALLER RECORD SIZE MOUNT HANDLE DATABASES TO THE
//*       NEW LARGER  RECORD SIZE MOUNT HANDLE DATABASES
//*       REQUIRED AS OF Z/OS NFS 1.7
//*
//*  THIS STEP IS ONLY REQUIRED IF THE OLD MOUNT HANDLE DATASETS
//*  DO NOT HAVE THE SAME RECORD SIZE DEFINITIONS AS THE CURRENT
//*  DEFINITIONS.
//*
//*
//*  REPLACE THE FOLLOWING FIELDS BELOW:
//*
//*     MVSNFS1.FHDBASE  WITH OLD MOUNT HANDLE DATASET 1 NAME
//*     MVSNFS1.FHDBASE2 WITH OLD MOUNT HANDLE DATASET 2 NAME
//*     MVSNFS.FHDBASE   WITH NEW MOUNT HANDLE DATASET 1 NAME
//*     MVSNFS.FHDBASE2  WITH NEW MOUNT HANDLE DATASET 2 NAME
//*
//*
//**************************************************************
//IDCAMS1 EXEC PGM=IDCAMS,COND=(EVEN)
//SYSPRINT  DD SYSOUT=*
//OLDFHDB1  DD DSN=MVSNFS1.FHDBASE,DISP=SHR
//OLDFHDB2  DD DSN=MVSNFS1.FHDBASE,DISP=SHR
//NEWFHDB1  DD DSN=MVSNFS.FHDBASE,DISP=SHR
//NEWFHDB2  DD DSN=MVSNFS.FHDBASE,DISP=SHR
//SYSIN     DD *
 REPRO INFILE(OLDFHDB1) OUTFILE(NEWFHDB1) REUSE
 REPRO INFILE(OLDFHDB2) OUTFILE(NEWFHDB2) REUSE
/*
```

Figure 51. Sample code for creating mount handle data sets and lock data sets (Part 2 of 3)

```
//***************************************************************
//*
//*                                                        @01A
//*  CREATE A VSAM KSDS DATASET FOR THE LOCKING DATABASE
//*
//*  THE LOCK DATA SETS MUST ALWAYS BE ALLOCATED EVEN IF NONLM
//*  IS SPECIFIED IN THE SITE ATTRIBUTES.
//*
//*  REPLACE THE FOLLOWING FIELDS BELOW:
//*
//*     MVSNFS.LDBASE    WITH DESIRE LOCK DATASET 1 NAME
//*     xxxxxx           WITH VOLUME SERIAL FOR ALLOCATION
//*
//*
//***************************************************************
//DEFINE3 EXEC PGM=IDCAMS,REGION=512K,COND=(EVEN)
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
 DEFINE CLUSTER ( NAME(MVSNFS.LDBASE) -
             VOL(xxxxxx) -
             CYL (1 1) -
             INDEXED -
             REUSE -
             KEYS(8 0) -
             SHAREOPTIONS(1 3) -
             RECSZ(1700 2000) )
 LISTC ENT(MVSNFS.LDBASE) ALL
/*
//***************************************************************
//*
//*                                                        @01A
//*  CREATE THE SECOND VSAM KSDS FOR THE LOCKING DATABASE
//*         ON A DIFFERENT VOLUME
//*
//*  THE LOCK DATA SETS MUST ALWAYS BE ALLOCATED EVEN IF NONLM
//*  IS SPECIFIED IN THE SITE ATTRIBUTES.
//*
//*  REPLACE THE FOLLOWING FIELDS BELOW:
//*
//*     MVSNFS.LDBASE2   WITH DESIRE LOCK DATASET 2 NAME
//*     yyyyyy           WITH VOLUME SERIAL FOR ALLOCATION
//*
//*
//***************************************************************
//DEFINE4 EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD  SYSOUT=*
//SYSIN DD *
 DEFINE CLUSTER ( NAME(MVSNFS.LDBASE2) -
             VOL(yyyyyy) -
             CYL (1 1) -
             INDEXED -
             REUSE -
             KEYS(8 0) -
             SHAREOPTIONS(1 3) -
             RECSZ(1700 2000) )
 LISTC ENT(MVSNFS.LDBASE2) ALL
/*
//*
```

Figure 51. Sample code for creating mount handle data sets and lock data sets (Part 3 of 3)

Appendix M. Setting up NFS functions with Kerberos Support

The topic gives step-by-step instructions on how to set up Kerberos security for certain NFS functions. The following functions are included:

- "Setting up a Linux Client/Server with NFS Version 4 Kerberos Support"
- "Windows recommendations and limitations" on page 536
- "Setting up a Kerberos Key Distribution Center" on page 537

Setting up a Linux Client/Server with NFS Version 4 Kerberos Support

These instructions are to be used as a guide for setting up a Linux client/server system (Redhat or Suse) with Kerberos support. All setup-related questions should be directed to Suse or Redhat.

Use these steps to configure Redhat Enterprise Linux 5 and Suse 10/11 with NFSv4 and Kerberos support. By default, base NFSv4 support is enabled in the kernel.

1. Edit /etc/hosts. Your fully qualified host name must be the first entry and the machine name's name must not be included on the localhost line. For example:
   ```
   10.1.0.100 hostname.domain.com
   127.0.0.1 localhost.localdomain localhost
   ```
2. Edit /etc/idmapd.conf. Make sure Domain is set to your system's dns domain name.
3. Edit /etc/sysconfig/nfs:

   Redhat:
   ```
   Uncomment/set SECURE_NFS="yes".
   Uncomment/set RPCGSSDARGS="-m".
   ```
   Suse:
   ```
   NFS_SECURITY_GSS="yes"
   NFS4_SUPPORT="yes"
   NFS_START_SERVICES="gssd,idmapd"
   ```
4. Before starting nfs, rpc.gssd, rpc.svcgssd, and rpc.idmapd, set up a keytab file and krb5.conf file. Edit /etc/krb5.conf to match your Kerberos configuration.
5. Unlike other NFSv4 implementations, Linux requires a keytab for the client in order to mount a secure share. This is because the Linux NFS client uses the nfs/hostname.domain credential in the keytab to mount. (This behavior is expected to change once the kernel keyring support is completed.) Create a keytab as documented in *Red Hat Enterprise Linux 5 Deployment Guide* or *Suse Linux Enterprise Server Administration Guide*. Then FTP the keytab in binary mode or (recommended) SCP the keytab to the Linux client and save it to /etc/krb5.keytab.
6. Restart the nfs client/server and related NFS services:

   Redhat:
   ```
   service rpcidmapd restart
   service rpcgssd restart
   service rpcsvcgssd restart
   service nfs restart
   ```
   Suse:
   ```
   rcnfs restart
   rcnfsserver restart
   ```

© Copyright IBM Corp. 1991, 2010

7. Check to see if nfsd, rpc.gssd, rpc,svcgssd and rpc.idmapd start with the ps –A command. If these daemons do not start, check the logs for error messages. You can also enable NFS debugging with the following:
   - NFS client debug command:

     ```
     #rpcdebug –m nfs –s all
     ```

   - NFS server debug command:

     ```
     #rpcdebug –m nfsd –s all
     ```

   Note: To disable NFS debugging issue the following commands:
   - Disable client debugging:

     ```
     #rpcdebug –m nfs –c all
     ```

   - Disable server debugging:

     ```
     #rpcdebug –m nfsd –c all
     ```

   Most problem are related to rpc.gssd and rpcsvcgssd. These daemons can be started from the command line with:

   ```
   #rpc.gssd. –mfvvv&
   #rpcsvcgssd –fvvv&
   ```

   This may provide information on why they did not start.

   Note: This debug level is very noisy. The output is sent to /var/log/messages.

8. Special considerations for Linux Clients:
   - Because Linux has the keytab, the system is able to perform secure NFS mount without having the credentials acquired by the "kinit" command. This behavior will change once kernel keyring support is completed.
   - Root user (uid = 0) uses Linux machine credentials, but not the regular User credentials obtained with kinit. Thus root user will be able to browse the NFS mount point without performing kinit. Regular users will need "kinit" to access the mount points. This behavior will change once kernel keyring support is completed.
   - Kdestroy will not destroy the context in the Linux kernel. This behavior will change once kernel keyring support is completed.
   - Extra configuration is needed for Linux remote realm setup since Linux sends nfs/hostname.domain instead of user principal during mount time. If the Linux's NFS principal is not defined to RACF, z/OS NFS server will reject mount requests. A simple way to solve this problem is to map the entire remote realm to RACF. Another more secure way to work around this is to map individual Linux machines to a special realm in the [domain_realm] section in the /etc/skrb/krb5.conf, and map that realm to a special RACF USER on the z/OS NFS server thus leaving all other machines in the remote realm intact.

Windows recommendations and limitations

For Windows users, it is recommended to use Connectivity Kerberos or MIT Leash Kerberos Ticket Manager with Hummingbird Maestro NFS client. For more information on Hummingbird NFS Maestro client and the configurations of the Kerberos on Windows platform, see the Hummingbird NFS Maestro User's Guide. If the z/OS NFS server security site attribute is set to SAF or SAFEXP, the only supported authentication level on Windows platform is authsys. RPCSEC_GSS authentication levels, such as krb5, krb5i, and krb5p, are not supported because Hummingbird NFS Maestro 10 and higher requires NIS/NIS+ or LDAP with Kerberos instead of PCNFSD, which serves as the mvslogin tool for Windows platform. This restriction has been lifted with the addition of mvslogin elimination with RPC_SECGSS mounts in z/OS NFS V1R11

Setting up a Kerberos Key Distribution Center

In order to start a z/OS NFS server with Kerberos authentication features, a Kerberos Key Distribution Center must be ready before the z/OS NFS server starts. This section lists the basic steps involved in setting up the z/OS KDC which will be compatible with the z/OS NFS server environment. For more advanced configurations and detailed explanations of the setup steps and the reasoning behind, please refer to *z/OS Integrated Security Services Network Authentication Service Administration*. For other platforms'KDC setups, please consult your vendor specific documentation

These steps assume that Resource Access Control Facility (RACF) is available in the system. If you have a different but equivalent external security manager, please refer to the documentation of the product for instructions.

1. Copy the SKRBKDC started task procedure from EUVF.SEUVFSAM to SYS1.PROCLIB. The SYS1.PROCLIB(SKRBKDC) should look like the following:

```
//*********************************************************************
//*
//* Procedure for starting the Kerberos SKRBKDC started task
//* Specify PARMS='-kdc' to enable the Kerberos KDC services.
//* Specify PARMS='-nokdc' to disable the Kerberos KDC services.
//*
//*********************************************************************
//SKRBKDC  PROC  REGSIZE=256M,OUTCLASS='A',PARMS='-kdc'
//*--------------------------------------------------------------------
//GO       EXEC  PGM=EUVFSKDC,REGION=&REGSIZE,TIME=1440,
//  PARM=('ENVAR("LANG=En_US.IBM-1047"),TERM(DUMP) / &PARMS      X
//             1>DD:STDOUT 2>DD:STDERR')
//STDOUT   DD  SYSOUT=&OUTCLASS,DCB=LRECL=250,
//  FREE=END,SPIN=UNALLOC
//STDERR   DD  SYSOUT=&OUTCLASS,DCB=LRECL=250,
//  FREE=END,SPIN=UNALLOC
//SYSOUT   DD  SYSOUT=&OUTCLASS,
//  FREE=END,SPIN=UNALLOC
//CEEDUMP  DD  SYSOUT=&OUTCLASS,
//  FREE=END,SPIN=UNALLOC
```

2. Copy the sample Kerberos configuration file in z/OS UNIX from /usr/lpp/skrb/examples/krb5.conf to /etc/skrb/krb5.conf. The permission bits of this file should allow only the administrator to modify it but everyone else to be able to read.

   Note: If the z/OS NFS server will be supporting multiple platforms of NFS clients, IBM recommends using des-cbc-crc encryption types only for compatibility, as shown in the copy below.

```
;---------------------------------------------------------------------;
;  Sample Kerberos configuration file                                 ;
;  ;                                                                  ;
;  Copy this file to /etc/skrb/krb5.conf and then tailor it for       ;
;  your Kerberos configuration                                        ;
;  ;                                                                  ;
;  Do not enable DES3 encryption unless all of the systems in the     ;
;  realm have DES3 support.  In order to use DES3 encryption for      ;
;  tickets, you must set the SKDC_TKT_ENCTYPES environment variable   ;
;  in /etc/skrb/home/kdc/envar.                                       ;
;---------------------------------------------------------------------;

[libdefaults]
```

```
default_realm = KRB390.IBM.COM
kdc_default_options = 0x40000010
use_dns_lookup = 0
; Default encryption types if DES3 is not supported
default_tkt_enctypes = des-cbc-crc
default_tgs_enctypes = des-cbc-crc

[realms]
KRB390.IBM.COM = {
kdc = dcesec4.krb390.ibm.com:88
kpasswd_server = dcesec4.krb390.ibm.com:464
admin_server = dcesec4.krb390.ibm.com:749
}
KRB2000.IBM.COM = {
kdc = sstone1.krb2000.ibm.com:88
admin_server = sstone1.krb2000.ibm.com:749
}

[domain_realm]
.krb390.ibm.com = KRB390.IBM.COM
.krb2000.ibm.com = KRB2000.IBM.COM
```

3. Copy the environment variable definitions from /usr/lpp/skrb/examples/skrbkdc.envar to /etc/skrb/home/kdc/envar. Depending on which type of KDC is being set up, the environment variable SKDC_DATABASE should be set to SAF or NDBM accordingly (default is set to SAF registry type KDC). The file permissions should allow only the administrator to read and update.
4. Add the path "PATH=/usr/lpp/skrb/bin:$PATH" in the z/OS UNIX to the user's ".profile" and export the "PATH".
5. Issue the following RACF commands (entering each command on a single line). If needed, consult *z/OS Integrated Security Services Network Authentication Service Administration* for explanations.

   ```
   RDEFINE FACILITY IRR.RUSERMAP UACC(read)
   SETROPTS RACLIST(FACILITY) REFRESH
   AU skrbkdc DFLTGRP(sys1) NOPASSWORD OMVS(UID(0) PROGRAM('/bin/sh')
          HOME('/etc/skrb/home/kdc'))
   RDEFINE REALM KERBDFLT KERB(KERBNAME(KRB390.IBM.COM)
          PASSWORD(password) MINTKTLFE(15) DEFTKTLFE(36000)
          MAXTKTLFE(86400))
   ```

   Note: KERBNAME(KRB390.IBM.COM) should be changed to match your Kerberos Realm.
6. For SAF registry KDC, continue with the following steps. For NDBM registry KDC, skip to step 7 on page 539.
   a. Issue the following RACF commands, entering each command on a single line:

      ```
      SETROPTS CLASSACT(appl) RACLIST(appl)
      RDEFINE APPL SKRBKDC UACC(read)
      SETROPTS CLASSACT(ptktdata) RACLIST(ptktdata)
      RDEFINE PTKTDATA SKRBKDC UACC(none) SSIGNON(keymasked(3734343237343131))
      SETROPTS RACLIST(appl ptktdata) REFRESH
      SETROPTS GENERIC(started)
      RDEFINE STARTED SKRBKDC.** STDATA(user(skrbkdc))
      RDEFINE STARTED SKRBWTR.** STDATA(user(skrbkdc))
      SETROPTS RACLIST(started) REFRESH
      AU KADMIN DFLTGRP(sys1) PASSWORD(password)KERB(KERBNAME(kadmin/admin))
      ALU KADMIN PASSWORD(password) NOEXPIRED
      PASSWORD KADMIN NOINTERVAL
      AU CHANGEPW DFLTGRP(sys1) PASSWORD(password)
      KERB(kerbname(kadmin/changepw))
      ALU CHANGEPW PASSWORD(password) NOEXPIRED
      PASSWORD CHANGEPW NOINTERVAL
      ```

b. For multiple Kerberos realms environment, create ticket-granting tickets with RACF commands:

```
RDEFINE REALM /.../KRB390.IBM.COM/KRBTGT/KRB2000.IBM.COM/
      KERB(PASSWORD(password))
RDEFINE REALM /.../KRB2000.IBM.COM/KRBTGT/KRB390.IBM.COM/
      KERB(PASSWORD(password))
```

Notes:

1) The /.../ and trailing slash are required.
2) KRB390.IBM.COM is the local realm. Change to match the realm of your KDC.
3) KRB2000.IBM.COM is the remote realm, Change to match the name of the remote realm.

c. Add Kerberos segments to existing user definitions. These Kerberos segments serve as the Kerberos principals in the Kerberos database.

To add a RACF userid, issue RACF command, for example:

```
AU (userid1) OWNER (IBMUSER) OMVS(UID(101))
```

To define Kerberos segment to this user definition, issue RACF command:

```
ALTUSER userid1 PASSWORD(password) NOEXPIRED
      KERB(KERBNAME(user1))
```

Note: The version value (-v option) and password need to match the version and password used when creating the NFS principal.

d. Start the skrbkdc task.

e. Add Kerberos segments to existing user definitions. These Kerberos segments serve as the Kerberos principals in the Kerberos database.

To add a RACF id, issue RACF command, for example:

```
AU (RACFID1) OWNER (IBMUSER) OMVS(UID(101))
```

To define Kerberos segment to this user definition, issue RACF command:

```
ALTUSER RACFID1 PASSWORD(password) NOEXPIRED  KERB(KERBNAME(user1))
```

f. Start the skrbkdc task.

g. Continue to step 8 on page 540 to complete KDC setup.

7. For NDBM registry type KDC, follow these steps.

a. To create initial registry database files, issue z/OS UNIX command:

```
kdb5_ndbm create
```

IBMUSER and IBMUSER/admin user principals are now created with initial password of IBMUSER.

b. Copy sample KDC configuration file from /usr/lpp/skrb/examples/kdc.conf to /etc/skrb/home/kdc/kdc.conf and set the values inside as needed or leave them to default values.

c. Copy the sample administration access control file from /usr/lpp/skrb/examples/kadm5.acl to /etc/skrb/home/kdc/kadm5.acl . The administrator can choose to customize it or leave it as default.

d. Start the skrbkdc task.

e. Create NFS principal for the z/OS NFS server using the kadmin interface in z/OS UNIX.

To enter the kadmin interface, issue z/OS UNIX command:

```
kadmin -p IBMUSER/admin -w IBMUSER
```

To create NFS principle, enter:

```
kadmin> addprinc nfs/host.domain
```

f. For multiple Kerberos realm environment, create ticket-granting tickets in kadmin interface:

```
kadmin> addprinc -e des-cbc-crc:normal krbtgt/KRB390.IBM.COM@KRB2000.IBM.COM
kadmin> addprinc -e des-cbc-crc:normal krbtgt/ KRB2000.IBM.COM@KRB390.IBM.COM
```

Note: The “-e des-cbc-crc:normal” should be used in order to support various UNIX platforms. The passwords specified for these two principals should be the same.

g. Create keytab for the z/OS NFS server by issuing in kadmin interface:

```
kadmin> ktadd -e des-cbc-crc:normal -k /path/temp.keytab nfs/host.domain
```

Note: : The "-e des-cbc-crc:normal" should be used in order to support various UNIX platforms. The passwords specified for these two principals should be the same.

h. Add Kerberos principals into the Kerberos database through the kadmin interface:

```
kadmin> addprinc user1
kadmin> addprinc user2
```

8. Put (or ftp in binary mode if different hosts) the keytab on z/OS NFS server’s /etc/skrb directory and rename it to /etc/skrb/krb5.keytab, in order to be able to start the z/OS NFS server later.

Appendix N. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I*for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

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

© Copyright IBM Corp. 1991, 2010

Appendix O. Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are

© Copyright IBM Corp. 1991, 2010

optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.
- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

  Notes:

  1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
  2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
  3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.
- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

© Copyright IBM Corp. 1991, 2010

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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

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

Programming interface information

This document documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS NFS.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Trademarks

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

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

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

Glossary

This glossary includes terms and definitions for Network File System (NFS).

The following cross-references are used in this glossary:

1. *See* refers the reader from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
2. *See also* refers the reader to a related or contrasting term.

To view glossaries for other IBM products, go to http://www-306.ibm.com/software/globalization/terminology/

A

access method. A technique for moving data between main storage and input/output devices.

access permission. A group of designations that determine the users who can access a particular file and how the users can access the file.

ACS. See automatic class selection.

address. A unique code or identifier for a register, device, workstation, system, or storage location.

address space. The range of addresses available to a computer program or process. Address space can refer to physical storage, virtual storage, or both.

alias. An alternative name for an integrated catalog facility (ICF) user catalog, a file that is not a Virtual Storage Access Method (VSAM) file, or a member of a partitioned data set (PDS) or a partitioned data set extended (PDSE).

alias entry. The correlation of an alias with the physical entry name of a user catalog or a data set that is not a Virtual Storage Access Method (VSAM) data set.

allocation. The process of temporarily connecting a program to a data set, file, or device.

American Standard Code for Information Interchange (ASCII). A standard code used for information exchange among data processing systems, data communication systems, and associated equipment. ASCII uses a coded character set consisting of 7-bit coded characters. See also Extended Binary Coded Decimal Interchange Code.

APAR. See authorized program analysis report.

APF. See authorized program facility.

API. See application programming interface.

application programming interface (API). An interface that allows an application program that is written in a high-level language to use specific data or functions of the operating system or another program.

ASCII. See American Standard Code for Information Interchange. See also Extended Binary Coded Decimal Interchange Code.

automatic class selection (ACS). A mechanism for assigning storage management subsystem (SMS) classes and storage groups to data sets. The storage administrator is responsible for establishing ACS routines appropriate for an installation's storage requirements.

automatic class selection routine (ACS routine). A procedural set of automatic class selection (ACS) language statements. Based on a set of input variables, the ACS routine generates, for a data set, the name of a predefined storage management subsystem (SMS) class or a list of names of predefined storage groups.

authorized program analysis report (APAR). A request for correction of a defect in a current release of an IBM-supplied program.

authorized program facility (APF). In a z/OS environment, a facility that permits the identification of programs that are authorized to use restricted functions.

B

basic sequential access method (BSAM). An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or a direct access device.

BIOD. The caching daemon that caches directory lookups and file data when remote files are accessed from the host.

block. A string of data elements recorded, processed, or transmitted as a unit. The elements can be characters, words, or physical records.

BSAM. See basic sequential access method.

© Copyright IBM Corp. 1991, 2010

C

CCSID. See coded character set identifier.

CDRA. See Character Data Representation Architecture.

Character Data Representation Architecture (CDRA). An IBM architecture that defines a set of identifiers, resources, services, and conventions to achieve consistent representation, processing, and interchange of graphic character data in heterogeneous environments.

client. A software program or computer that requests services from a server. See also server, host.

client/server. Pertaining to the model of interaction in distributed data processing in which a program on one computer sends a request to a program on another computer and awaits a response. The requesting program is called a client; the answering program is called a server.

coded character set identifier (CCSID). A 16-bit number that includes a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other information that uniquely identifies the coded graphic-character representation.

credential. Detailed information, acquired during authentication, that describes the user, any group associations, and other security-related identity attributes. Credentials can be used to perform a multitude of services, such as authorization, auditing, and delegation. For example, the sign-on information (user ID and password) for a user are credentials that allow the user to access an account.

current directory. See working directory.

D

daemon. A program that runs unattended to perform continuous or periodic functions, such as network control.

DASD. See direct access storage device.

DASD volume. A direct access storage device (DASD) space identified by a common label and accessed by a set of related addresses. See also primary storage.

data control block (DCB). A control block used by access method routines in storing and retrieving data.

Data Encryption Standard (DES). A cryptographic algorithm designed to encrypt and decrypt data using a private key.

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access. See also file.

data set control block (DSCB). A control block in the volume table of contents (VTOC) that describes data sets.

data set organization (DSORG). The type of arrangement of data in a data set, such as sequential organization or partitioned organization.

DBCS. See double-byte character set.

DCB. See data control block.

DES. See Data Encryption Standard.

DES authentication. A type of encryption algorithm that requires a client to send credentials (name, conversation key, window key, and a time stamp) to the server. The server then returns a verifier to the client. Data Encryption Standard (DES) credentials are sometimes called secure credentials because they are based on a sender's ability to encrypt data using a common time reference; a randomly generated key is required to encrypt a common reference time that is then used to create a conversation key.

DFSMS (Data Facility Storage Management Subsystem). An operating environment that helps automate and centralize the management of storage. To manage storage, the storage management subsystem (SMS) provides the storage administrator with control over data class, storage class, management class, storage group, and automatic class selection (ACS) routine definitions.

DFSMSdfp. A DFSMS functional component and a base element of z/OS that provides functions for storage management, data management, device management, and distributed data access.

direct access. A file access method allowing reading and writing of records in an arbitrary order.

direct access storage device (DASD). A device that allows storage to be directly accessed, such as a disk drive.

direct data set. A data set that has records in random order on a direct access volume. Each record is stored or retrieved according to its actual address or its address relative to the beginning of the data set. See also sequential data set.

directory. In UNIX, a file that maps the names of other directories and files to their locations.

double-byte character set (DBCS). A set of characters in which each character is represented by two bytes. These character sets are commonly used by national languages, such as Japanese and Chinese, that have more symbols than can be represented by a single byte.

DSCB. See data set control block.

E

EBCDIC. See Extended Binary Coded Decimal Interchange Code. See also American Standard Code for Information Interchange.

entry-sequenced data set (ESDS). A data set whose records are loaded without respect to their contents, and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

ESDS. See entry-sequenced data set.

exports data set. In z/OS, an MVS file on the server containing entries for directories that can be exported to Network File System (NFS) clients. It is used by the server to determine which MVS files and prefixes can be mounted by a client, and to write-protect MVS files on the server.

Extended Binary Coded Decimal Interchange Code (EBCDIC). A coded character set of 256 8-bit characters developed for the representation of textual data. See also American Standard Code for Information Interchange.

External Data Representation (XDR). A standard developed by Sun Microsystems, Incorporated to represent data in machine-independent format. Because XDR is a vendor-independent method for representing the data, new computer architectures can be integrated into the network without requiring the updating of translation routines.

F

file. A collection of related data that is stored and retrieved by an assigned name. See also data set.

file handle. A number that is used by the client and server sides of the Network File System (NFS) to specify a particular file or prefix.

file system. The collection of files and file management structures on a physical or logical mass storage device, such as a diskette or minidisk.

File Transfer Protocol (FTP). In the Internet suite of protocols, an application layer protocol that uses TCP and Telnet services to transfer bulk-data files between machines or hosts.

FMID. See function modification identifier.

FTP. See File Transfer Protocol.

function modification identifier (FMID). With SMP/E, a code that identifies the release levels of a program product.

G

gateway. A device or program used to connect networks or systems with different network architectures.

GID. See group ID.

group. (1) With respect to partitioned data sets (PDSs), a member and the member's aliases that exist in a PDS or partitioned data set extended (PDSE), or in an unloaded PDSE. (2) A collection of users who can share access authorities for protected resources.

group ID (GID). In the UNIX operating system, an integer that uniquely identifies each group of users to the operating system.

H

handle. A character string that represents an object, and is used to retrieve the object.

HFS data set. See hierarchical file system data set.

hierarchical file system data set (HFS data set). A data set that contains a particular type of file system that is compliant with the Portable Operating System Interface (POSIX). An HFS data set is a collection of files and directories organized in a hierarchical structure that can be accessed using z/OS UNIX System Services (z/OS UNIX).

host. A computer that is connected to a network and provides an access point to that network. The host can be a client, a server, or both a client and server simultaneously. See also server, client.

I

IDCAMS. An IBM program that is used to process access method services commands. It can be invoked as a job or jobstep, from a TSO terminal or from within a user's application program.

Interactive System Productivity Facility (ISPF). An IBM licensed program that serves as a full-screen editor and dialog manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogs between the application programmer and terminal user. See also Time Sharing Option.

Internet. The worldwide collection of interconnected networks that use the Internet suite of protocols and permit public access.

Internet Protocol (IP). A protocol that routes data through a network or interconnected networks. This protocol acts as an intermediary between the higher protocol layers and the physical network. See also Transmission Control Protocol.

interprocess communication (IPC). The process by which programs send messages to each other. Sockets, semaphores, signals, and internal message queues are common methods of interprocess communication.

IP. See Internet Protocol. See also Transmission Control Protocol.

IPC. See interprocess communication.

J

JCL. See job control language.

job control language (JCL). A command language that identifies a job to an operating system and describes the job's requirements.

K

key-sequenced data set (KSDS). A VSAM file or data set whose records are loaded in key sequence and controlled by an index.

KSDS. See key-sequenced data set.

L

library. A partitioned data set or a series of concatenated partitioned data sets. See also partitioned data set extended.

local host. The computer to which a user's terminal is directly connected.

M

management class. A user-defined schedule for moving objects from one storage class to the next. Management class describes the retention and class transition characteristics for a group of objects in a storage hierarchy.

master catalog. A key-sequenced data set (KSDS) or file with an index containing extensive data set and volume information that the Virtual Storage Access Method (VSAM) requires to locate data sets or files, allocate and deallocate storage space, verify the authorization of a program or operator to gain access to a data set or file, and accumulate usage statistics for data sets or files.

maximum transmission unit (MTU). The largest possible unit of data that can be sent on a given physical medium in a single frame. For example, the maximum transmission unit for Ethernet is 1500 bytes.

MBCS. See multibyte character set. See also double-byte character set, single-byte character set, Unicode.

mount. To place a data medium in a position to operate.

mount handle data set. In z/OS, a data set used to store the file handles of Network File System (NFS) mount points.

mount point. (1) A directory established in a workstation or a server local directory that is used during the transparent accessing of a remote file. (2) In Linux operating systems and in UNIX operating systems such as AIX, the directory at which a file system is mounted and under which other file systems may be mounted.

MTU. See maximum transmission unit.

multibyte character set (MBCS). A character set that represents single characters with more than a single byte. See also double-byte character set, single-byte character set, Unicode.

Multiple Virtual Storage (MVS). An IBM operating system that accesses multiple address spaces in virtual storage.

MVS. See Multiple Virtual Storage.

N

network. In data communication, a configuration in which two or more locations are physically connected for the purpose of exchanging data.

Network Lock Manager (NLM). A service used by Network File System (NFS) when using version 2 or 3 of the NFS protocol that allows a client on the host to lock a range of bytes or an entire file on the NFS server.

Network Status Manager (NSM). A service used by Network File System (NFS) when using version 2 or 3 of the NFS protocol to determine whether resources, such as file open share or byte range locks, are still in use by a remote client.

NLM. See Network Lock Manager.

NSM. See Network Status Manager.

null credential. A type of credential that is usually associated with diskless workstations. Because there is no repository of information that is local to the workstation, it is not possible to obtain identifying information.

O

object. A directory or file.

P

partitioned data set (PDS). A data set on direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data. See also sequential data set.

partitioned data set extended (PDSE). A system-managed data set that contains an indexed directory and members that are similar to the directory and members of partitioned data sets (PDSs). See also library.

PDS. See partitioned data set. See also sequential data set.

PDS directory. A set of records in a partitioned data set (PDS) that is used to relate member names to their locations within the data set.

PDSE. See partitioned data set extended. See also library.

permission code. A 3-digit octal code or a nine-letter alphabetic code that indicates the access permission for a UNIX file. The access permissions are read, write, and execute.

permission field. One of the 3-character fields within the permissions column of a UNIX directory listing. The permission field indicates the read, write, and run permissions for the file or directory owner and for the group. It is used by file systems to control access.

PFS. See physical file system.

physical file system (PFS). The part of the operating system that handles the actual storage and manipulation of data on a storage medium.

port. An end point for communication between applications, generally referring to a logical connection. A port provides queues for sending and receiving data. Each port has a port number for identification.

Portable Operating System Interface (POSIX). An IEEE family of standards designed to provide portability between operating systems that are based on UNIX. POSIX describes a wide spectrum of operating-system components ranging from C language and shell interfaces to system administration

Portmapper. A program that maps client programs to the port numbers of server programs. A portmapper is used with remote procedure call (RPC) programs. Portmapper does not support IPv6. RPCBIND is required for IPv6. See also RPCBIND.

port number. The part of a socket address that identifies a port within a host.

POSIX. See Portable Operating System Interface.

primary storage. A direct access storage device (DASD) volume available to users for data allocation. The volumes in primary storage are called primary volumes.

primary volume. A volume managed by DFSMShsm containing data sets that are directly accessible to the user. See also primary storage.

program temporary fix (PTF). For System i, System p, and System z products, a fix that is tested by IBM and is made available to all customers.

protocol. A set of rules controlling the communication and transfer of data between two or more devices or systems in a communication network.

PTF. See program temporary fix.

Q

QSAM. See queued sequential access method.

queued sequential access method (QSAM). An access method for storing and retrieving logical records in a continuous sequence. Input data blocks awaiting processing or output data blocks awaiting transfer to auxiliary storage are queued on the system to minimize delays in I/O operations.

R

RACF. See Resource Access Control Facility.

relative record data set (RRDS). A type of Virtual Storage Access Method (VSAM) data set whose records have fixed or variable lengths, and are accessed by relative record number.

Remote Procedure Call (RPC). A protocol that allows a program on a client computer to run a program on a server.

Resource Access Control Facility (RACF). An IBM licensed program that provides access control by identifying users to the system; verifying users of the system; authorizing access to protected resources; logging unauthorized attempts to enter the system; and logging accesses to protected resources.

Resource Measurement Facility (RMF). A feature of z/OS that measures selected areas of system activity and presents the data collected in the format of printed reports, System Management Facility (SMF) records, or display reports.

RMF. See Resource Measurement Facility.

root. (1) The UNIX definition for a directory that is the base for all other directories. (2) The user name for the system user with the most authority.

root user. A system user who operates without restrictions. A root user has the special rights and privileges needed to perform administrative tasks.

RPC. See Remote Procedure Call.

RPCBIND. A program that maps client programs to the port numbers of server programs. RPCBIND is used with remote procedure call (RPC) programs. RPCBIND is required for IPv6. See also Portmapper.

RRDS. See relative record data set.

S

SAF. See System Authorization Facility.

SDSF. See System Display and Search Facility.

sequential file. A type of MVS file that has its records stored and retrieved according to their physical order within the file. It must be on a direct access volume.

sequential data set. A data set whose records are organized on the basis of their successive physical positions, such as on magnetic tape. See also partitioned data set, direct data set.

server. A software program or a computer that provides services to other software programs or other computers. See also host, client.

sharing. Using a file on a remote system. Sharing is performed by mounting the remote file system and then reading or writing files in that remote system.

single-byte character set (SBCS). A coded character set in which each character is represented by a 1-byte code. A 1-byte code point allows representation of up to 256 characters. See also double-byte character set.

SMF. See System Management Facilities.

SMP/E. See SMP/E for z/OS.

SMP/E for z/OS. An IBM licensed program used to install software and software changes on z/OS systems.

SMS. See storage management subsystem (SMS).

stale file handle. A file handle for a file or prefix that is no longer valid.

stateless. Having no record of previous interactions. A stateless server processes requests based solely on information provided with the request itself, and not based on memory from earlier requests.

storage management subsystem (SMS). Software that automates as much as possible the management of physical storage by centralizing control, automating tasks, and providing interactive controls for system administrators.

superuser. See root user.

System Management Facilities (SMF). A component of z/OS that collects and records a variety of system and job-related information.

System Authorization Facility (SAF). An MVS interface with which programs can communicate with an external security manager, such as RACF.

System Display and Search Facility (SDSF). An IBM-licensed program that provides a menu-driven full-screen interface that is used to obtained detailed information about jobs and resources in a system.

system-managed storage. Storage managed by the storage management subsystem (SMS). System-managed storage attempts to deliver required services for availability, performance, space, and security to applications.

T

TCP/IP. See Transmission Control Protocol/Internet Protocol.

Time Sharing Option (TSO). A base element of the z/OS operating system with which users can interactively work with the system. See also Interactive System Productivity Facility.

Time Sharing Option Extensions (TSO/E). A licensed program that is based on Time Sharing Option (TSO). With TSO/E, MVS users can interactively share computer time and resources.

Transmission Control Protocol (TCP). A communication protocol used in the Internet and in any network that follows the Internet Engineering Task Force (IETF) standards for internetwork protocol. TCP provides a reliable host-to-host protocol in packet-switched communication networks and in interconnected systems of such networks. See also Internet Protocol.

Transmission Control Protocol/Internet Protocol (TCP/IP). An industry-standard, nonproprietary set of communication protocols that provides reliable end-to-end connections between applications over interconnected networks of different types.

TSO. See Time Sharing Option. See also Interactive System Productivity Facility.

TSO/E. See Time Sharing Option Extensions.

U

UDP. See User Datagram Protocol.

UID. See user identification.

Unicode. A character encoding standard that supports the interchange, processing, and display of text that is written in the common languages around the world, plus some classical and historical texts. The Unicode standard has a 16-bit character set defined by ISO 10646.

UNIX. A highly portable operating system that features multiprogramming in a multiuser environment. The UNIX operating system was originally developed for use on minicomputers, but was adapted for mainframes and microcomputers. The AIX operating system is IBM's implementation of the UNIX operating system.

UNIX authentication. The process of identifying a client process, which requires that the client process send credentials to the server.

user catalog. An optional catalog used in the same way as the master catalog and pointed to by the master catalog. Employing a user catalog lessens the contention for the master catalog and facilitates volume portability.

User Datagram Protocol (UDP). An Internet protocol that provides unreliable, connectionless datagram service. It enables an application program on one machine or process to send a datagram to an application program on another machine or process.

user ID. See user identification.

user identification (user ID). The name used to associate the user profile with a user when a user signs on to a system.

V

Virtual Storage Access Method (VSAM). An access method for direct or sequential processing of fixed-length and variable-length records on disk devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry sequence), or by relative-record number.

volume. A discrete unit of storage on disk, tape or other data recording medium that supports some form of identifier and parameter list, such as a volume label or input/output control.

VSAM. See Virtual Storage Access Method.

W

working directory. The active directory. When a file name is specified without a directory, the current directory is searched.

X

XDR. See External Data Representation.

Z

zFS. See z/OS file system.

z/OS. An operating system for the IBM z Series product line that uses 64-bit real storage.

z/OS file system (zFS). A type of file system that resides in a Virtual Storage Access Method (VSAM) linear data set (LDS).

Index

Special characters

- /etc/rpc file, client setup 20
- /etc/services and tcpip.profile files, server setup 20

Numerics

- 0C4 ABEND 295
- 0C4 protection exception 212
- 80A ABEND 140
- 878 ABEND 140

A

- ABEND
  - 80A 140
  - 878 140
  - X'0F3' 314
  - X'806' 314
  - X'A03' 314
  - X'x13' 314
  - X'x22' 314
  - X'x37' 314
  - X'x3E' 314
- ABEND0C4 295
- ABENDxxx keyword 295
- abytes value 457
- access to remote files
  - for BSAM applications 87
  - for QSAM applications 87
  - for VSAM ESDS applications 87
- accesschk client attribute 103, 111
- accessibility 541
- accessing MVS data sets
  - command syntax for AIX 63
  - command syntax for DFSMS 77
- accessing MVS files 73
  - changing attributes 70
  - end-of-line terminators 35
  - getting authorization 71, 92
  - mount command 70, 85
  - mvslogin command 84
  - overriding server default attributes 70
  - showattr command 99
- accessing z/OS UNIX System Services 51
- acdirmax client attribute 103, 111
- acdirmin client attribute 103
- acregmax client attribute 103
- acregmin client attribute 104
- ACS (automatic class selection) routine 120, 276
- active data sets, displaying 228
- activity trace
  - for HFS 527
  - for TCP/IP 527
  - for z/OS UNIX 527
- addds operand 225
- afiles value 457
- AIX
  - client activity trace 528
  - command reference 66
  - command syntax 63
  - downloading client commands 200
  - gcc 4.0.x compiler command 205
  - make command for 202
  - mvslogout command for clients 76
  - NFS client 63
  - showattr command for clients 72
  - umount command for clients 76
  - vi command 70
  - XLC v8 205
- alias names
  - for MVS files 24
- allocated buffers
  - logical I/O processing 139
- allocated storage
  - side file 148
- allocating
  - attributes data set 171
  - exports data set 172
  - lock data sets 182
  - mount handle data sets 182
- altsym keyword 172, 173
- American National Standard Code for Information Interchange (ASCII) 34, 52
- ASCII to EBCDIC conversion 121
- Assembler header macro, reference 515
- assignments, port 20
- assortedparms 269
- async processing attribute 49, 127, 280
- asynchronous block input/output (I/O) daemon (BIOD) 104
- atime 453, 456
- attention messages
  - collecting 233
- attrcaching attribute
  - tuning client with 287
- attrcaching client attribute 104
  - tuning client with 287
- attribute caching
  - specifying 104
- attribute statement syntax 172
- attributes
  - accesschk client attribute 103, 111
  - acdirmax client attribute 103, 111
  - acdirmin client attribute 103
  - acregmax client attribute 103
  - acregmin client attribute 104
  - async processing attribute 49, 127, 280
  - attrcaching client attribute 104
  - attrtimeout 139
  - attrtimeout processing attribute 121
  - binary processing attribute 121, 278
  - biod client attribute 104
  - blankstrip processing attribute 35, 122
  - blks data set creation attribute 116
  - blksize data set creation attribute 116
- attributes *(continued)*
  - bufhigh client attribute 134
  - changing 70
  - cln_ccsid client attribute 104
  - cln_ccsid processing attribute
    - native ASCII environment 130
    - specifying 122
  - convserv client attribute 104, 122
  - cyls data set creation attribute 116
  - data set creation 70
  - datacaching client attribute 105
  - dataclas data set creation attribute 116
  - delaywrite client attribute 105
  - delim client attribute 106
  - dir data set creation attribute 117
  - dsntype data set creation attribute
    - for PDS 26
    - for PDSE 26
    - syntax 117
  - dsorg data set creation attribute 117
    - for direct access (DA) file 25
    - for partitioned data set (PDS) 276
    - for physical sequential (PS) file 24
    - partitioned data set extended (PDSE) 276
  - dynamicsizeadj client attribute 106
  - executebitoff processing attribute 123
  - executebiton processing attribute 123
  - extlink processing attribute 49, 123
  - fastfilesize processing attribute 123
  - fileextmap processing attribute 40, 123
  - for z/OS UNIX System Services 115
  - hard client attribute 109
  - hfs site attribute 49
  - keys data set creation attribute 117
  - llock client attribute 107
  - logout site attribute 69, 139
  - lrecl data set creation attribute 118
  - mapleaddot processing attribute 124
  - maplower processing attribute 31, 34, 124
  - mapped processing attribute 124
  - maxtimeout 139
  - mgmtclas data set creation attribute 118
  - mintimeout 139, 140
  - MVSMNT processing attribute 124
  - nfstasks site attribute 142, 280, 284
  - nfsv4domain client attribute 107
  - NFSV4DOMAIN site attribute 143
  - noattrtimeout processing attribute 121
  - noblankstrip processing attribute 122
  - nofastfilesize processing attribute 123
  - nofileextmap processing attribute 40, 123
  - nomapleaddot processing attribute 124

© Copyright IBM Corp. 1991, 2010

- attributes *(continued)*
  - nomaplower processing attribute 31, 34, 124
  - nomaxtimeout 139
  - nordrcache processing attribute 125
  - nordrverf processing attribute 125
  - noreadtimeout 139
  - noreadtimeout processing attribute 125
  - noretrieve process attribute 280
  - noretrieve processing attribute 38, 126, 280
  - norlse data set creation attribute 119
  - notag processing attribute 127
  - nowritetimeout processing attribute 128, 139
  - processing 70, 121
  - proto client attribute 107
  - public client attribute 107
  - rdrverf processing attribute 125
  - readahead client attribute 108
  - readtimeout 140
  - readtimeout processing attribute 125
  - recfm data set creation attribute 118
  - recordsize data set creation attribute 118
  - recs data set creation attribute 116
  - restimeout 71
  - retrans client attribute 108
  - retrieve process attribute 280
  - retrieve processing attribute 38, 126, 280
  - retrieve(wait) 280
  - retry client attribute 108
  - rlse data set creation attribute 119
  - rpcbind client attribute 108
  - rsize client attribute 108
  - saf 68
  - safexp 68
  - secure client attribute 109
  - security 68, 241
  - security site attribute 13
  - security(exports) 159, 163
  - security(none) 163
  - security(saf) 160, 503
  - security(safexp) 160, 161, 503
  - setownernobody processing attribute 126
  - setownerroot processing attribute 126
  - shareoptions data set creation attribute 119
  - side file processing attribute 126
  - sidefile processing attribute 40
  - site 134
  - soft client attribute 109
  - space data set creation attribute 119
  - spanned data set creation attribute 119
  - srv_ccsid processing attribute
    - native ASCII environment 130
    - specifying 127
  - storclas data set creation attribute 119
  - sync processing attribute 49, 127, 280
  - tag processing attribute 127
  - text processing attribute 121, 278
- attributes *(continued)*
  - trks data set creation attribute 116
  - trusted 153
  - unit data set creation attribute 120
  - vol data set creation attribute 120
  - writetimeout 140
  - writetimeout processing attribute 128
  - xlat 183, 184
  - xlat processing attribute 128
  - z/OS UNIX 49, 280
  - znfsclient processing attribute 128
- attributes data set
  - allocating 171
  - changing 171
  - GFSAPATT sample member 171
  - modifying 171
- attributes, data set 13
- attrtimeout processing attribute 121, 128, 279
- attrtimeout site attribute 139, 140
- authentication error 85
  - restart processing 69
- authentication, protocols 163
- authority, trusted 153
- authorization checking for z/OS UNIX files 50
- Authorized Program Analysis Report (APAR) 293
- authorizing file operations 163
- authsys (system authentication) security 158
- automatic class selection (ACS) routine 120, 276
- automatic logout, recovering 71, 92
- automount facility 85, 168

B

- badcall statistic 264
- badxid statistic 264
- binary files 35
- binary processing attribute 121
- binary processing mode 34
- biod (block input/output (I/O) daemon) 288
- biod client attribute 104
- blanks
  - handling of 35
  - trailing 122
- blankstrip processing attribute 35, 122
- blks data set creation attribute 116
- BLKSIZE (block size) 275
- blksize data set creation attribute 116
- block input/output (I/O) daemon (BIOD) 104, 134, 268, 282, 284, 288
  - tuning client with 287
- block size (BLKSIZE) 275
- BPXVCLNY load module 212
- BSAM (basic sequential access method)
  - access to remote files 4, 87
  - line delimiter for record access 106
- buffer client block writes
  - out of order 134
- buffer size
  - write buffer 110
- buffers
  - caching 281
- buffers *(continued)*
  - number of physical block buffers 139
  - reclaimed
    - percentage of 144
  - usage of 281
- bufhigh client attribute
  - specifying 134
  - tuning client with 287, 288
- bufhigh operand 225, 226
- bufhigh site attribute
  - buffer reclamation 282
  - displaying value of 100
  - logicalcache value 139
  - percentsteal value 144
  - storage considerations 284
- byte-range locking 61

C

- C header macro, reference 507
- cached buffers, disk 128
- cached data
  - reading 144
  - writing 33
- cached directory attributes 103
- cachewindow site attribute
  - displaying value of 100
  - for write requests 282, 284
  - modified only at server startup 282, 284
  - specifying 134
- cancel command 218
- cancel mvsnfsc command 212
- carriage-return newline (CRNL) 82
- case_insensitive value 457, 458
- case_preserving value 457, 458
- cat command 26, 33, 35
- cataloged data sets 6, 41
- cbsniff operand 226
- changing
  - attributes 70, 85
  - attributes data set 171
  - mount handle data sets 232
  - your MVS password 83
  - your z/OS password 67
- Character Data Representation Architecture (CDRA)
  - for data conversion 86, 164
- checking UNIX permission bits 147
- checklist data set 181
- checklist site attribute 135, 161
- chmod command 32
- chown_restricted value 457, 458
- client
  - inactive time limit 139
  - storage limit 134
- client activity trace
  - for AIX 528
  - for SUN 529
- client attributes
  - accesschk 103, 111
  - acdirmax 103, 111
  - acdirmin 103
  - acregmax 103
  - acregmin 104
  - attrcaching 104
  - biod 104

client attributes *(continued)*
  bufhigh 134
  cln_ccsid 104
  convserv 104, 122
  datacaching 105
  delaywrite 105
  delim 106
  disablella 106
  dynamicsizeadj 106
  hard 109
  llock 107
  nfsv4domain 107
  proto 107
  public 107
  readahead 108
  retrans 108
  retry 108
  rpcbind 108
  rsize 108
  secure 109
  soft 109
  srv_ccsid 109
  stringprep 109
  syntax 103, 110
  timeo 110
  vers 110
  wsize 110
  xlat 110
client commands
  installing 198
client hanging 145
client id
  specifying in exports data set 179
client log data set
  setting up 523
client nfs 263
client program requirements 454
client rpc 263
client-server relationship
  overview of 3
client, z/OS NFS
  stopping 212
client/server, user specified port range support 20
cln_ccsid client attribute 104
cln_ccsid processing attribute
  native ASCII environment 130
  specifying 122
closing a data set 229, 230
coded character set identifier (CCSID)
  for data conversion 86, 164
  for native ASCII environment support 130
  for native ASCII support 21
  in mount tag option 113
  specifying for local mounted file system 104, 127
  specifying for remote mounted file system 109
collecting
  diagnostic messages 233
  usage statistics 165
command
  cancel 218
  cancel mvsnfsc 212
  cat 26, 33, 35
  chmod 32
command *(continued)*
  copy 27, 271
  cp 23, 271
  crnl2nl 82
  date 271
  f omvs,stoppfs=NFS 212
  for mount operation 42
  force 212
  ln 53
  ls 26
  ls (UNIX) 123, 450
  make 202
  make example 502
  mkdir 23, 25, 26, 43, 276
  modify 70, 71, 172, 222
  more 53
  mount 5, 23, 38, 41, 64, 69, 70, 74, 77, 85, 162, 263, 275, 277, 280
    example 70
  mvslogin 63, 67, 68, 78, 84, 243
  mvslogin (client) 198
  mvslogout 66, 67, 76, 83, 101, 243
  mvslogout (client) 198
  net use 23
  netstat -s 264, 266
  nfsstat 79, 94, 263, 268
  nl2crnl 82
  rm 26, 54
  rm (UNIX) 123
  rmdir 26
  showattr 24, 25, 50, 65, 68, 72, 73, 79, 84, 99
  showattr (client) 198
  showmount 81, 98
  start 173, 217
  stop 217
  time 271
  timex 271
  touch 202
  TSO ALLOCATE 87
  TSO HELP MOUNT 77
  TSO HELP UNMOUNT 77
  TSO MOUNT 87, 164
  umount 65, 75
  unmount 77, 92
command syntax
  AIX user 63
  for AIX client 63
  for DFSMS 77
  for NFS clients 77
  for UNIX client 63
comment symbol 216
comments from the requestor 505
commit procedure
  cached data writing 33
compatibility
  POSIX 48
component identification keyword 294
component trace
  command for starting on z/OS NFS client 212
  command for starting on z/OS NFS server 218
concurrent write, PDSE 27
configuring
  attributes data set 171
  NFS 153
configuring the server
  attributes data set 172
  exports data set 172
  mount handle data sets 172
contiguous port range 20
control files 13
control timeout, HFS vnode token 137
controlling access to data sets 172
convserv client attribute 104, 122
convserv processing attribute
  values 165
    customized conversion 165
    enforced subset conversion 165
    Language Environment-Behavior conversion 165
    Modified Language Environment-Behavior conversion 165
    roundtrip conversion 165
CONVXLAT utility 184
cookie verifier checking 125
copy command
  creating VSAM files with 27
cp command 23
creating external link 53
creating VSAM files 27
creating z/OS conventional MVS data sets 6
credentials, UNIX-style 164
crnl2nl command
  syntax for DFSMS 82
ctime 453
CTINFC00 member of SYS1.PARMLIB 302
CTINFS00 member of SYS1.PARMLIB 299
ctrace
  operand of the START mvsnfs command 216
customized conversion 165
customizing
  exit routines 239
  NFS 153
  translation table 183
cyls data set creation attribute 116

D

DASD volume
  for data set creation 120
data access/creation commands 33
data class
  specifying 116
data conversion
  parameters used 164
  Unicode Standard 86
  xlat client attribute 110
data labeling (RACF option)
  z/OS NFS server support for 194
data set
  attributes 6, 13
  binary format 121
  cataloged 6
  cataloged, organizations supported 41
  checklist 181
  creation attributes 116

data set *(continued)*
definition 5, 40
DSNTYPE 276
DSORG 276
exports data set 13
MVS 252
organization 117
organization (DSORG) 276
readdirtimeout site attribute 145
record format and characteristics 118
release unused space 119
released after a read 125
released after a write 128
rules for file extension mapping 126
serialization 44
share options (VSAM) 119
side file 40
spanned data set creation attribute (VSAM) 119
structure 116
text format 121
timeout specification 279
type (DSNTYPE) 276
unit to create on 120
volume for 120
data set creation attributes
blks 116
blksize 116
cyls 116
dataclas 116
dir 117
dsntype 117
dsorg 117
keys 117
lrecl 118
mgmtclas 118
norlse 119
recfm 118
recordsize 118
recs 116
rlse 119
shareoptions 119
space 119
spanned 119
storclas 119
trks 116
unit 120
vol 120
data transmitted, write request 127
data types, mixed set 124
databufferpoolsize 269
datacaching attribute 110
datacaching client attribute 105
tuning client with 287
DATACLAS 276
dataclas data set creation attribute 116
DCB (data control block) parameters 116
deallocating a data set 230
debug operand 233
debugging
collecting messages for 233
default attributes
displaying 73
default attributes, overriding 70, 85
defaults
displaying 72
defaults, displaying 100
delay detection timeout 135
delay write
maximum number of disk blocks for 105
delaywrite client attribute 105
tuning client with 287
delaywrite mount parameter
tuning client with 288
delegation site attribute 135
delegation/nodelegation 45
deleting
entries from mount handle data sets 232
migrated files 126
delim client attribute 106
denyrw site attribute 135
development toolkits 203, 205
DFSMS
command syntax for 77
DFSMShsm
recall or delete migrated files 126
dhcp site attribute 135
diagnosis
diagnostic aids 313
modify command for 233
of problems 293
reporting problems 293
using keywords 293
using RETAIN 313
diagnostic errors
messages for 314
dir data set creation attribute 117
direct (DA) data sets
file size value for 450
direct access (DA) data set
attribute for creating 117
direct access (DA) data sets
supported by z/OS NFS server 23
time stamps for 453
using fastfilesize for 451
direct access (DA) files
creating 25
directory
client commands 117
directory statement
in the exports data set 173
disability 541
disablella client attribute 106
disconnecting a mount point 75, 92
disk blocks
maximum number for delay write 105
read ahead 108
dispatching priority 269
displaying
client statistical information 94
default and mount point attributes 73, 99
default attributes 73, 100
mount point attributes 72
mount points and active data sets 228
remote server mount information 98
site and mount point attributes 24
status of active subtasks 231
displaying attributes 73
DlyDtimeout site attribute 135
DOC SCnnnnnnnn keyword 296
Domain Name Server 215
double byte character set (DBCS) 183
for data conversion 86
downloading
commands to the client 198
NFSTARB files 198
source code to the client 501
DSNTYPE (data set type) 276
dsntype data set creation attribute
for PDS 26
for PDSE 26
syntax 117
DSORG (data set organization) 276
dsorg data set creation attribute 117
dsorg(ps) attribute
for physical sequential (PS) file 24
dsps
operand of the START mvsnfs command 217
dtpref value 457, 458
Dump Analysis and Elimination (DAE) 314
dump data set, abnormal end 311
dynamic IP addresses for clients 135
dynamic IP addressing 20
server configuration 191
dynamicsizeadj client attribute 106
tuning client with 287
dynamicsizeadj mount parameter 287

E

EBCDIC code page 0037 183
EBCDIC to ASCII conversion 121
Electronic Technical Response (ETR) 312
encoded password 505
encoded user name 505
end-of-file mark (EOF) 82
end-of-line specifiers 35, 122
end-of-line terminators 35
ending your z/OS session 76, 101
enforced subset conversion 165
Enterprise Storage Server (ESS) 276
entry-sequenced data set (ESDS) 87
creating VSAM files 27
supported by z/OS NFS server 23
error messages
collecting 233
ESDS (entry-sequenced data set) 87
ESS 2105 DASD 276
executebitoff processing attribute 123
executebiton processing attribute 123
exit routines
customizing 239
export entry, return first character 138
export spanning pathnames 18
exportfs operand 70, 227
exporting a file system 162
exports data set 13
exports list and SAF checking 159, 160
extended address volumes
data sets on 29
extended binary-coded decimal interchange code (EBCDIC) 34, 52
extended format data sets, exploiting 29
External Data Representation (XDR) 4

external link 53
externalized return codes
NFS version 2 protocol 417, 420
NFS version 3 protocol 417
NFS version 4 protocol 418
extlink processing attribute 49, 123

F

fastfilesize 269, 278
fastfilesize processing attribute 123, 450
FAT file system 203
fbytes value 457
ffiles value 457
file attributes, NFS version 4 protocol 457
file creation
attributes 116
data class 116
management class 118
storage class 119
file extension mapping 40
side file 40
file name
mapping between lower and upper case 124
file naming conventions
for MVS files 24
file processing
overriding default translation table 128
file security exit
GFSAUSEC 250
parameter list 250
purpose 247
request code 252
return codes 252
file size
determination of 32, 449
number of physical block buffers 139
file system 5, 40
file system attributes for MVS 44
file system size 55
file tag
for new files 127
File Transfer Protocol (FTP) 198
fileextmap processing attribute 40, 123
fileidsize site attribute 135
files
file tag for 127
locking 59
saving fixed-length MVS file 35
source for client enabling commands 203
filesystype parmlib statement 167
filtering NFS ctrace records in IPCS 307
fixed-blocked file format 116
flushlog operand 234
FMIDs for z/OS NFS 294
fn_delimiter site attribute 136
force command 212
freeds operand 227
freeze operand 228
freeze=off operand 71
freeze=offhfs operand 71
freeze=on operand 71
freeze=onhfs operand 71
FSF_CANSETTIME, properties 457
FSF_HOMOGENEOUS, properties 457
FSF_LINK, properties 457
FSF_SYMLINK, properties 457
fsinfo, static file system 457, 458
fsstat, dynamic file system 457

G

Get Attribute
data set timeout specification 279
file size determination 278
getattr call 264
getattr operation
data set timeout specification 121
GFSAPATT sample member 465
GFSAPROC
in NFSSAMP library 495
GFSASSMF macro 507
GFSAUDSA user storage block 239
GFSAULOG
for login exit parameter list 243
GFSAUSEC 250
GFSAUSMF macro 515
GFSCPROC
in NFSSAMP library 499
GID
mapping to group name 504
global exit block (GXB) 247
Global Exit Block (GXB) 240, 244
grace period for reclaiming locks 60, 61
granting access to data sets 172
group name
mapping to GID 504
group number (GID) 32, 50

H

hard client attribute 109
HFS
activity trace 527
hfs site attribute 49, 50
hfsfbtimeout site attribute 137
hfsprefix site attribute 136
HFSPREFIX site attribute 9
hfssec site attribute 137
hierarchical file system (HFS) 4, 47, 53, 54, 130, 277, 285
high-level qualifier (HLQ)
for MVS data sets 5
mapping to workstation file system 40
hostname
mvslogin command 84

I

IBMLink/Service
diagnosis using 293
implicit prefix support
restrictions 11
impprefix site attribute 138
IMPPREFIX site attribute 9
inactive time limit 139
incorrout keyword 296
increase network bandwidth 266
informational messages
collecting 233
input errors 293
installation default settings, overriding 70
installation exits 163
installation parameters 111
installing
mvslogin (client) command 198
mvslogin command 501
mvslogout (client) command 198
mvslogout command 501
showattr (client) command 198
showattr command 501
Internet Protocol (IP) address 13
dynamic 20, 135, 191
internet protocol version 6
SMF records for 166
invarsec value 457
IPCS
using to view NFS ctrace records 306
iptrace utility 317
ISHELL utility 48
ISO 8859-(ASCII) 183

J

job control language (JCL) 116

K

Kerberos
acquiring Kerberos tickets 155
authentication
in the exports data set 176
NFS V4 protocol 154
on hfssec site attribute 137
Kerberos V5 based integrity 176
key distribution center 537
NFS client
RACF requirements 155
support for Linux 535
windows recommendations 536
key-sequenced data set (KSDS)
creating VSAM files 27
supported by z/OS NFS server 23
keyboard 541
keys
for VSAM KSDS data set
length and offset 117
keys data set creation attribute 117
keyword
altsym 173
for diagnosis 293
KSDS (key-sequenced data set) 87

L

Language Environment-Behavior conversion 165
large format data sets
accessing 29
leadswitch site attribute 138
leasetime for locks 60, 61
leasetime site attribute 139
lf end-of-line specifier 35

limitations, PDS 27
limiting access to data sets 172
line delimiter for record access 106
link request 44
linkmax value 457, 458
Linux
  CITI web site for 536
  client/server definitions with Kerberos support 535
  command for mount operation 42
  des-cbc-md5 encryption type 537
  downloading client commands 200
  Fedora 7 535
  gcc 4.0.x compiler command 205
  make command for 202
  mount command example 42
  mvslogout command for clients 76
  NFS client
    RACF requirements 155
  Redhat Enterprise 535
  showattr command for clients 72
  umount command for clients 76
list operand 228
listing lock holders for a file 60, 62, 234
listlock operand 60, 62, 234
llock client attribute 107
ln command 53
local host name 505
local mounted file system
  specifying CCSID for 104, 127
localpath 64
lock data sets 182
lock holders
  listing 60, 62, 234
lock time limit 60, 61
lock time, control 139
lockd protocol 463
locking and access control 59
log data set
  how used 14
  NFS client
    setting up 524
  NFS server
    setting up 523
  NFSLOG1 222, 233
  NFSLOG2 222, 233
  setting up 523
log operand 235
logical file system (LFS) 167
logical I/O processing
  buffer client block writes 134
  cache windows 139
logicalcache site attribute 139
  displaying value of 100
  for write requests 282, 284
  modified only at server startup 282, 284
  storage considerations 284
login exit
  GFSAULOG 243
  parameter list 243
login exit routine 241
logout
  forced logout 241
logout attribute, maxtimeout site attribute 139
logout site attribute
  displaying value of 100
  purpose of 139
  when exceeded 69
logout, recovering from automatic 71, 92
LookAt message retrieval tool xv
lookup 264
lookup lookaside (LLA) caching
  enabling or disabling 106
lookup operation
  data set timeout specification 121
Lookup operation
  data set timeout specification 279
lookup request 17, 144
loop keyword 296
low system usage 266
lower case file name
  mapping to upper case 124
lrecl data set creation attribute 118
ls (UNIX) command 123, 450
ls command 26

M

mainframe
  education xix
make command
  for AIX 202
  for Linux 202
  for Sun 202
  for zLinux 202
make command example 502
management class
  file creation 118
mapfile operand 229
mapleaddot processing attribute 124
maplower processing attribute 31, 34, 124
mapped keyword processing attribute 129
mapped processing attribute 124
mapping
  between lower and upper case file names 124
  file extensions 126
  GID to group name 504
  group name to GID 504
  return codes 420
  UID to user name, 504
maxfilesize value 457
maximum size
  allocated buffers 139
maximum transmission unit (MTU) 268
maxrdforszleft site attribute 139, 282
  displaying value of 100
maxtimeout site attribute 139, 140, 279
  displaying value of 100
message log data set
  setting up 523
message retrieval tool, LookAt xv
messages
  for diagnostic errors 314
messages, display in mixed/upper case 140
mgmtclas data set creation attribute 118
migrated files
  recall or delete 126
mintasks site attribute 140
mintimeout site attribute 139, 140, 279
  displaying value of 100
mixcase site attribute 140
mixed case, message display 140
mixed set of data types 124
mkdir command 23, 26, 43, 276
  for UNIX 25
mknod request 44
MLNAMES (RACF option)
  z/OS NFS server support for 194
Modified Language Environment-Behavior conversion 165
modify command 70, 71, 166, 172, 222
modify command, operands of 217
modifying
  attributes data set 171
  mount handle data sets 232
modifying file attributes 70, 85
monitored locks 59
more command 53
mount command 5, 23, 38, 41, 70, 77, 85, 162, 263, 275, 277, 280
  changing site and mount point attributes 70
  creating a physical sequential (PS) file with 24
  overriding default attributes 70
  syntax on AIX 64
  z/OS examples 89
mount emulation if NFS V4 124
mount handle data set 14, 182
mount operation
  number of times to retry 108
mount point 116, 135
  changing attributes 70
  command 43
  creating a mount point for a PDS 26
  definition 41
  disconnecting 75, 92
  displaying attributes of 72
  multiple 43
  retention period for 146
  showattr command 72
mount point, access 26
mount point, definition 5
mount points, saving 71, 92
mount processing parameters 111
mount requests 136, 138, 140
mount tag option 112
  CCSID in 113
MSGGFShnnnt keyword 295
mtime 453, 456
MTU (Packet Size) 316
multiple byte character set (MBCS) 183
multiple data set creation attributes, specifying 116
multiple NFS servers 217
MVS
  password and user ID 83
MVS files
  attributes 103, 115
  changing attributes 70
  creating 23
    direct access files 25
    PDSs and PDSEs 25
  end-of-line terminators 35

MVS files *(continued)*
mount command 70, 85
mvslogin command 84
mvslogout command 76, 101
naming 24
overriding creation attributes 23
physical sequential 24
restrictions on using alias names 24
saving 35
selecting format 31
showattr command 72, 99
umount command 75
unmount command 92
MVS programs 43
mvslogin command 139
authentication errors 84
examples of 68
exports list checking 159
installing 198
restricting
based on SERVAUTH 192
based on terminal ID 191
security attribute 78
SERVAUTH based restrictions 192
syntax for clients 84
syntax for DFSMS 78
syntax on AIX 63
terminal ID based restrictions 191
to access MVS data sets 67
to access remote z/OS files 83
to access z/OS UNIX files 67
mvslogout command 67, 76
installing 198
syntax for DFSMS 83
syntax on AIX 66
to end access to remote z/OS files 83
MVSMNT processing attribute 124
mvsprefix site attribute 140
MVSPREFIX site attribute 9
mvssec site attribute 141

N

name_max value 457, 458
name-hiding (RACF option)
z/OS NFS server support for 194
namesrv 215
naming conventions
for MVS files 24
native ASCII environment support 130
native ASCII NFS client support 112
native ASCII support 21
native path 17
netgroups
specifying in the exports data set 180
netstat -s command
for tuning NFS 264
monitoring network activity with 266
Network File System (NFS)
overview of 3
Network Lock Manager (NLM) 59
Network Status Monitor (NSM) 60
newline (NL) 82
NFS
configuring 153
customizing 153
NFS client
acquiring Kerberos tickets 155
commands for 77
MD5 Checksum with DES encryption 155
native ASCII environment support 112
non-z/OS based 112
operating 211
starting 211
stopping 212
storage limit 134
tested 14
translation support 112
nfs link command 23
NFS protocol attributes
z/OS NFS server support for 459
NFS server
non-z/OS based NFS clients 131
NFS server attributes 457
NFS version 2 protocol
externalized return codes 417
mapping to externalized return codes 420
NFS version 3 protocol
externalized return codes 417
return codes 422
NFS version 4 file attributes 457
NFS version 4 protocol
attributes for NFS server 15
externalized return codes 418
return codes 424
tested clients for z/OS NFS server 14
NFSC_001 93
NFSCTRDS data space 304, 305
NFSLOG1 log data set 222, 233
NFSLOG2 log data set 222, 233
NFSSAMP library
GFSAPROC in 495
GFSCPROC in 499
sample startup procedures 495, 499
nfsstat command 94
nfsstat -c 263, 268
nfsstat -in 268
nfsstat -s 268
nfsstat -z 263
syntax for DFSMS 79
nfsstat mount parameter 289
NFSTARB files
downloading 198
nfstasks site attribute 142, 280, 284
displaying value of 100
nfsv4domain client attribute 107
NFSV4DOMAIN site attribute 143
nl2crnl command
syntax for DFSMS 82
nlm site attribute 143
no_trunc value 457, 458
noattrtimeout processing attribute 121
noattrtimeout site attribute 139
noblankstrip processing attribute 122
nochecklist site attribute 135
nodelegation site attribute 135
nodenyrw site attribute 135
nodhcp site attribute 135
nofastfilesize 269, 278
nofastfilesize processing attribute 123, 279, 451
nofileextmap processing attribute 40, 123
noleadswitch site attribute 138
nomapleaddot processing attribute 124
nomaplower processing attribute 31, 34, 124
nomaxtimeout site attribute 139
non-monitored lock 59
non-monitored locks 59
nonlm site attribute 143
nonspanned data set creation attribute 119
nopcnfsd site attribute 143
nopublic site attribute 144
nordrcache processing attribute 125
nordrverf processing attribute 125
noreadtimeout processing attribute 125
noreadtimeout site attribute 139
norec878 site attribute 146
noremount site attribute 146
noretrieve process attribute 280
noretrieve processing attribute 38, 126, 280
norlse data set creation attribute 119
normalization, stringprep
enabling 149
nostringprep site attribute 149
notag processing attribute 127
noudpqueuelimit 269
nowritetimeout processing attribute 128, 139
nowritetimeout site attribute 139
number of physical block buffers 139
number representation 36

O

obtaining an MVS password 67, 83
OEDIT editor, UNIX 27
OEMVS311 translation table 52, 184
omvs stoppfs command 212
operands of the MODIFY command 217
operating the client
starting 211
stopping 212
operating the NFS NSM, NLM 218
operating the server
starting 215
stopping 217
ordering out-of-sequence data 282
out of order
buffer client block writes 134
overflow site attribute 143
overriding default attributes 70, 85
overriding default translation table 128
overview
of Network File System (NFS) 3

P

packet size adjustment
for remote procedure call (RPC) 106
partial record identification 36

partitioned data set (PDS)
  attribute for creating 117
  CONXLAT utility 184
  creating 25, 26
  directory records 117
  dsorg 276
  extending 27
  file extension mapping 40
  file size value for 450
  limitations of 27
  members 26
  removing 26
  serializing and sharing data sets 6
  supported by z/OS NFS server 23
  time stamps for 454, 456
  timing out while writing 27
  updating 27
  using as a directory 43
  using fastfilesize for 451
  wildcard copy restriction 27
partitioned data set extended (PDSE)
  attribute for creating 117
  concurrent write 27
  CONXLAT utility 184
  creating 25, 26
  dsorg 276
  extending 27
  file extension mapping 40
  members 26
  removing 26
  serializing and sharing data sets 6
  time stamps for 454, 456
  timing out while writing 27
  updating 27
  using as a directory 43
  using fastfilesize for 451
  wildcard copy restriction 27
password
  for MVS 83
password (MVS) 160
password (z/OS) 67
pathconf, retrieve POSIX information 457, 458
PCNFSD
  accessing data with 503
  authentication 153, 503
  authentication request 69
  version 1 protocol 504
  version 2 protocol 504
pcnfsd site attribute 143
percentsteal site attribute 144
  buffer reclamation 282
  displaying value of 100
perfm keyword 297
perform mapping 504
permission bits, checking UNIX 147
permission denied message 69, 84
permissions
  security 247
physical block
  maximum length 116
physical block buffers
  number of 139
physical sequential (PS) data sets
  time stamps for 453
physical sequential (PS) files
  supported by z/OS NFS server 23
physical sequential data sets
  serializing and sharing data sets 6
physical sequential files
  creating 24
ping utility 317
planning for installation
  security measures 153
port assignments 20
port range, user specified 186
portable operating system interface (POSIX) 3, 48
porting commands for clients 203
portmap 215
  required for starting NFS client 211
portmapper 18, 316
Portmapper
  configured as generic server 195
  configured with transport affinity 195
  rcpinfo utility for 317
  use only for IPv4 216, 316
procedure 13, user authentication 505
process attribute
  noretrieve 280
  retrieve 280
processing attribute
  async 127
  attrtimeout 121
  binary 121
  blankstrip 122
  cln_ccsid 122
    native ASCII environment 130
  executebitoff 123
  executebiton 123
  extlink 123
  fastfilesize 123
  fileextmap 40, 123
  mapleaddot 124
  maplower 31, 34, 124
  mapped 124
  MVSMNT 124
  noattrtimeout 121
  noblankstrip 122
  nofastfilesize 123
  nofileextmap 40, 123
  nomapleaddot 124
  nomaplower 31, 34, 124
  nordrcache 125
  nordrverf 125
  noreadtimeout 125
  noretrieve 38, 126, 280
  notag 127
  nowritetimeout 128, 139
  rdrverf 125
  readtimeout 125
  retrieve 38, 126, 280
  setownernobody 126
  setownerroot 126
  side file 126
  sidefile 40
  srv_ccsid
    native ASCII environment 130
    specifying 127
  sync 127
  tag 127
  text 121
  writetimeout 128
processing attribute *(continued)*
  xlat 128
  znfsclient 128
processing attributes 121, 129
PROCLIB updates 168
protecting the file system
  with NFS V4 security 154
  with the security site attribute 158
protecting your programs and files 153
protecting your z/OS UNIX System Services files 50
proto attribute 18
proto client attribute 107
protocol attributes
  z/OS NFS server support for 459
PTFs, APARs, search 312
public client attribute 107
public file handle 144, 147
  forcing use of 107
public keyword 17
public site attribute 144
pubsec site attribute 145

Q

QSAM (queued sequential access method)
  access to remote files 4, 87
  line delimiter for record access 106
qualifiers
  high-level 5, 40

R

RACF (Resource Access Control Facility)
  authorizing the z/OS NFS client 154
  data labeling option 194
  MLNAMES option 194
  name-hiding option 194
  ownership and permissions 32
  protecting server control files 153
  protecting z/OS UNIX files 50
  SAF checking 160
racf_cache_reset operand 229
RACROUTE
  authorizing remote client 154
RAMAC3 276
random access to files 33
rcpinfo utility 317
rdrverf processing attribute 125
read ahead
  maximum disk blocks 108
read buffer size
  specifying 108
read statistic 264
readahead buffers 144
readahead client attribute 108
  tuning client with 287
readahead mount parameter 288
readaheadmax site attribute 145, 282
  displaying value of 100
readdirplus request 44, 95, 97
readdirtimeout site attribute 145
reading out of cached data 144
readlink request 44

readtimeout processing attribute 125, 128, 279
readtimeout site attribute 140
reason codes 427
rec878 site attribute 146
recalling migrated files 126
recfm data set creation attribute 118
record access
  line delimiter for 106
record boundaries 35
record format (RECFM) 276
record length (LRECL) 275
record type-42 (subtype 7, subtype 8) 166
records
  locking 59
recordsize data set creation attribute 118
recs data set creation attribute 116
reduce data transfer 266
reference
  Assembler header macro 515
  C header macro 507
relative record 27
relative record data set (RRDS)
  creating VSAM files 27
  supported by z/OS NFS server 23
release level keyword 295
release operand 230
releases and FMIDs 294
releasing locks 60, 62, 230
remote file system access control lists 55
remote mount function 70
remote mounted file system
  specifying CCSID for 109
remote procedure call (RPC) 4, 95, 212
  maximum number of retries 109
  number of times to retransmit 108
  packet size adjustment for 106
  timeout for 110
remount site attribute 146
remove request 54
removing entries from mount handle data sets 232
request code
  file security exit 252
  login exit 252
Resource Access Control Facility (RACF)
  authorizing the z/OS NFS client 154
  data labeling option 194
  MLNAMES option 194
  name-hiding option 194
  ownership and permissions 32
  protecting server control files 153
  protecting z/OS UNIX files 50
  SAF checking 160
Resource Measurement Facility (RMF) 272
restart processing 85
restarting the z/OS NFS client 211
restimeout attribute 71
restimeout site attribute 14, 146
restricting access to data sets 172
restrictions 15, 17
resuming mount processing 228
RETAIN
  search symptom strings 313
retrans client attribute 108
retrieve attributes 129
retrieve process attribute 280
retrieve processing attribute 38, 126, 280
retrieve(nowait) 280
retrieve(wait) 280
retrieve(wait) attribute 280
retrieving client commands 200
retry client attribute 108
return codes
  externalized
    NFS version 2 protocol 417
    NFS version 3 protocol 417
    NFS version 4 protocol 418
  file security exit 252
  login exit 252
  mapping 420
  NFS version 3 protocol 422
  NFS version 4 protocol 424
reuse option
  creating VSAM files 27
RFC 2055 17
rlse data set creation attribute 119
rm (UNIX) command 123
rm command 26, 54
rmdir command 26
root directory 4, 47
roundtrip conversion 165
RPCBIND
  configured as generic server 195
  configured with transport affinity 195
  rcpinfo utility for 317
  required for IPv6 216, 316
rpcbind client attribute 108
RRDS (relative record data set) 87
rsize 275
rsize client attribute 108
rsize mount parameter 289
rtmax value 457, 458
rtmult value 457, 458
rtpref value 457, 458

S

saf attribute 68
saf checking, bypass 135
safexp attribute 68
SAM striped files 28
samples
  exports data set 487
  startup procedures for z/OS NFS client 499
  startup procedures z/OS NFS server 495
saving MVS files 35
saving of mount points 71, 92
sdump service 314
secure client attribute 109
security 159
security and z/OS UNIX System Services 50
security attribute 68, 241
  affect on mvslogin command 78
Security Authorization Facility (SAF) 241, 285
security exit 250
security measures
  planning for 153
security options, none, saf, safexp, exports 147
security site attribute 147
security, user specified port range support 20
security(exports) attribute 163
security(none) attribute 163
security(saf) attribute 161, 503
security(safexp) attribute 503
security(safexp) attributes 161
separation character in exports data set 173
sequential access method striped data sets 269
sequential byte stream 35
SERVAUTH
  restricting mvslogin by 192
server
  z/OS NFS
    control files 13
    creating z/OS conventional MVS data sets 6
server attributes 115
  async processing attribute 49, 127
  attrtimeout processing attribute 121
  binary processing attribute 121
  blankstrip processing attribute 122
  blks data set creation attribute 116
  blksize data set creation attribute 116
  cln_ccsid processing attribute
    native ASCII environment 130
    specifying 122
  cyls data set creation attribute 116
  dataclas data set creation attribute 116
  dir data set creation attribute 117
  dsntype data set creation attribute
    for PDS 26
    for PDSE 26
    syntax 117
  dsorg data set creation attribute 117
  executebitoff processing attribute 123
  executebiton processing attribute 123
  extlink processing attribute 49, 123
  fastfilesize processing attribute 123
  fileextmap processing attribute 40, 123
  hfs site attribute 49
  keys data set creation attribute 117
  lrecl data set creation attribute 118
  mapleaddot processing attribute 124
  maplower processing attribute 31, 34, 124
  mapped processing attribute 124
  mgmtclas data set creation attribute 118
  MVSMNT processing attribute 124
  noattrtimeout processing attribute 121
  noblankstrip processing attribute 122
  nofastfilesize processing attribute 123
  nofileextmap processing attribute 40, 123
  nomapleaddot processing attribute 124

server attributes *(continued)*
- nomaplower processing attribute 31, 34, 124
- nordrcache processing attribute 125
- nordrverf processing attribute 125
- noreadtimeout processing attribute 125
- noretrieve processing attribute 38, 126, 280
- norlse data set creation attribute 119
- notag processing attribute 127
- nowritetimeout processing attribute 128, 139
- rdrverf processing attribute 125
- readtimeout processing attribute 125
- recfm data set creation attribute 118
- recordsize data set creation attribute 118
- recs data set creation attribute 116
- retrieve processing attribute 38, 126, 280
- rlse data set creation attribute 119
- setownernobody processing attribute 126
- setownerroot processing attribute 126
- shareoptions data set creation attribute 119
- side file processing attribute 126
- sidefile processing attribute 40
- space data set creation attribute 119
- spanned data set creation attribute 119
- srv_ccsid processing attribute
  - native ASCII environment 130
  - specifying 127
- storclas data set creation attribute 119
- sync processing attribute 49, 127
- tag processing attribute 127
- text processing attribute 121
- trks data set creation attribute 116
- unit data set creation attribute 120
- vol data set creation attribute 120
- writetimeout processing attribute 128
- xlat processing attribute 128
- znfsclient processing attribute 128

server log data set
- setting up 523

server messages 319

server processes
- start number 142

server, bypasses saf checking 135

server, HFS vnode token 137

server, return first character '/' 138

server/client, user specified port range support 20

service level keyword 297

setattr request 44

setgid site attribute 148

setownernobody processing attribute 126

setownerroot processing attribute 126

setting time stamps 456

setup files, user specified port range support 20

sfmax site attribute 148

share options, cross-region and cross-system 119

share reservations 61

shareoptions
- managing VSAM data sets with 6

shareoptions data set creation attribute 119

shell environment 77

shortcut keys 541

showattr command 25, 50, 68, 72, 84, 99
- installing 198
- syntax for DFSMS 79
- syntax on AIX 65
- terse option 73
- to see logout value 69

showmount command 98
- syntax for DFSMS 81

shutdown of the client 212

shutdown of the server 217

side file
- allocated storage 148

side file data set
- for file extension mapping 40

side file processing attribute 126

side files
- forces 229

sidefile processing attribute 40

site attribute
- attrtimeout 139, 140
- bufhigh 139
- cachewindow 134
- checklist 135, 161
- delegation 135
- denyrw 135
- dhcp 135
- DlyDtimeout 135
- fileidsize 135
- fn_delimiter 136
- hfsfbtimeout 137
- hfsprefix 136
- HFSPREFIX 9
- hfssec 137
- impprefix 138
- IMPPREFIX 9
- leadswitch 138
- leasetime 139
- logicalcache 139
- logout 139
- maxrdforszleft 139
- maxtimeout 139, 140
- mintasks 140
- mintimeout 139, 140
- mixcase 140
- mvsprefix 140
- MVSPREFIX 9
- mvssec 141
- nfstasks 142, 280, 284
- NFSV4DOMAIN 143
- nlm 143
- noattrtimeout 139
- nochecklist 135
- nodelegation 135
- nodenyrw 135
- nodhcp 135
- noleadswitch 138
- nomaxtimeout 139
- nopcnfsd 143

site attribute *(continued)*
- nopublic site 144
- noreadtimeout 139
- norec878 146
- noremount 146
- nostringprep 149
- overflow 143
- pcnfsd 143
- percentsteal 144
- public site 144
- pubsec 145
- readaheadmax 145
- readdirtimeout 145
- readtimeout 140
- rec878 146
- remount 146
- restimeout 146
- security 147
- setgid 148
- sfmax 148
- smf 149
- stringprep 149
- upcase 140
- writetimeout 139, 140

site attributes 134, 151, 281

SMF (System Management Facilities) 165

smf operand 231, 236

smf site attribute 149

smfwtm macro 166

snoop trace for SUN client 529

snoop utility 317

socket hang
- specifying dump for 236

sockhang operand 236

soft client attribute 109

Solaris
- mvslogout command for clients 76
- showattr command for clients 72
- umount command for clients 76

source code
- downloading to the client 501

space data set creation attribute 119

space, unused 119

spanned data set creation attribute 119

specifying
- NFS attributes 171
- the exports data set 172

srv_ccsid client attribute 109

srv_ccsid processing attribute
- native ASCII environment 130
- specifying 127

stale NFS file handle 232

start command 173, 217

start number
- server processes 142

start PCNFSD server 143

startfail
- operand of the START mvsnfs command 217

starting
- NFS client 211

starting an MVS session 84

starting the NFS NSM, NLM 218

starting the server 215

startup procedures for z/OS NFS client
- in NFSSAMP library 499

startup procedures for z/OS NFS client *(continued)*
samples 499
startup procedures for z/OS NFS server
in NFSSAMP library 495
statd protocol 463
state 7
statelessness 7
statfs request 44
statistics, SMF 165
status operand 231
stop command 217
stop operand 218
stopping the client 212
stopping the server 217
storage block 240, 250
storage block sample, GFSAUDSA 239
storage class
for file creation 119
storage considerations 284
storage limit
NFS client 134
storclas data set creation attribute 119
stringprep
definition 21
site attribute 149
stringprep client attribute 109
striped files, SAM 29
subtype 7, record type-42 166
subtype 8, record type-42 166
Sun
downloading client commands 200
gcc 3.4.x compiler command 205
make command for 202
Sun Studio 11 for Solaris 10 205
SUN
client activity trace 529
snoop trace 529
suspending mount processing 228
swapldb operand 231
swapmhdb operand 231
switchlog operand 237
symbolic link
symlink procedure 44
sync processing attribute 49, 127, 280
synchronous write to z/OS UNIX file 49
syntax diagrams, how to read xvi
SYSERR DD statement 523
sysplex
multiple NFS servers in 217
SYSTCPD DD statement
not required 495
system authentication (authsys) security 158
System Authorization Facility (SAF) 158, 160, 161, 163
System Display and Search Facility (SDSF) 272
System Management Facilities (SMF) 165
system toolkits 204, 206

T

tag processing attribute 127
tar utility 199
tbytes value 457
TCP/IP
activity trace 527
TCP/IP tuning 269
tcpip.ETC.RPC, modifying 185
tcpip.profile and /etc/services files, server setup 20
tcprcvbrfsize 269
tcprecvbfrsize 268
tcpsendbfrsize 269
terminal ID
restricting mvslogin by 191
terse option
showattr command 73, 100
tested NFS clients 14
text mode
blankstrip processing attribute 122
end-of-line specifiers 122
noblankstrip processing attribute 122
text processing attribute 121
text processing mode 34
text processing, z/OS UNIX System Services 52
tfiles value 457
time stamps 453
time_delta value 457
time, data set released after a write 128
timeo client attribute 110
timeo value 280
timeout 70
maximum 139
minimum 140
timeout attributes 128
timeout option 263
timeout, control 137
to end access to MVS data sets 67
toolkits
development 205
system 204, 206
toolkits, development 203
touch command
for AIX 202
for UNIX 202
trace
client activity trace 528, 529
collecting messages for 233
HFS activity trace 527
SUN snoop 529
TCP/IP activity trace 527
z/OS UNIX activity trace 527
TRACE CT command
for z/OS NFS client 212
for z/OS NFS server 218
traceroute utility 317
trailing blanks 122
translation table
customized 128
for z/OS UNIX 184
translation table, for z/OS UNIX System Services 52
translation tables 183
Transmission Control Protocol/Internet Protocol (TCP/IP) 3, 215, 316
transport protocol
specifying 109
trks data set creation attribute 116
trusted attribute 153
TSO ALLOCATE command 87
TSO HELP MOUNT command 77
TSO HELP UNMOUNT command 77
TSO MOUNT 87
TSO MOUNT command 87, 164

U

UDP checksum processing 266
udprcvbrfsize 269
udprecvbfrsize 268
udpsendbfrsize 269
UID
mapping to user name 504
umount command
syntax on AIX 65
Unicode 112
Unicode Services
creating conversion environment for 164
required for text translation 130
Unicode Standard
for data conversion 86, 164
unit data set creation attribute 120
UNIX
command for mount operation 42
command reference 66
file name delimiter 136
mkdir command 25
mount command example 42
mvslogout command for clients 76
NFS client 63
permission bits, checking 159
showattr command for clients 72
umount command for clients 76
vi command 70
UNIX permission bits, checking 147
UNIX-style credentials 164
unmntall operand 232
unmnthfs operand 232
unmntmvs operand 233
unmount (immediate, force) 93
unmount all mount points 232
unmount command 77, 92
unmount HFS mount points 232
unmount legacy mount points 233
unmount operand 232
upcase site attribute 140
updating your MVS password 83
updating your z/OS password 67
upper case file name
mapping to lower case 124
upper case, message display 140
usage statistics, collecting 165
user authentication, procedure 13 505
user exit block (UXB) 241
User Exit Block (UXB) 244
user group (UIG) 68
user number (UID) 32, 50, 55, 68
user specified port range, set up 186
user storage block, login exit 240
user-specified port range support 20
UTF-8 data format 21, 52

V

v4delg operand 233
validation of MVS passwords and user IDs 160
Variable Recording Area (VRA) 314
vers client attribute 110
vers mount parameter
  tuning client with 289
version operand 237
vi command
  creating a physical sequential (PS) file with 24
  creating PDS or PDSE member with 26
  editing files with 33
vi editor
  creating VSAM files with 27
vnode tokens 137
vol data set creation attribute 120
volume
  for data set creation 120
VSAM (virtual storage access method)
  access to remote files 4, 87
  line delimiter for record access 106
  serializing and sharing data sets 6
VSAM data sets
  creating VSAM files 27
  keys for KSDS data set 117
  record size 118
  serializing and sharing data sets 6
  supported by z/OS NFS server 23
  time stamps for 453, 454
  using fastfilesize for 451
VSAM KSDS data set
  attribute for creating 117
VSAM RRDS data set
  attribute for creating 117
VTAM 316

W

wait keyword 296
WebNFS support 17
wildcard characters
  specifying in the exports data set 180
wildcard copy restriction 27
window size 134
Windows
  command for mount operation 42
  mount command example 42
  umount command for clients 76
write operation
  writing a file on MVS 33
write request, nonvolative media 127
write statistic 264
writetimeout processing attribute 128, 279
writetimeout site attribute 139, 140
writing
  fixed-length records 35
  text data files 35
wsize 275
wsize client attribute 110
wsize mount parameter 289
wtmax value 457, 458
wtmult value 457, 458
wtpref value 457, 458

X

X'0F3' ABEND 314
X'806' ABEND 314
X'A03' ABEND 314
X'x13' ABEND 314
X'x22' ABEND 314
X'x37' ABEND 314
X'x3E' ABEND 314
XDR (External Data Representation) 4
xlat attribute 183, 184
xlat client attribute 110
xlat processing attribute 128
XLC v8
  for AIX 5.3.x.x 205

Z

z File System (zFS) 277
z/OS
  password and user ID 67
  showattr command for clients 72
z/OS Basic Skills information center xix
z/OS DFSMS
  command syntax for 77
z/OS NFS client
  operating 211
  overview of 3
  sample startup procedures 499
  starting 211
z/OS NFS server
  overview of 3
  restricting mvslogin to 191, 192
  sample startup procedures 495
z/OS UNIX
  activity trace 527
z/OS UNIX attributes 49
z/OS UNIX file system 47
z/OS UNIX segment 503
z/OS UNIX System Services
  accessing 51
  delete external link 54
  display external link 54
  external link to MVS 53
  mount command 51
  overview of 3
  security 50
zLinux
  make command for 202
znfsclient processing attribute 128
zSeries File System (zFS) 47

Program Number: 5694-A01

Printed in USA

SC26-7417-11